AUTOMATED KNOWLEDGE EXTRACTION FROM TEXT

Paul Richard Bowden

A thesis submitted in partial fulfilment of the requirements of The Nottingham Trent University for the degree of Doctor of Philosophy

March 1999

Acknowledgements

I would like to thank my supervisors Peter Halstead and Lindsay Evett, and my fellow researchers Mark Edwards and Gavin Long, for their guidance, support and encouragement during the research reported here. I would also like to thank my wife Linda for her support during the four years this research took to complete.

Abstract

Knowledge Extraction (KE) is the automated extraction of facts from machine-readable text. KE is a branch of Natural Language Processing (NLP). Within NLP, processing techniques may be *deep* or *shallow*. Deep techniques are the traditional methods of NLP and computational linguistics, and are aimed at language understanding. They are mostly domain independent techniques. Shallow techniques are currently a focus of interest and may be defined as methods which achieve NLP goals without recourse to attempts to understand fully the input text. These are mostly domain specific techniques.

Deep processing approaches are considered with respect to the problems they entail. These problems can be both theoretical and practical. These and other difficulties are used to justify shallow attempts at NLP tasks. After a review of several existing KE and similar systems this work describes the knowledge extraction program developed by the author (KEP). KEP aims to be shallow and non domain specific, and extracts factual knowledge from explanatory texts. A pattern-matching approach is used which cuts fact-bearing sentences into fragments so that concepts and the facts relating to them can be extracted. Various *conceptual relations* are searched for, including at present *definitions* (definitions of concepts), *hypernyms* (parent classes of concepts), *exemplifications* (examples of concepts) and *partitions* (lists of the component parts of a concept).

One of the motivating factors for doing this research was the desire to answer the question: *how useful can a specific set of shallow techniques be in a non domain specific NLP application?* This is an important question at a time when shallow techniques are viewed favourably by the NLP community. To this end, the performance of KEP has been evaluated using the *recall* and *precision* measures. As a final demonstration of the program's abilities, KEP has also been run on a large part of the text from this work to produce a first-cut glossary for that text. This glossary successfully captures the main concepts from the text and provides useful explanations of them in many cases.

It is concluded that KEP is a working program which demonstrates the usefulness of shallow, non domain specific methods, and which has opened up the possibilities of several new research directions, including automatic index creation, student assignment marking, and information retrieval from the Internet for the automatic construction of semantic-net knowledge bases.

CONTENTS

List of Figures

List of Tables

1. Introduction

1.1 Natural Language Processing and Knowledge Extraction

1.1.1 What is Knowledge Extraction?

Knowledge Extraction (KE) is the process of obtaining knowledge from text. Human readers are able to perform KE almost effortlessly, but the term KE is used in this thesis to refer to KE by computer program. This thesis discusses a novel KE approach, which has been realised as a computer program. In general there are two basic approaches to the KE task: *shallow* and *deep*. Deep processing involves the use of the full range of techniques and resources available to the traditional natural language processing (NLP) researcher, such as full parsing. Deep techniques intend to *understand* the input text. Shallow processing on the other hand aims to achieve its goals in a faster, simpler manner, without the need for the whole panoply of traditional techniques. Shallow approaches rarely attempt to understand the input texts. These themes will be expanded upon shortly. For now, it is enough to state that the KE program introduced in this thesis aims to be a *shallow* system. Furthermore, this system, which is called KEP (for Knowledge Extraction Program), aims to be independent of the subject domain of the input text, i.e. it is a non domain specific (NDS) system.

The research reported upon here explores the limits of a shallow non domain specific system. In particular it is argued that the deep approach to NLP in general, and KE in particular, involves many difficulties which make it worthwhile to try shallow approaches instead. One of the major goals of this research is to see how far a specific set of shallow techniques can go for NDS knowledge extraction. This entails building an actual computer program to test the proposed techniques. A second goal of the research is to create a practical new program which could be incorporated into existing software tools (such as word processors, WP) to perform KE in a useful way. For example, a completely automatic glossary maker would be a useful WP feature. Such a feature does not currently form part of any commercial WP package. The KEP system described in this thesis makes good progress on both of these goals.

Knowledge extraction is an exciting and challenging new discipline. It is challenging because at first sight it would seem that only deep methods could work, since it appears reasonable to assume that a KE program must *understand* the text from which knowledge is to be extracted. Deep methods are difficult and time consuming to develop, and so it would seem that KE must also be a difficult goal. (It is a theme of this thesis that this is not necessarily the case.) KE is also exciting because, if successful KE programs could indeed be developed, a whole range of genuinely useful new applications and features would arise. For example, in the domain of word processing, such features include the completely automatic creation

of document indexes, glossaries and summaries. In an age when people are swamped with vast amounts of text, much of which may be irrelevant to the reader's needs or interests, programs which could cut down this textual mountain to a readable hillock would be invaluable. This is particularly apposite in these days of the Internet. Searches on the World Wide Web (WWW) often return thousands of document titles, and although the search engines attempt to order these by relevance, such ordering is based solely upon keyword matching at present. How much better it would be if the search engine could return "more of the same" documents based upon the topic of the text, as indicated by the knowledge it contains.

But KE possibilities are not limited merely to the domain of text processing. Many anticipated computer systems require a knowledge of "what the master wants". Thus video cassette recorders might tape those programs which interest their owners, houses set the environmental conditions to suit the inhabitants, cars adjust automatic-gearbox change-up points to suit the driving style of a particular driver etc. Where the knowledge involved is written, a KE program may compare the user's choices with the written descriptions. Thus for example the VCR might compare TV programmes actually watched by its owner with the descriptions attached to programmes in the electronic TV listings guide, and hence determine which forthcoming programmes will probably be of interest to its owner, so that it may tape them without being specifically instructed to do so.

The list of potential KE applications is huge and varied, and new ideas are added continuously. For example, companies swamped with CVs in response to job advertisements need to pre-process them automatically, specialised news agencies want to automatically prepare articles from newswire feeds, administrators of databases of scientific papers require consistent abstracts from all of them, overloaded university lecturers want systems to pre-mark hundreds of student essays, company executives want accurate summaries of thick reports, historical researchers want systems to find articles on specific incidents or themes, booksellers want to tell their customers about books which might interest them, and direct-marketing organisations want to better target their mailshots in order to reduce waste and minimise public hostility. KE systems may eventually provide solutions to all of these needs, and indeed to many scenarios not yet envisaged. This is why they are exciting systems worth attempting.

1.1.2 Artificial Intelligence, Natural Language Processing and Knowledge Extraction

Artificial Intelligence (AI) is an interdisciplinary subject which aims to build computer systems having the appearance of intelligence. AI systems may be genuinely intelligent, or may merely appear to be intelligent; either way, they display characteristics of an intelligent entity to some degree. Intelligence is extremely difficult to define, but it is relatively easy to identify a system which is *apparently* intelligent within its application domain. Intelligence is a property possessed by humans, and so Rich and Knight (1991) have defined AI as "the study of how to make computers do things which, at the moment, people

do better". Rich and Knight admit that this definition is a deliberate attempt to sidestep the issue of defining *intelligence* or *artificial*, and recognise that their definition is ephemeral (for it contains the deictic phrase *at the moment*), but suggest that it "provides a good outline of what constitutes artificial intelligence". With a touch of humour, Rich and Knight point out that unlike other new fields such as physics (which broke away from philosophy and grew as a separate area of endeavour) the field of AI as defined above may one day, if it progresses well enough, reduce itself to the *empty set*. It is a strange idea to have a field which shrinks as it progresses, so perhaps the definition given in the first sentence of this paragraph is the better one, i.e. AI aims to build *apparently intelligent* systems. This definition places its emphasis on the simulation of intelligent behaviour, rather than on questions of whether the AI program is "really" intelligent or not.

Several traditionally separate academic fields are interested in intelligence, both human and otherwise, and so AI practitioners have come to include linguists, psychologists and computer scientists, amongst others. Linguists are involved because the use of human language is inseparably bound up with the property of intelligence – it seems that in order to use language one needs to be intelligent, and yet conversely it would appear that in order to be intelligent (at the human level) one needs to be able to use some kind of language. Psychologists are interested in human behaviour, which is likewise inextricably linked with the attribute of intelligence. Finally, computer scientists, engineers and ergonomists have recently (within the last few decades) become interested in the idea of simulating human intelligence, for both theoretical and practical reasons. The relatively new interdisciplinary field known as *cognitive science* attracts all such interested parties. More recent joiners also include neuroscientists, who are interested in how the human brain actually does what it does, and whether AI can help in the understanding of this vastly complex organ.

It is a matter of great debate as to whether a computer simulation of some aspect of intelligence can say anything about real human intelligence, but this is not a relevant topic for this thesis. Instead, this thesis is concerned more with the *possibility* of simulating a specific intelligence-requiring task rather than in debating whether such a simulation says anything about how a human thinks. The interest here lies with one of the major sub-fields of the AI discipline: language use. Natural Language Processing (NLP) is the branch of AI which concerns itself with the processing of human languages (as opposed to computer programming languages, which are un-natural in the sense that they were invented for a specific purpose, that of communicating to (but not from) computers). Knowledge Extraction (KE), the major topic of this thesis, is itself a branch of NLP, since it involves the extraction of facts from texts written in natural languages. The field of NLP also includes endeavours such as natural language interfaces to computers (NLI), and machine translation from one natural language to another (MT), both of which are motivated

not only by theoretical interest but also by the potentially huge practical benefits to be gained from successful simulation of intelligent human behaviour[1].

NLP has always been one of the driving forces within AI, but it has also captured public imagination to an unusual degree. Beloved of science fiction writers, the ability of computers to understand English was for a time regarded as something inevitable; in "the future" (it was thought) we will all be able to converse freely with computers, freeing us all from the need to learn complex programming languages and keyboarding skills. Alas, progress has not been anywhere near as straightforward as the enthusiasts of the fifties and sixties expected. The history of the discipline of AI in general is one of enormous initial excitement and optimism followed by a growing realisation of the difficulties involved (together with a corresponding rise in pessimism and a fall in funding), but leading eventually to a new pragmatism concerning what is or may be achievable. A new sense of cautious optimism today pervades the discipline. This story has been told admirably by various exponents of the field and by interested journalists (see e.g. Crevier (1993), Rich and Knight (1991)) and will not therefore be expanded upon here.

For the purposes of the research reported upon here, the 'NL' in NLP is the English language. However, some of the earliest NLP programs were motivated by the desire to translate from one natural language to another. The early experimenters in the MT field quickly came to realise that the problem was much more difficult than most had envisaged. A significant subset of the problems which arose in MT also exists for KE, and indeed for all NLP fields. Many of these problems arose because a deep processing approach had been taken, either through necessity or choice. In a later section of this chapter some of these problems will be discussed, with the aim of demonstrating that a deep approach should not be taken if there is the possibility of using a shallow method.

1.1.3 Concerning the Nature of Knowledge

1.1.3.1 Introduction to Knowledge Categorisation

What is this **knowledge** which KE aims to extract? Many AI practitioners deal with this question simply by listing rhetorical questions such as "what do people have inside their heads when they know something?", rarely attempting to actually answer these questions (see e.g. Sowa (1984)). In the following sections an attempt is made to discuss what knowledge might be, and to introduce different ways of categorising knowledge. This is a difficult task. Although knowledge may come in different types, detecting and categorising a particular piece of knowledge is not simple.

[1] The importance of *simulation* in AI is reflected in the title of an AI society, the SSAISB (Society for the Study of Artificial Intelligence and the Simulation of Behaviour).

One valid starting point is to regard knowledge as being made up of *facts*, and this is the approach taken below. Since this thesis is concerned with knowledge as it is held in texts, rather than with knowledge as it might exist within a person's head, the following discussions are biassed towards the former. This is a reduced view of knowledge because there are undoubtedly many types of knowledge which cannot be represented in textual form, such as the *learned behaviour* knowledge which allows a person to drive a car without having to think consciously about every single movement of their limbs as they do so. This thesis is not about such types of knowledge. It is about the type of knowledge which one human wishes to convey to other humans via text. Thus *facts*, which are usually easily expressed in natural language, are of prime interest.

What is a *fact*? Philosophers distinguish *facts* from *values*, i.e. *what is* from *what ought to be* (Collins Dictionary of Philosophy, Harper Collins (1990)). Thus philosophers view facts in a similar manner to the commonly held perception. For our purposes, a fact may be defined simply as a true statement about the universe or its contents. (Rich and Knight (1991) use the phrase "truths in some relevant world".) AI is largely concerned with how facts can be represented (the issue of *knowledge representation*, or KR). Thus systems such as propositional logic play a part in many AI systems; indeed, much of AI is concerned with translation from one KR (e.g. natural language) to another (e.g. logical statements). However, the medium of KR is fixed for the research reported upon in this thesis – it is English text. Thus we require a definition of *fact* for the textual medium. The problem with the simple definition given above is that it does not allow for statements which are believed to be true by an author and yet are false in reality. Therefore, a better definition for a fact in text would be: *a statement asserted to be true by the author of the text.* The assertion need not be explicit (e.g. "I assert that the electron is a lepton") but may exist implicitly within the statement (e.g. "Electrons are leptons."). It is also not necessary for the author to *believe* the statement to be true – merely to assert it.

This definition of a fact as found in text also allows us to bypass philosophical doubts concerning the very existence of facts. Harré (1972) points out that the inductivist school's principle that science grows as an accumulation of facts simply will not do, because "facts" are no such thing in reality: "a change in theory can change seeming facts into falsehoods". A Kuhnian paradigm shift (Kuhn (1970)) may well force us to re-interpret a "fact" from the superseded theory, even if the experimental evidence which gave rise to that fact remains unchanged. Furthermore, if we accept the Popperian view that we can never be one hundred percent certain that a theory is correct (as does the author of this thesis), then it follows that there is always scope for "facts" to change (see Popper (1972)). But such problems do not concern us here; as far as this work is concerned, a fact is something asserted to be true in a text. Furthermore, we shall not become diverted at this point as to the various meanings attached to the word "knowledge" by philosophers. (Interested readers are directed to a concise summary of the meanings of this term in Collins Dictionary of Philosophy (1990)). We shall stick to our definition of knowledge as a collection of facts.

1.1.3.2 Knowledge vs. Information

If we start with the assumption that knowledge is an aggregate of facts as defined above, then it is useful to ask whether *all* facts may be used. Not all types of fact may be useful, so we must ask what sorts of fact are knowledge-like. There is an intuitive feeling that knowledge tends to be about "important" issues, and also that it tends to be about "how things are". Thus the types of fact involved in knowledge should reflect these ideas. One way of allowing this is to consider only those facts which describe longstanding situations. Such situations tend to reflect "how things are" because they exist over extended timespans and so always return the same answer to the question "how are things?". They also tend to describe important issues, because situations which are stable over long periods of time (e.g. comparable with a human lifespan) must be taken into account in human affairs, i.e. they must be built into the human world-view.

To this end, **knowledge** may be defined as a collection of facts which are true for extended timespans. By this definition, the phrase *atomic nuclei are composed of protons and neutrons* contains knowledge. As far as we know, this situation has always been true and it will always remain so. Similarly, *PASCAL is a high-level language* is true, and has always been so since PASCAL was invented. By appending the phrase 'for extended timespans' to 'facts' we rule out *fleeting* facts. In the sentence *this package contains three separate manuals* there is a fact, but this would not generally be regarded as knowledge. The fact given in this sentence is fleeting. What constitutes "fleeting" is of course subjective, but the above statement does not seem to be something which could be regarded as true "for all time".

This problem is linked to the issue of whether historical facts should be included in the definition. Most people would argue that historical facts *en masse* constitute knowledge, even though they describe fleeting events. However, even historical facts are true for extended timescales. The statement *King Harold died when hit in the eye by an arrow* may describe a fleeting event, but it is nevertheless true for all time in the sense that it will always be a true statement.

As mentioned above, there is also the underlying implication of some sense of *importance* relating to those facts which are part of knowledge, and the 'package' sentence does not have this (unlike *atomic nuclei are composed of protons and neutrons*). The fact held within *this package contains three separate manuals* is really **information** rather than knowledge. Information is distinguished from knowledge in that it is *intended to be used within a short time after its reception.* Information is conveyed for a specific purpose. It can become out of date. It is not a true-for-all-time fact which is worthy of inclusion in an encyclopaedia. Information may arrive in textual form, or it may be numerical ("data").

It is interesting to note that those practitioners who are attempting to build computer systems capable of extracting facts from newswire streams and the like also use the word 'information' in this sense. Their

research area is usually known as IE, for *information extraction*. The facts they attempt to extract (with some fair degree of success) relate to pieces of information such as "Henry Smith has just been appointed the new chairman of company A" or "company A has just taken over company B". Such historical facts may be true for all time in the sense that once an event has happened it cannot un-happen, but they are still fleeting in the sense that the information has a shelf-life (a period during which it is useful information, i.e. a period during which the purpose of conveying the information is actionable by the recipient). Thus, although the information may be very important to a particular group of people, it generally does not convey those properties which would allow it to be classified as "part of all human knowledge".

Knowledge as defined above may be extremely specialised. This can sometimes make it seem more like information than knowledge. In these cases, a reader may have difficulty in deciding whether a piece of text contains information or whether it contains knowledge. For example, consider a computer program which attempts KE in a restricted domain such as that of computer printers (e.g. Reimer (1989)). Such a program might regard some statement about a specific printer as knowledge to be extracted and placed in a knowledge base. (In one sense, anything placed in a knowledge base is by definition knowledge, but this is merely a linguistic trick arising from the decision to call the fact-repository a 'knowledge base', so we may ignore this argument.) In this domain, one might regard a statement such as *the DMP-55 printer allows full-colour A3 printing* as information rather than as knowledge. Perhaps this feeling arises because the printer itself has a finite product life, or perhaps it arises because computer printers are not seen as fundamental to the way the world works. For whatever reason, it is not easy for a human to state with certainty whether the 'printer' statement above contains knowledge or information.

In summary then, **knowledge** may be regarded as *a collection of true-for-all-time facts*, whereas **information** comprises *immediately useful* facts or data *which may become false in the near future*. The distinction between knowledge and information is not always clear cut even for a human reader, and so it would not be surprising if a KE program had difficulty in distinguishing the two. Whether it is important for a KE program to be able to distinguish knowledge from information will largely depend upon the application. It is likely to be less of a critical issue for an automatic glossary maker, for example, than for an automatic encyclopaedia constructor.

1.1.3.3 Episodic Knowledge

NLP research distinguishes a class of knowledge called *episodic* knowledge (see e.g. Burkert (1995)). The term is actually used in two different ways as follows. Firstly, specific instances of concepts are episodic. The phrase *Fido is a dog* demonstrates episodic knowledge, but *dogs are mammals* does not. In other words, specific instances of things (such as dogs) are episodes rather than all-time facts.

This type of episodic knowledge is not of use to all KE systems, such as those interested only in classes of objects. For other KE systems, this kind of episodic knowledge is perfectly acceptable. For the latter, a statement such as *An example of a tall building is the Empire State Building* would certainly be regarded as a fact worth extracting, despite the point that this statement describes a specific instance of the tall-building concept. KEP does attempt to extract such facts, since *examples* of classes are regarded as useful facts.

More problematical, however, is the second mode of usage of the term *episodic knowledge*. This is where the knowledge occurs as *historical facts*, i.e. facts describing past episodes. Such facts are often the target of IE programs. Newswire reportage is not the only type of reporting, however. For example, Bross, Shapiro and Anderson (1972) describe the scientific sublanguage used by surgeons to report upon operations they have performed. These reports are essentially lists of descriptions of episodes, couched in a concise and unambiguous (to surgeons) sublanguage. This sublanguage utilises a constrained vocabulary, standard phrases used by all surgeons (e.g. many reports end with the standard phrase 'the patient left the operating theatre in good condition') and certain standard syntactical features such as the use of the passive voice. Specific syntaxes are also used to indicate causation and the order in which events occurred, such as with the *temporally-follows* relation, which often uses the pattern *with the <nominalisation>*, e.g. as in '*with the excision* of the tumour...'. The point here is that such texts do **not** contain knowledge-type facts as defined above. Such texts are not explanatory; they are historical narratives. By their very nature they will not contain many definitions, part-whole descriptions etc, since experienced surgeons do not need to tell each other what a myocardial infarction is, or what the major parts of the heart are. Such facts are already part of the surgeons' knowledge. Swales (1981) has also pointed out that texts such as these, which are "high brow" i.e. between experts, are less likely to contain definitions than middle- or low-brow texts designed by experts for lower-status readers, and this viewpoint is also supported by Darian (1981), who presents five levels of material based upon writer-reader degrees of specialism.

Although such historical reports may be legitimate source texts for domain specific KE systems aiming to summarise their contents, e.g. by constructing a standard abstract where textual gaps are filled from sets of allowed role fillers (see e.g. Oakes and Paice (1998)), they are not generally useful for fact extraction KE programs. Since the purpose of the research reported in this thesis is primarily to investigate the possibility of creating an NDS fact extraction system, historical texts of the surgeons' report type will not be used as input.

1.1.3.4 Generic vs. Specific Knowledge

The discussion above has already touched upon the distinction between individual objects and classes of objects, i.e. between *generic* and *specific* items. It is usually the case that generic facts are more knowledge-like than facts about specific single objects. For example, facts about cars in general are

knowledge-like, as are facts about a specific type of car (such as the Ford Mondeo). One could imagine such facts appearing in the glossary section of some document. On the other hand, facts about Mr. Smith's car, a specific instance of a car, are not likely to be useful in a glossary. Clearly the degree of importance ascribed to an object matters; in a text about the SALT talks a specific (critical) meeting between Russians and Americans might well be detailed in a glossary.

This issue is tied closely to that of information vs. knowledge, for specific-object facts tend to look more like information and generic facts more like knowledge, as defined above. Thus although an IE system might wish to extract specific-object related facts, KE systems in general will not.

1.1.3.5 Declarative vs. Procedural Knowledge

A further division in types of knowledge has been given by Skuce et al. (1985). Here the distinction is made between *declarative* and *procedural* knowledge. The former is equivalent to factual knowledge, but the latter concerns knowledge of procedures. Thus *the bracket is held on by a nut* is declarative, but *to remove the nut, perform steps 1 - 3 as follows: 1)* ... is procedural.

Procedural knowledge is less likely to be present within a single sentence. It is not the target of the current research, since the inclusion of procedural KE would broaden the attempted KE task unacceptably. Procedural KE is however an established research field; e.g. automatic construction and understanding of instruction manuals are established areas of research (see e.g. Vander Linden and Martin (1995), Sutcliffe et al. (1995), Skuce et al. (1995)).

1.1.3.6 General Knowledge and World Knowledge

However knowledge is defined and categorised, it is certainly true that all human beings hold large amounts of it in their heads. Much of this knowledge is applicable only to certain tasks or domains ('specialist knowledge', or domain-specific knowledge), and much is regarded as 'general knowledge'. In human terms, 'general knowledge' usually means "facts which most reasonably educated people have at their disposal", such as the names of capital cities, the names of famous people, historical facts, names of types of animals etc.

In NLP terminology, knowledge about the world in general is termed World Knowledge (WK). WK is deemed to be essential for good NLP programs, and this is considered in the following section. It is worth noting, however, that WK as used in NLP programs is not quite the same thing as traditional human general knowledge. WK includes facts about the world which are *so* obvious that a human would not even bother to classify them as general knowledge. Such facts are often about physical laws, as evidenced in our four-dimensional environment. Some examples of such facts will be presented shortly.

In the discussions which follow, the term WK will be used to refer to both knowledge about the world required by a program, and knowledge about the world, of any sort, held by a human.

1.1.3.7 Knowledge and KE Programs

It is paramount that a developer of a KE program has a clear idea of the type(s) of knowledge which that program will extract. Since knowledge as defined earlier is composed of facts, then it is likely that KE programs will attempt to extract individual facts from the input texts. As has been discussed above, the types of fact to be extracted are likely to depend upon the KE application. This thesis concerns a specific KE application (chosen for its interest, challenge, potential usefulness and well-defined boundaries) and so the discussion will now be confined to that approach. Since the major application chosen in the research reported here is the automatic construction of a glossary, then facts which might appear in a glossary are the target facts. What sort of facts might these be?

Chambers English Dictionay (1988 edition, W. R. Chambers Ltd. and Cambridge University Press) defines a *glossary* as "a collection of glosses: a partial dictionary for a special purpose." A *gloss* is defined as "...an explanation... a collection of explanations of *words*" (author's italics). The *words* which are to be explained clearly *require* explanation, e.g. because they are specialist or technical words within the domain covered by the glossary. Thus they are *terms*. Therefore glossaries comprise lists of terms present in the text, together with explanations of those terms. Thus a fact in a glossary is a term-explanation pair. In this thesis the word elucidation is preferred to 'explanation', since it is broader in scope (an explanation tends to say "what something is", whereas an elucidation may additionally give facts relating to the term or characteristic of it). Examination of glossaries from a variety of documents show that elucidations include *definitions* of the term, *examples* of the concept represented by the term, class-inclusion information (e.g. that the term *is a type of* something), *parts-and-pieces* information, *characteristics* of the concept represented by the term, alternative *names* for it, what it is *used for*, what *causes* it, what it is *made from*, and so forth. Thus these pieces of knowledge relating to the term are the ones which a glossary maker needs to extract.

Note that the terms given in a glossary represent concepts (tangible and intangible things) in the text. More than one term may represent the same concept (since there may be alternative names for it). Thus the elucidations in a glossary are actually for the underlying concept, or "abstraction", which the term represents. The different types of elucidation are usually called *relations*, and so the glossary comprises instances of *conceptual relations*. These conceptual relations will be introduced and discussed in a forthcoming chapter.

It is clear that the extraction of a term-elucidation pair must involve two steps: firstly, terms must be identified from the text, and secondly the relation between that term and its elucidation must be detected.

The former is an emerging research field (terminology extraction), but the latter appears to be largely confined to linguistic discussions in books and papers. In Chapter 4 of this thesis a detailed description is given of how KEP approaches these tasks.

Having introduced the types of knowledge that KE programs may need to extract, and more specifically the types extracted by a glossary maker, it is now necessary to return to a more general discussion in order to consider how this might be done.

1.1.4 Text Understanding and World Knowledge

The process of interpreting a printed text in the light of the reader's World Knowledge (WK), i.e. the process of extracting *meaning* from an utterance or text, is called *understanding* and has been discussed by NLP researchers such as Schank, Wilks and others (see e.g. Schank and Abelson (1977), Wilks (1975)). It is understanding which distinguishes deep approaches. The understanding process involves both semantic and pragmatic aspects. The former deals with the *possible* meanings of an utterance, and the latter feeds WK into this process and thus aids in the selection of *the correct* meaning. This process can be illustrated with an example. In a sentence such as *All the bananas are blue!* the semantic content is in contradiction to the pragmatic knowledge that (unless you have painted them etc) no bananas are this colour. The conflict with reader-WK created by the above sentence would cause a human reader to search the text (or the burgeoning knowledge-construct already built in the mind of the reader at this point – see e.g. Kieras (1982)) for other information which might explain the semantic content of the unexpected sentence, i.e. for other information to aid understanding. This illustrates the need for WK in the understanding process.

The traditional route to text understanding within the field of computational linguistics has been one involving a linear progression from lexical aspects of the text, through to syntax, to semantics, and thence to pragmatics. Grishman has provided an excellent introduction to these separate aspects from the viewpoint of a computational linguist (Grishman (1986)), and Allen discusses them from the NLP practitioner's perspective (Allen (1995)). We shall see, however, that a strict linear succession of processing stages as listed above is not sufficient for full text understanding. As has been hinted at above, it is highly unlikely that a human reader performs grammatical processing, then semantic processing, and then pragmatic processing in a sequential manner. It is more likely that all three levels are performed simultaneously, in the presence of a mental representation of "the story so far" constructed as a text is read. This theme is returned to shortly, when the difficulties of simulating simultaneity in a computer program are considered. This is just one of the difficulties of doing deep NLP. This and other obstacles are considered next.

1.2 Problems with the Deep Approach

1.2.1 Introduction

The purpose of this section is to introduce important issues involved in *deep* approaches to NLP, and by inference to deep KE. This will by necessity involve some description of deep techniques, such as that of automatic parsing. The difficulties encountered in such techniques will be examined. The purpose of this exercise is to demonstrate that the deep approach in general is afflicted with serious practical and theoretical problems not yet completely solved. The implication is that shallow techniques should be used wherever feasible.

1.2.2 The Problem of Message Information Content

Utterances, either spoken or written, do not contain all the information that they convey. This truth was recognised by early workers such as Shannon and Hartley (Shannon (1948), Hartley (1928)), working in the domain of information theory (sometimes called communication theory). Most NLP researchers re-discover this fact for themselves when they first attempt KE or MU. Human communication relies upon the vast reserves of WK that all adult people have access to, and this WK is used to extract the intended meaning from an utterance. Spoken or written messages contain *omissions*, because this makes them much shorter than they would be if all the relevant factors were described in detail. The recipient's WK is *compared with* or *added to* the message content, so that the gaps present in the message may be plugged (McDonald (1992)).

Furthermore, in the case of dialogues, utterances must be interpreted in the light of previous speech acts made since the start of the conversation, so knowledge of discourse conventions is needed. (Grammatical (syntactical) and lexical (vocabulary) knowledge are of course also required, being those parts of human knowledge traditionally regarded as language skills.) Discourse analysis is an important field because the meaning of an utterance depends upon its place in the discourse. However, it is not possible to go into it here (for an introduction see e.g. Brown and Yule (1983)).

A deep NLP system must be able to spot points where WK is required to complete the information content of the message, and then it must actually provide that extra knowledge and perform the addition to create the effect desired by the sender of the message. Thus a deep NLP system must include a reserve of WK, as well as a mechanism for applying it. However, this mechanism cannot be something as simple as a call to a function such as "get missing knowledge", because it would be impossible to know when to call that function. In a sense the mechanism must be active all the time, because it is needed to *spot* the gaps as well as to fill them. This "priming" function is difficult to simulate in a computer program, although some ideas have been put forward (see e.g. Hapeshi (1994), although this paper is concerned more with lexical priming than with semantic priming).

The need for WK and ways of using it are thus critical for deep NLP systems. However, even if such problems were to be solved, there would still remain another major problem related to WK use, described in the next section.

1.2.3 The Problem of KB Size

As has been argued above, no general-purpose NLP system which attempts to extract meaning from an utterance can be effective without some way of holding the necessary WK. The entity used to hold knowledge is usually referred to as a Knowledge Base (KB). We may ignore for now the various ways in which KBs can be structured, because these are irrelevant to the problem under consideration. The problem is that the KB needs to be huge. For example, Lenat's CYC project (Lenat (1995a), Lenat (1995b), Lenat and Guha (1990)) is an attempt to produce a useful KB for NLP applications. It has been running for over a decade and still has yet to reach the point at which it might be said to contain a useful amount of WK (for general purpose NLP). Much of the problem is caused by the need to encode "obvious" facts, such as the fact that we cannot remember things that have not yet occurred, or that once a person dies he stays dead, or that things fall downwards etc. It is remarkably easy for an NLP developer to overlook such WK, since humans take such knowledge for granted. (Recall the comments made previously concerning knowledge which is too obvious even to be classed as 'general knowledge'.)

It is hardly surprising that many years of work have been needed to build the CYC knowledge base; infants require many years of intense interaction with knowledgeable adults and their environment in order to reach similar levels of WK, much of this time being spent in deliberate attempts by mature humans to build up the new person's KB (i.e. in school and at college), but most of it spent in self-learning from the external world and objects in it. Young humans clearly maintain an intense energy to build up their knowledge bases - as evidenced for example by exchanges between the young child and the adult which start with a 'Why...?' question and continue with a string of secondary 'Why?' questions. This phase is possible only after the language learning phase has reached a certain maturity, but it is difficult and possibly erroneous to separate these two types of knowledge acquisition. They are both large and fascinating subjects in their own right, although it is not possible to go into them here. (Interested readers should consult e.g. Pinker (1994) for a populist account of language acquisition, and Karmiloff-Smith (1992) for an account of mental development in the child.)

In conclusion, the provision of adequate WK represents an enormous practical problem for all NLP researchers who are not taking a shallow approach to their tasks. It may be, of course, that this problem essentially only needs to be solved *once*. If this is the case, then the CYC project and others like it may one day be regarded as worth the huge investment in time, effort and money. It remains to be seen, however, how long this "one-off" solution will take to create.

1.2.4 Problems of Parsing

A non-trivial obstacle to the provision of deep NLP concerns the grammatical function (syntactical processing) required. NLP uses parsers because the meaning of a sentence is dependent not only upon the actual words used but also upon the way in which they are ordered and grouped together. Traditional grammar books exist, as constructed by linguists and grammarians (for example Quirk et al. (1985)). However, it is difficult to translate such knowledge into a form suitable for a parsing program. The difficulties relate both to theoretical and practical matters. On the theoretical side, the grammars described may not be complete, in the sense that they omit certain constructions or combinations of features. Practically, it requires much effort to extract the rules given in a grammar book, convert them into a usable form, and insert them into a computer program.

In order to better understand these problems of parsing, it is necessary to describe a parsing technique. The description which follows will be kept as short and as simple as possible whilst still allowing the essential points to be made.

Many parsers, notably those referred to as rule-based parsers, use a set of *production rules* to analyse sentences. These rules define the set of allowed sentence forms for a particular language by permitting repeated replacement of sentence elements. The application of a production rule is called a *production.* For example, if we use the symbol S for sentence, NP for noun phrase, VP for verb phrase, N for noun, V for verb, ADJ for adjective, and ADV for adverb, then a grammar for a language might be specified as:

S → NP VP
NP → N
NP → ADJ N
VP → V
VP → V ADV

In the above, the pattern on the left of the re-write rule may be replaced by that to the right of the arrow. Then given a list of *terminal symbols* (instances of nouns, verbs, etc, such as *dogs*, *attack* etc) sentences such as *Fierce dogs attack ferociously* may be constructed. Figure 1 shows a parse tree for this sentence, which may be regarded as a diagram showing the grammatical structure of the sentence, or alternatively as a pictorial way of showing how the production rules have been applied (S in the top row is replaced by NP and VP in the second row, then NP in the second row is replaced by ADJ and N in the third row, and then VP in the second row is replaced by V and ADV in the third row).

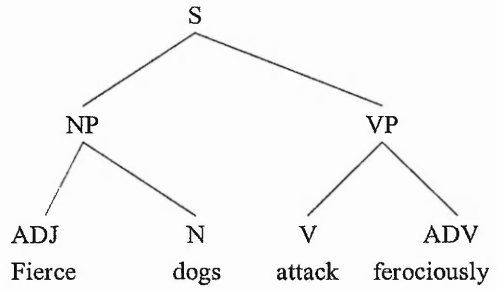

Figure 1. A simple parse tree

Grammars utilising production rules are known as *generative* grammars because the rules show how to generate all the legal (grammatical) sentences of the language. However, in many NLP applications the task is not to generate sentences, because they are already there. The task is to check that the presented sentences are indeed legal, i.e. to parse them. This involves finding a parse tree for the sentence allowed by the set of productions. One of the most successful algorithms for doing this is embodied in the *chart parser*, a parser which finds a path to any correct parse (and there may be several for a single sentence) in an efficient manner which re-uses partial parses made along paths ultimately found to lead nowhere (e.g. Winograd (1983)). It is not necessary for the purposes of this discussion to detail this further; interested readers may consult Allen (1995) for a full description of the general principles of the chart parser.

However, the problem for NLP is that it is very difficult to construct a complete set of production rules. Sets of production rules do allow infinite grammars (grammars in which the number of possible sentences is infinite) via recursion, but in practice it is difficult to write down all the rules needed to parse real texts. This problem grows more acute as the number of non-terminal symbols (N, NP, V, VP etc) increases. The example given above defines a very small (finite) grammar based on only a handful of tokens. For a real natural language such as English, any meaningful parse will require much finer subdivisions (e.g. to distinguish between plural nouns and singular nouns, or common nouns and proper nouns, or count nouns and mass nouns etc) and large numbers of production rules (usually thousands) utilising these symbols. Clearly, the number of rules is dependent upon the number of different symbols employed, because if a category such as 'noun' is further sub-divided into 'count noun' and 'mass noun', the number of rules will approximately double. In addition, the lexicon which holds the terminal symbols must categorise them using the enlarged symbol set, so that many terminals will have multiple senses. There is therefore a combinatorial explosion in the number of possible parses of a sentence which have to be tested. Due to the need for a very long list of rules, it is not uncommon for parsers to be unable to construct valid parses for a large percentage of the input sentences; many sentences are unparsable not because they are ungrammatical, but simply because the relevant production rules and/or non-terminal symbols have not been provided. Furthermore, many parses fail due to an inadequate lexicon, neologisms (recently coined words) being a particular problem.

The above describes failures to provide any parses for a sentence, but the opposite problem often occurs: too many parses for those sentences which do get parsed. Even where the parser succeeds in creating parses for a sentence, the output may not in practice be useful, due to very large numbers of possible parses (e.g. hundreds) caused by multiple word senses multiplying up. This is an inevitable consequence where a language containing a high percentage of polysemous words (in the same part of speech category) is processed by a purely syntactic parser as described above. For example, in the sentence *We went to the bank to get some cash* a purely syntactic parser would provide two parses identical save for the different senses of the word *bank* (financial institution, side of a river). Furthermore, in addition to the problem of multiple word senses, some whole sentences may be inherently structurally ambiguous, and this also multiplies up the total number of possible parses. Sentences such as *Put the block on the box on the table* illustrate this (see Church and Patil (1982) for a discussion on this problem.) In both such types of ambiguity, extra semantic processing is required in order to obtain the single correct parse, possibly including knowledge of the discourse preceding the sentence being parsed, or even of some extralinguistic situation described by the text.

Grammar Type	Name
0	Unrestricted Phrase Structure
1	Context Sensitive
2	Context Free
3	Regular (Finite State)

Table 1. The Chomsky Hierarchy

Grammatical formalisms have been categorised by linguists and computer theorists. The Chomsky Hierarchy categorises grammars as being of four types, as listed in Table 1. In the Chomsky hierarchy, natural languages correspond to Type 0 grammars. The context free grammar (CFG) is perhaps the grammar type which has most often been used to underpin parsers. Clearly, then, there is also a theoretical mismatch between the abilities of most parsers and real human language. This was recognised as long ago as 1979, when Gross launched a swingeing attack on the generative approach (Gross (1979)). Gross described the failure of an attempt to construct a generative grammar of French with a coverage comparable to traditional grammars, and as a result questioned the validity of the whole generative grammar approach.

The KEP system as described in Chapter 4 does not in fact employ full parsing of the kind described above, and for this reason this topic will not be further detailed. However, deep processing systems do usually parse text and so readers interested in delving further into this topic should consult introductory texts such as Grishman (1986) (which introduces the Chomsky hierarchy from the viewpoint of the computational linguist) or Cohen (1986) (which introduces the Chomsky hierarchy from the viewpoint of computer science).

1.2.5 The Need for Integration

The problems described above cannot be tackled in isolation from each other if full text understanding is to be achieved. Communication of information must be employed between all levels (lexical, syntactic, semantic, pragmatic) to ensure correct understanding. For example, in the sentence *I went to the bank to get some cash* the word *bank* is used. In isolation, this word has at least two possible meanings (financial institution or the side of a river) and so at the lexical level the sentence is ambiguous. In fact, the sentence is still ambiguous at the semantic level, since it is conceivable that a large pile of money has been stored on a river bank for some reason. To disambiguate the sentence, pragmatic-level knowledge is required (financial institutions usually have cash, but river banks rarely do). This knowledge is communicated to the earlier levels so that the correct sense of *bank* is selected in the parse. Note also that it may be that at an even higher level, that of the discourse, information may need to be extracted. If the sentence was part of a text describing a shipwreck in which banknotes were spread out on the river bank to dry off, then clearly this would override the usual pragmatic interpretation.

Thus a strict linear progression of processing from the lexical through to the discourse level will not work for successful NL understanding. Sparck Jones (1983) has illustrated this point in a discussion on the problems of parsing compound nouns, such as *park border plants*, where there are issues such as bracketing (is it *park (border plants)* or *(park border) plants* ?), lexical disambiguation (what sense of *border* ?) and meaning characterisation (is a *border plant* one that is actually **in** a border, or one that is **for** a border?). After an examination of the processing and information required at each level, Sparck Jones concludes that "there are problems about the conventional natural language program in which the contents of clearly demarcated information boxes labelled syntax, semantics, and pragmatics are applied in successive processing steps". In fact, Sparck Jones goes further and suggests that even a staged processor which passes on all possibilities from one stage to the next may not work on practical grounds, because "in the limit *(it may be)* so inconvenient as to undermine the idea of effective processing on which the staged processing is based" (italicised text is the author's addition).

The need for communication of information between levels is greater for some applications than others, for example in systems where there are actually more levels. Thus in MT and in handwriting recognition the backwards communication must be extended even further backwards to allow word boundaries to be detected correctly and individual words to be correctly identified. For example, Rose and Evett (1992) have shown how semantic knowledge may be used in handwritten word recognition. However, the KE practitioner is at least fortunate in that his starting point (see next section) is often the end result of another discipline's work, such as that of handwriting recognition, where the desired output would be a machine-readable text.

Sparck Jones (1983) describes a system in which all syntactic processing is finished before semantic processing starts, and all semantic processing is finished before pragmatic processing starts. An alternative method would be to allow syntactic processing to be paused until information had been passed back to it from the semantic and higher levels, and so on. Allowing all information from all levels to be available at any level is in effect simultaneous processing of all levels. But even this will present problems for the coder, since in any sequential program it will be difficult to decide upon the correct order of processing. The conclusion arrived at by Sparck Jones is that "concentrating on sentence parsing in its own right is of limited utility".

1.3 The Starting Point for Textual Knowledge Extraction

For the KE task discussed here, the input resource is text in machine readable form. Secondly, by choice this text is explanatory in nature, and may be regarded as the written equivalent of a single speaker lecturing upon some topic. Thus certain difficulties inherent in speech recognition and dialogue do not arise. The input text is also assumed to be free from spelling mistakes and grammatically correct. In addition it is assumed to be coherent; in other words, it is assumed to be a *text*, and not merely a jumble of unconnected sentences (see Halliday and Hasan (1976) for an extended discussion on what makes a text a text). Despite this advanced starting point, the issues discussed above must be resolved if *deep* text KE is to be performed. A lexicon will be required. The difficulties of providing reliable parses for all the sentences as described above exist. For full text understanding, WK in its widest sense is required. Also, anaphoric devices within text must be detected and resolved. Even with all these items in place, decisions will have to be made regarding *what* to extract.

Note that the term *full understanding* has been used above; as has already been suggested, it may well be that a deep understanding of the text is not in fact necessary for the intended application of the output knowledge. A *shallow processing* approach might prove feasible, in which some or all of the processes described in the preceding sections may be avoided or omitted. These choices will be discussed in the following chapter. It is safe to say that all current successful IE/KE systems rely upon aspects peculiar to their applications in order to reduce the amount of processing required. KE from text is hard. It is not easy to say just how difficult it is, but Ristad (1993) has argued that the NLP task in its deepest sense is *NP-complete* (nondeterministic polynomial complete)[2] i.e. is not solvable in a time which grows only polynomially with the problem size (but which almost certainly requires a time which grows exponentially with problem size – see Aho, Hopcroft and Ullman (1974)). Therefore any ways of making the problem easier are welcome.

[2] Ristad's argument is vague, and it is not clear exactly what he meant by this assertion; it is repeated here merely as supporting evidence for the hypothesis that doing deep NLP is difficult. However, the author of this thesis found Ristad's book to be unsatisfying.

1.4 Motivation for Reported Work

1.4.1 The Intelligent Recognition Systems Group

The Intelligent Recognition Systems group (IRSG) is a research group within the Department of Computing at Nottingham Trent University. Members of this group are interested in NLP systems capable of recognising and processing text at various levels. These range from handwriting recognition to fact recognition (the subject of this research). In addition the group contains researchers interested in computer aided learning (CAL) from the viewpoints of both the students and the teachers. Thus a wide range of pedagogical applications are also investigated. The research reported upon here was initially motivated by the desire to automate certain aspects of examination marking, but recent developments have increased the interest in providing knowledge for tutoring systems, and in extracting knowledge for automatic creation of glossaries.

1.4.2 Motivation for doing Knowledge Extraction

It is useful to ask the question: why is KE worth doing? It has already been argued that the KE task is both challenging and exciting (section 1.1.1), and some practical benefits arising from successful KE have been suggested. However, in addition to purely practical benefits, KE systems are also of theoretical interest to linguists and epistemologists, who are interested in the ways in which knowledge is expressed in language and the nature of that knowledge respectively. Therefore the interest in KE systems is not restricted to potential applications.

KE systems are also of interest to the NLP practitioner, because successful knowledge extraction from text involves most of the major difficulties of doing NLP. KE thus forms an ideal sub-domain of NLP in which various techniques and approaches can be tried out. It also has the advantage of being a domain in which results can be compared with human performances in a relatively simple manner, so that the degree of success of KE systems may be measured. In Chapter 5 the performance of the KEP program (developed to test a novel pattern-matching approach) is evaluated in this manner.

In the following paragraphs, some of the applications for KE mentioned earlier are considered in more detail. One of these, automatic glossary construction, is the KE task chosen for the research reported in this thesis. The motivation for choosing this particular application is discussed in section 1.4.5.

1.4.3 Automatic Marking Systems

There have been several attempts to automate the marking of student assignments (such as essays and collections of short paragraphs of text or single-sentence answers). For example, Lou and Foxley (1994) have developed the STAMS system, which aims to assess the semantic content of students' NL responses by comparing teacher and student answers in a fuzzy manner based upon online thesaural

entries. This is an area of increasing interest, particularly in the UK following the recent expansion of the higher education sector, which has resulted in a tendency towards higher student-staff ratios and hence increasing academic staff workloads.

One prototype examination marking system is that of Allott, Fazackerley and Halstead (1994), which uses an activation-passing network to mark single-sentence examination/test answers as correct or incorrect. The approach taken ignores syntactical information in the answers, which at first may sound surprising, but which due to the highly-constrained nature of the subject domain (Computer Science, CS), turns out to be feasible. The method is to look for combinations of key words to trigger nodes in a hierarchical structure called an *activation passing network* (APN). A node at a given point in the network is activated if the sum of its inputs exceeds a preset activation potential, whereupon it produces output(s) which are input(s) to higher-level nodes. For example, an *evidential* node might be triggered by the presence of a particular word or one of its synonyms. The correctness of the whole student response is determined by whether the single top-level node is activated or not. The APN used to mark student responses to a given question acts as the knowledge base for that question; each question has its own distinct APN. These APNs are presently designed and input by the human examiner. The question therefore arises as to how far the production of APNs could be automated, using KE techniques.

Although the research direction taken by the author does not directly answer the above question, it may ultimately aid in the automatic marking process. For example, where students are asked to give examples of concepts, the list of "correct" examples may be collected automatically from text using a KE program. This list might then be used to construct the APN used to mark student responses.

1.4.4 Automatic Teaching and Learning Systems

CAL systems have a long and distinguished history, extending back almost as far as the computer itself. However it is only recently that the very large storage capacities required for effective multimedia teaching systems have become a reality. In addition, research into the ways in which people learn and how these apply to computerised systems is now mature. Thus the stage is at last set for really effective teaching systems which are technically feasible, affordable, effective, and above all user-friendly.

Developed within the IRSG, the HypeLab/HyperTutor system (Edwards, Powell and Palmer-Brown (1995)) utilises a hypertext knowledge base and a windowed front end with a semi-NL interface (Long, Powell and Palmer-Brown (1995)) to provide a complete CAL system for any subject domain. This product is discussed further in section 6.3.5, but the motivating factor here is the use of a semantic net as a KB within HypeLab/HyperTutor. Within this semantic net, nodes hold concepts, which are joined by various link types indicating specific conceptual relations, such as the partition relation (*has part* link

type). Since KEP aims to extract this sort of conceptual knowledge, the possibility of automatically populating HyperTutor's KB arises. This subject is further discussed in a Chapter 6.

1.4.5 Automatic Glossary Creation

The word processor (WP) is today an essential part of office life. Leading WP packages such as Microsoft Word and Corel WordPerfect have attempted to provide a comprehensive list of features so that the user may produce documents in a variety of styles with a minimum of effort. Unfortunately, the plethora of features available may result in incomplete understanding of the product by the user. The WP manufacturers have recognised this problem and traditionally have provided 'help' functions within the software. More recently, intelligent software agents have been enlisted in order to second-guess the intentions of the user who finds himself in difficulty, and intervene when appropriate.

In addition, many WP features have been automated to a certain degree; for example, spelling may be checked *continuously* in the latest versions of the leading packages. Thus the user need not perform a distinct spell-check operation at the end of typing. Other features are less well advanced. For example, index creation relies upon the user indicating in some way which phrases are to be placed in the index (which is then created automatically, involving the page numbers on which the selected phrases appear). However, the user is still required to mark phrases for index inclusion, a time-consuming task. Similar comments apply for glossary and bibliography creation. It would be much easier for the user if there were a button/icon for completely automatic index/glossary creation. Clicking this button would create an entire index or glossary for the current document, without the need for the user to specify which phrases were to be included. Clearly this goal is not achievable unless the WP software is capable of producing a list of index terms or glossary terms without human intervention.

Automatic glossary creation was chosen for the subject of the research reported here. As mentioned earlier, it is a challenging area offering great practical rewards whilst allowing for the possibility of interesting insights into how one should do NLP. It is a relatively self-contained task that might be achieved through either deep or shallow methods. Attempting to do it in a *shallow* way might reveal important results in a field which in many ways is a microcosm of NLP in general. For example, it might allow one to answer the question *Is full text understanding necessary*? Questions might also be answered concerning how easy it is to make non domain specific systems.

Automatic glossary creation is not a trivial task. It is sufficiently difficult to present a real challenge in a way which, say, automatic index creation does not. Both glossary making and index compilation require the collection of technical or specialist terms from the text, but a glossary has the extra dimension of requiring explanations and elucidations of such terms. Intuitively, this is the more difficult part. These elucidations were introduced earlier, and include such things as *definitions* and *hypernyms*. Identifying a

concept in a text and a definition of it sounds like a job for a deep system. The challenge of this research was to do it using a shallow system. Furthermore, this challenge was amplified by the attempt to create a domain-independent system. As will be described in Chapter 4, the challenge was met using a shallow pattern-matching approach which uses no external knowledge resources. The reasons for choosing a pattern-matching approach are developed in that chapter.

Automatic acquisition of *technical terms* (TT) is an established field (usually called *terminology extraction*) which has been driven in part by the desire to automatically extract index terms. A brief survey based on some recent papers is given later (in Chapter 5). The KEP program contains a TT acquisition function based upon the scheme of Justeson and Katz (1995), modified to use part-of-speech tagging information, and with extra design features intended to detect single-word terms. (This is described in detail in Chapter 4). The TT acquisition function in KEP is combined with a novel acronym extractor, with KEP's pattern-matching conceptual relation extractors, and with linking, cross-referencing and ordering code, to produce a glossary output. The process of creating the glossary is indeed completely automatic, but, as will be seen later, the glossary so produced requires manual post-editing. However, even though post-editing is required in order to produce a complete and error-free glossary, the effort required is still lower than that needed for systems requiring manual TT identification.

1.5 NLP and Linguistics

Natural Language Processing is a branch of AI, and that means computers and computer programs. However, the discipline of linguistics has been around for many centuries in one form or another, and obviously predates the computer. It is worth making some comments regarding the relationship between NLP and linguistics, not least because the empirical discipline of corpus linguistics is of direct relevance to the reported research.

1.5.1 Traditional Linguistics

The discipline of linguistics has a long and distinguished past, stretching back to the ancient Greek philosophers, such as Plato (who in his *Theory of Forms* was interested in the connection between words and the concepts they represent). However, it is not the intention of this section to relate a history of this subject, for it is modern linguistics which is of relevance here. Thus we shall skip over almost the entire history of this subject and emerge into the mid twentieth century. The most important event to be aware of for the purposes of the following discussions is the publication in 1957 of Noam Chomsky's book *Syntactic Structures* (Chomsky (1957)), which marked a watershed in the relationship between linguistics and other disciplines such as psychology and philosophy. Chomsky's ideas have been constantly developing since this time, but his influence has resulted in a new flowering of linguistic studies.

The central idea in Chomsky's work has been that of *innateness*, the ability of humans to learn certain types of language. The idea that humans learn only a small subset of all the imaginable language types is a powerful one. The suggestion is that we do in fact all speak the same basic language, with grammatical differences merely being switch settings set at an early age, and of course with vocabulary differences. For example, one switch might be the function-order switch which determines the basic pattern of subject, object and verb in a sentence (in English this is set to SVO, but for example in Welsh it is VSO). This suggestion leads to the ideas of *deep structures* and *surface structures*, in which the latter are alternative ways of expressing the former. Surface structures correspond to different parse trees for utterances with the same meanings, such as the *transformation* from active to passive voice. It is impossible to encompass this huge subject here, and interested readers should consult one of the many introductory books on modern linguistics (e.g. Smith and Wilson (1990), Pinker (1994)).

Within modern linguistics various sub-fields have arisen, all of them influenced to a greater or lesser degree by the ideas of the Chomskyan revolution. These sub-disciplines include the study of language acquisition, language variation (geographical), language change (historical), semantics, pragmatics, and sociolinguistics. Linguists even study traditional schoolbook grammar. It is the latter which provides the first link to computer science and NLP. The early computer programming languages required formal definitions of allowed syntax, and hence methods of expressing grammars. Notations such as Backus-Naur Form (BNF) were developed since these could represent the transition networks capable of describing the simple syntaxes of the early programming languages. Naturally, the thoughts of some individuals turned to the use of transition networks, especially augmented transition networks (ATN), for the representation of *natural* languages. ATN approaches have largely fallen out of favour by the NLP community today, but in the early days they showed some promise and provided limited practical success (see e.g. Noble (1988) for an ATN-based NLP study). As the theoretical and practical limitations of ATN approaches became apparent, those computer scientists interested in processing human language moved on to other formalisms, and the discipline of NLP became truly established. It would be fair to say then that the field of NLP arose mostly through the practical efforts of early computer scientists, rather than through interest from traditional linguists.

It is only recently that linguists have taken to the computer *en masse*. The reasons include purely practical factors to do with the processing power and storage capacity of modern computer systems. In addition, a certain change in attitude towards experimentation has arisen in the linguistic community. This alteration of viewpoint is better discussed in the section which follows this one.

Finally, it should be mentioned that the term *computational linguistics* is often encountered. To a certain extent this overlaps NLP. However, the computational linguist has historically tended to concentrate on the building of parsers (see e.g. Sparck Jones and Wilks (1985)), and to a lesser extent on semantic

analysers. The typical computational linguist hails from the linguists' camp rather than from the AI field. Grishman (1986) gives an excellent idea of the flavour of this subject.

1.5.2 Corpus Linguistics

It is no exaggeration to say that the field of corpus linguistics has largely been made feasible by the computer. Empirical studies involving large bodies of text are only possible if they can be achieved within reasonable timescales and budgets. Prior to the invention of the computer, all such studies had to be performed manually, from paper resources. Naturally, this prevented many experiments from being carried out; in fact, it prevented them from even being conceived.

The history of corpus linguistics has been outlined in McEnery and Wilson (1996). It is one of initial enthusiasm followed by a period of disfavour (on theoretical grounds), followed by a new period of interest (the period we find ourselves in today). In some sense, then, this history echoes that of AI itself. However, the period of disinterest within corpus linguistics arose not through lack of success or stagnation, but because of theoretical objections raised notably by Chomsky. The thrust of Chomsky's argument was that no corpus could ever be representative of a particular language because of the infinite nature of language. On the other hand, introspection by a linguist could indeed generate all aspects of a language, and so this was a preferable route towards linguistic truth. In linguistic terminology, Chomsky argued that linguists should examine *competence* rather than *performance*.

It is now recognised that although no corpus could be representative of a language as a whole, it does indeed contain examples of a language as actually used, and so may implicitly contain valid and interesting linguistic data. This realisation has grown with the increasing practicality of actually doing the envisaged experiments. It is now easy to obtain large corpora, and relatively cheap in computing terms. For example, the British National Corpus (BNC), a recently released corpus of about one hundred million words of general English (spoken and written), is available on only three compact disks and comes complete with sophisticated accessing and processing software (Burnard (1995)). The BNC is fully part-of-speech tagged, an extremely useful property which will be referred to in some detail later (see Chapter 4 for details of the tagger, CLAWS4). Other easily available corpora include the Lancaster Oslo/Bergen corpus (LOB) (Johansson et al. (1986)), and the Longman-Lancaster corpus (Summers (1991)).

Although new technology has recently fostered corpus linguistics, linguists were in any case moving away from Chomsky's position of competence over performance. The availability of corpora merely accelerated this process. For example, Sampson (1987) argues that there is good corpus-derived evidence against the grammatical/ungrammatical distinction for individual sentences, to such an extent that "the enterprise of formulating watertight generative grammars appears doomed to failure". The evidence was

gathered from about 40,000 words of the parsed LOB corpus, whose manual parsing preceded any thoughts of carrying out such a study (and thus could not be accused of being biased towards or against the grammatical/ungrammatical evidence thesis). Noun phrases not occurring in coordinate structures were selected, giving 8328 instances. These were each categorised as a pattern of constituents from a 47-member set of constituents, e.g. **DT* *S , F** meaning *determiner + plural noun + comma + finite clause.* This categorisation revealed that the 8328 noun phrases fell into 747 different categories (patterns), but that one pattern (*determiner + singular noun*) accounted for 1135 (about 14%) of the 8328, and that there were 468 patterns represented by *only one* NP instance. Sampson makes the point that it is extremely difficult to determine the boundary between "grammatical" (generatable) and "ungrammatical" (not able to be generated) phrases if a high proportion of the grammatical phrases are very rare. The implication is that if the latter were true, then it would be *practically* impossible to build a rule-based generator of only grammatical phrases. Sampson goes on to support the thesis that it is indeed true that a high proportion of grammatical phrases are very rare, by plotting a graph of the logarithm of constituent-type frequency expressed as a proportion of frequency of commonest constituent-type (x-axis) against the logarithm of the proportion of constituent-tokens in samples belonging to types of frequency <= x (y-axis). This turns out to be a straight line approximating to $y = 0.4x$, and shows no sign of increasing in gradient at the right-hand end. Sampson argues that such an increase would be expected to occur in an abrupt manner if there were a distinct grammatical/ungrammatical boundary. Sampson also discusses possible extrapolations of the graph for larger text samples and argues that grammatical constructions can indeed be extremely rare statistically.

Sampson's arguments have not been left unchallenged, however. Taylor, Grover and Briscoe (1989) have disputed the hypothesis that there are many singletons which a generative grammar cannot handle. Using the ANLT (Alvey Natural Language Tools) they re-process Sampson's data, and come to the conclusion that there simply are not as many singleton types as he claimed, and what is more, the singletons that do occur are not odd in any way. (Where ANLT fails to parse them, it is usually due to an obvious oversight in the design of the ANLT parser). They say that "Sampson's result is suggested by his *analysis* of this data, not the data itself". The author of this thesis also believes that the presence of a very large *hapax legomenon* does not indicate that there is no clear grammatical / ungrammatical divide. Clearly a human is capable through introspection of stating whether a sentence is grammatical or not, or is in some way "odd", even if he cannot state exactly why this feeling arises. In the vast majority of cases he will be able to make a boolean decision. The test text used by Sampson contained (by definition) only grammatical sentences; how then could it say anything about *un*grammatical sentences, or about the grammatical / ungrammatical divide? It is probable that the Sampson argument says more about the *practical* difficulties of creating a good parser than about the theoretical existence or otherwise of a sharp grammatical / ungrammatical boundary.

The Sampson paper is an example of the use of corpora to assist in the competence/performance discussion, and shows how performance data can give rise to practical NLP implications. It is important to note that corpora are being used today by linguists and NLP practitioners largely to provide *linguistic* information, that is to say information regarding lexis, syntax, semantics etc. Even the most recent attempts to use corpora for *automatic learning systems* (see review articles Ng and Zelle (1997), Cardie (1997)) derive essentially linguistic knowledge, even if that is at the semantic level. But there is another sort of information that corpora contain - the actual *factual* information, i.e. the knowledge, that the texts contain. This latter aspect has not been exploited to any great degree yet, largely because corpora are not usually created with such an aspect in mind (unlike online encyclopaedias etc). Furthermore, until recently there have simply not been enough corpora to support such applications. This situation is now changing. Corpora exist in many specific domains, such as: computer science undergraduate textbooks (HKUST corpus, available from the Language Centre, Hong Kong University of Science and Technology), telecommunications (ITU corpus, available from the Department of Linguistics and Modern English Language, Lancaster University), agricultural research theses (Reading Academic Text corpus, RAT), (Carne, Furneaux and White (1996)), and contract law (Aarhus corpus, available from the Aarhus School of Business, Aarhus, Denmark).

The development of the KEP program described in this thesis utilised corpora in both of the aspects described above. Linguistic knowledge aids in the extraction of factual knowledge, and corpora themselves contain texts holding factual knowledge. About 75% of the 3209 written texts in version 1.0 of the BNC are classed as 'informative', these texts being classified into eight categories covering topics as diverse as Arts, Commerce and Pure and Applied Science. Many of these 'informative' texts contain or comprise excerpts from textbooks and explanatory texts, so that they are likely to contain definitions, examples, explanations etc. Since KEP is designed to be non domain specific, this variety of explanatory and introductory texts from diverse subject areas acts as a good test of KEP's NDS claimed credentials. Therefore the BNC was chosen to provide both training and evaluation texts for the development of KEP. In addition, the CLAWS tagger used to provide the part of speech tags for the BNC texts is available as a separate product, so that any text may be tagged in a similar manner to that of BNC texts. Thus the choice of the BNC/CLAWS combination has allowed KEP to access pre-prepared texts as well as any other text which is available in machine-readable form (and hence which may be tagged in a manner identical to that of BNC texts prior to KEP processing).

1.6 Chapter Summary

In this chapter the nature of Knowledge Extraction from text (KE) and how it relates to the wider field of natural language processing (NLP) have been discussed. The motivation for doing KE has been examined, and the choice of *automatic glossary creation* as the subject of the research has been justified. Difficulties inherent in *deep* NLP and by implication deep KE have been considered. These problems

include that of incomplete message content, the need for large knowledge bases, the difficulty of creating good parsers, and the need to integrate techniques at all levels (lexical, syntactic, semantic, pragmatic, and discourse).

The Chomskyan debate regarding competence vs. performance has been introduced, and the rise of modern corpus linguistics has been discussed within this framework. In the course of these discussions questions have been raised regarding the degree of *practical* effectiveness of traditional syntactic parsers, and indeed regarding the *theoretical* basis upon which they are founded (i.e. the notion of there being a definite boundary between grammatical and non-grammatical sentence types). The use of corpora to aid in KE has been suggested, and the important idea put forward that full text understanding i.e. deep processing may not actually be necessary for successful KE, so that some or all of the obstacles described above may be avoided.

2. Some Existing Extraction Approaches

2.1 Introduction

In the previous chapter the field of knowledge extraction (KE) was introduced and its place within NLP research examined. In this section a representative sample of real message understanding (MU), information extraction (IE) and KE systems is described. The major developments and approaches are considered. The descriptions given will be couched in terms of a categorisation system described in the following section. By introducing existing KE/MU/IE systems the aim is to provide a snapshot of the current state of the art, together with an assessment of the degree of success of such systems.

2.2 A Two-dimensional Categorisation of Extraction Systems

When considering a given MU or KE system it is useful to be able to place it into a framework which identifies its general approach. A two-axis framework can be provided using the orthogonal aspects of *domain specificity* and *processing depth.*

2.2.1 Domain Specificity

MU and KE systems may be *domain specific* or *non domain specific.* The word "domain" is often used by NLP practitioners in the sense of subject or topic, but within the IE/KE arena it also carries an additional meaning relating to the type of medium or communication channel used. Media/channels include newspapers, telexes, textbooks, posters, reports, papers, essays etc. Where a channel such as *telex* is specified, what is meant by this is not the fact that a particular electronic method of delivery was used, but that the *genre* of the text is that which results from using that channel, either by technological necessity or by convention. The name of a specific domain usually contains within it an indication of the medium/channel/genre in addition to its subject area.

An example of a specific domain is *banking telexes.* The subject area is that of banking, with all that entails. There will be specialist or technical terms such as *credit, debit, account, deposit* etc. Thus a domain specific approach entails a specialist vocabulary. This specialist vocabulary will be closed, i.e. finite and small in comparison to the full English lexicon. However, there is evidence that in such situations individual words may be used in ways representing a greater number of parts of speech than in general usage (McEnery and Wilson (1996)). Thus sublanguages do not necessarily imply that we can simply throw away half of the lexicon.

Despite this, the finiteness of the domain specific lexicon is a boon for practical computer systems, since a small word list means faster accessing time and faster processing where more than one word sense is possible, and where the correct sense must be selected (disambiguation). Fast speed of access is not just a

desirable (but relatively unimportant) feature of practical NLP systems - it can make the difference between a system that works and one that does not. A parser that takes twenty hours to process a small paragraph of text (and this is not an unusual figure) may not in practice be regarded as a program that works, even if it does eventually come up with a solution. (In many cases, there may not be a successful parse even after this time.) But a program that does the job in twenty seconds would almost always be regarded as a working program. To illustrate this point, consider that Keenan (1993) found that the Alvey Natural Language Tools (ANLT) parser (Briscoe (1988)) produced parses in only about 70% of cases, when run on a test set of sentences taken from the Longman Lancaster corpus. This low rate was caused by a combination of inherent inability to parse particular sentences, as well as hardware limitations. System crashes accounted for about 33% of the failures. Furthermore, Keenan reported single-sentence processing times of up to nine hours for a SUN 3/160 machine, during which time other users were refused logins due to insufficient swap space. In addition, where sentences were successfully parsed, there were often very large numbers (hundreds) of possible parses, so that the output was effectively useless. It is not surprising that Keenan rejected the use of this parser for his chosen application (syntactic analysis in handwriting recognition).

Note that the *banking telexes* domain concerns only *telex* messages. Such restricted communication channels often utilise a sublanguage. That is to say, the grammar and vocabulary used for telexes is constrained not only by the subject of the message but also by the fact that it is a *telex* message. Constraints may arise through convention, where a certain style of writing arises within a specific user community, or through technological necessity, as was the case with the old-style GPO telegrams, which were printed in capital letters only and utilised the word STOP to indicate a full stop. The study of textual variety is a topic within linguistics which has received increasing interest along with the current resurgence of computer corpora and their use in empirical linguistics (see e.g. Rademann (1998)).

As hinted at above, it is reasonable to suppose that the restricted nature of specific domains will allow various short cuts to be taken on the MU/KE path, and indeed this is usually the case. Examples of domain specific (DS) systems are described later in this chapter. Hahn (1989) argues that in order to mimimise processing effort, systems *should* be domain specific - "adapted to the domain involved". However, the paper discusses semantic parsers, which require large amounts of domain knowledge in order to avoid a separate syntactic processing stage. We shall see later from examples of existing NLP systems that it is indeed true that being DS reduces processing effort, as well as *creation* effort (the effort needed to create the NLP system in the first place) since the whole of human knowledge does not need to be made available to the program.

Non domain specific systems (NDS), also called *domain independent* systems, are those which are not restricted to specific subject areas. However, it is possible to conceive of a non domain specific system which looks only at telexes, say (but on any topic). Thus this definition is somewhat fuzzy because of the

possibility of specifying the communication channel(s) allowed. An example of a NDS program would be one which holds light conversations on any topic with a human user at a computer terminal (keyboard and screen). It is immediately obvious that NDS systems must be more challenging than DS systems for the NLP developer. Examples of NDS systems are given later in this chapter.

Although for the reasons discussed above most KE/IE systems are DS, there is also a practical disadvantage to the DS approach. This occurs when the system is ported to a new domain. The new domain may require not only the creation of a new DS knowledge base, but it may also require changes to syntactic information held in pattern files or even in the system code itself. The latter can occur when the system has been optimised for a particular domain, this optimisation extending to the program code rather than being confined to external data files. Domain-dependent code should be avoided even in DS systems if there is any chance that, in the future, further domains are to be added. However, even for systems which have been sensible enough to confine DS aspects to external files, there is usually a large amount of work to be done to add a new domain. This problem has been discussed in the review paper by Cardie (1997). The solutions to this problem currently being investigated include the provision of learning mechanisms for automatic acquisition of patterns. These may be completely automatic, or may require feedback from a human trainer. Completely automatic pattern detection code is the most desirable, but even where this is not possible the use of an *interactive training program* can reduce porting times by large factors. For example, Cardie (1997) cites the example of the AUTOSLOG system (Riloff (1996), Lehnert et al. (1992)) which was able to reduce average port times from 1200 hours to five hours using a semi-automatic extraction pattern finder. This is indeed a significant reduction in porting time.

The KEP program described in Chapter 4 also contains a training function for human-assisted extraction pattern finding, and the use of this mode is described in Chapter 5. Although as a NDS system KEP cannot suffer from the porting problem, the training mode assists greatly in the construction of token and pattern files.

2.2.2 Processing Depth

The second dimension along which one may categorise an NLP system is that of processing depth. Processing may be *deep* or *shallow*. These terms are generally used as follows: deep processing aims for full text understanding, and utilises parsers, KBs etc to the full; shallow processing means the achievement of NLP goals without necessarily utilising all of the above tools, due to the availability of "short cuts" for a particular application. The above represents the author's own definitions of deep vs. shallow processing. The terms tend to be used within the IR/MUC community without any formal definition, although Hahn (1989) touches on the subject, and Jacobs (1990) discusses the subject in relation to methods of skimming text for 'interesting' sections.

Hahn (University of Freiburg) is one of the few researchers worldwide attempting KE from text. In Hahn (1989) he discusses the idea that (even in a domain specific program, looking at a text from that domain) the depth of processing should vary with the perceived degree of relevance of each section of the text - "relevant parts of a text are analysed in-depth, while irrelevant material is processed in less detail, or simply skipped". Naturally, such an approach presupposes heuristics for identifying "relevant" sections. In KEP terms this is called *triggering*, and "relevant sections" becomes "relevant sentences". This is described fully later. Jacobs calls the search for relevant sections *text skimming*. In Jacobs (1990) the text skimming approach taken in the SCISOR program (Rau and Jacobs (1988)) is discussed (SCISOR is considered in some detail later). Depth of processing again varies with section relevance, and ranges from skipping a sentence altogether, through limited parsing to determine roles of actors, to full semantic parsing. Although the depth of processing varies from section to section of a text in systems such as SCISOR, these systems are best classified as deep processing systems, because they are able to do deep processing when necessary. On the other hand KEP is not designed to do deep processing and so is labelled as a shallow system.

It is usually the case that NDS systems take a deep processing approach, and DS systems take shallower approaches. This situation has arisen mostly through necessity. Considering the MT application, a conversational translator will undoubtedly require deep processing techniques if it is not to distort the speakers' meanings or create complete mis-translations. However, there is nothing to prevent deep processing for a DS application, and this approach is sometimes used, not least because of the reduced lexicon which goes with DS systems (see comments above, concerning SCISOR). It is rare, however, to find a NDS system which relies upon shallow processing methods. The KEP system described in this thesis is one such system and is therefore unusual in this respect.

Note that the depth of processing metric is necessarily a continuum. It is usually possible to state whether a given program uses a deep or a shallow approach, based upon whether it has the ability to do deep processing when the need arises. In addition one can usually say that Program X uses deeper methods than Program Y etc. Despite this subjectivity, the deep/shallow dichotomy is a useful one.

One other factor should be mentioned within this section, although it is not directly connected with depth of processing. This is the issue of *robustness*. This term can be used to mean 'unbreakable' in the sense that a given program never crashes as a result of a particular (textual) input (robustness sense 1). However, it is sometimes taken to mean that some kind of output is always produced, even if that output contains less detail than would ideally be provided (robustness sense 2). In this sense, a robust program will produce shallow output under adverse conditions (hence there is a connection between depth of processing and robustness (sense 2)). For example, the TACITUS system described in Hobbs et al. (1992) degrades gracefully because the abductive inference mechanism it uses is "inherently robust". Stede (1992) has considered the property of robustness in some depth. The issue of robustness is

mentioned here since it is usually the case that deeper systems are less robust (sense 1) than shallow systems. This is predictable; deep systems usually contain more modules, perform more processing on the input text, and spend a longer time doing that processing. There is therefore more opportunity for them to fail, either through bugs in the code or through omissions in the design. Very large programs may also fail for hardware-related practical reasons, such as running out of disk space or memory.

Table 2 shows a grid categorising a selection of KE/MU systems according to domain specificity and depth of processing. These systems and others are discussed in the following sections, where their positions in the grid are justified.

	DS	NDS
Shallow	FASTUS 'wit' JASPER	KEP
Deep	MEDLEE ATRANS SCISOR	Conceptual Dependency Preference Semantics

Table 2. Some KE/MU systems categorised

2.3 Non Domain Specific Systems

2.3.1 Deep Processing NDS Systems

2.3.1.1 Conceptual Dependency

Conceptual Dependency (CD) is an approach to text processing which attempts to construct an understanding of the whole text. First suggested in the early 1970's, the CD suite of programs has been developed over the intervening years by several AI researchers, most notably Schank, Abelson, Rieger, and Riesbeck. The system is described in the book Schank and Abelson (1977).

CD parsing amounts to the creation of a CD representation of a whole sentence based on the CD representations of the individual words in the sentence. The first step is to identify the main verb and noun in the sentence. The main verb is categorised as stative, transitive, or intransitive, and this is used to extract the correct CD verb representation from a dictionary. CD representations of verbs form a smaller set of *primitives* than the set of verbs in English. This approach allows all the actions expressable by English verbs to be mapped to a smaller set of 'atomic' acts; the aim is that the NL involved is irrelevant, since the acts are valid for all human societies. For example, the **PTRANS** act is used for all verbs indicating **P**hysical **TRANS**fer of people/objects from place to place. Thus it encompasses *walk*, *run*, *crawl*, *roll*, *fly*, *go*, etc. Table 3 shows a typical list of CD primitives (taken from Schank and Abelson (1977)).

CD uses the acts (or ACTs, in CD terminology) together with PPs (picture producers, i.e. objects/people), AAs (action aiders i.e. modifiers of actions), and PAs (picture aiders, i.e. modifiers of PPs) in allowed combinations called *dependencies*. These correspond to semantic relationships between the concepts, and are expressable in diagrammatic form. An example of one such dependency is the relationship between a PP and an ACT, and another is the relationship between an ACT and the PP that is the object of that ACT. These are illustrated in Figure 2, together with a sentence that brings these two together. The table has three columns: the first is the dependency rule together with its graphical symbol (e.g. double-headed double arrow, single-headed single arrow, superscript **o** for object); the second shows an example of the dependency rule with the variables (PP, ACT etc) filled out; the third shows a fragment of English which would be parsed to the column-2 representation.

Primitive Act	**Used For**
PTRANS	Transfer of physical location of object (e.g. *go*)
ATRANS	Transfer of abstract relationship (e.g. *give*)
PROPEL	Application of physical force to object (e.g. *push*)
MOVE	Movement of a body part by its owner (e.g. *kick*)
GRASP	Grasping of an object by an actor (e.g. *clutch*)
INGEST	Ingestion of an object by an animal (e.g. *eat*)
EXPEL	Expulsion of something from the body of an animal (e.g. *cry*)
MTRANS	Transfer of mental information (e.g. *tell*)
SPEAK	Production of sounds (e.g. *say*)
ATTEND	Focussing of a sense organ towards a stimulus (e.g. *listen*)
MBUILD	Building new information out of old (e.g. *decide*)

Table 3. CD Primitive Acts

The superscript **o** represents the object relationship. Other superscripts are available, such as **p** for past tense (for example, placing a **p** over the <=> symbol in the first row of the table allows the English fragment *John pushed* to be represented), and the slash (/) allows negation (*John does not push*). There are several more complex basic dependencies which have not been mentioned, such as the relationship between a conceptualisation and another that caused it. Each has its corresponding pictorial arrangement.

Dependency	**Example of Use**	**Example English**
PP <=> ACT	John <=> PROPEL	John pushes
ACT <-- **o** PP	PROPEL <-- **o** cart	pushes the cart
PP <=> ACT <-- **o** PP	John <=>PROPEL <-- **o** cart	John pushes the cart

Figure 2. Conceptual dependencies and examples (from Schank and Abelson (1977))

Conceptual dependency allows the description of fairly simple actions and events to be pictorialised, but with a certain loss of granularity. For example, the nuances of meaning inherent in *John shoved the cart* compared with *John pushed the cart* are lost. The above example is an extremely basic one; even simple English sentences may take several pages of CD diagrams to represent, so the scheme does become unwieldy. However, the claimed advantages of the CD method are that fewer inference rules are needed for systems processing CD representations (compared with systems which process other representations, such as full natural language text), that many inferences are inherent in the representation itself (in the sense that it is only a matter of reading them off the CD diagram), and that where there are gaps in the CD representation they are obvious and hence easy to detect and resolve.

As was stated above, during conceptual parsing the main verb of a sentence is first identified. This is translated to the correct representation and the empty roles are then used to create the full representation for the sentence. (The initial categorisation into stative, transitive and intransitive type verbs assists in role filling, since for example one would not expect there to be a direct object role for an intransitive main verb.) This process involves checking the types of objects, so that certain ambiguous constructions can be investigated and the correct representation chosen. For example, only animate objects may perform certain roles (waste paper baskets do not *run*, for example). The parser may also consult its 'memory' (any CD representations created so far, from previous sentences), so that with a sentence such as *John went to the park with the girl* it can be decided whether the park had a girl in it, and John went there, or John and the girl went together to the park. (Prepositional phrase attachment is a topic of interest for English NLP practitioners, such as Jensen and Binot (1987) who used MRD-based methods, but the methods employed rarely utilise memory of situations, so CD is unusual in this respect.)

For textual units larger than sentences, CD representations of individual sentences may be combined into one single diagram. This is useful for applications such as story understanding. The process is not simple; it involves the use of WK to make links between sentences, i.e. to turn a collection of sentences into a *text*. One of the most useful mechanisms for doing this is the use of **scripts**. A script is a broad framework of a situation that often occurs in real life, such as going to a restaurant, catching a bus to work, going to the dentist for a check-up etc. The SAM program (Script Applier Mechanism, also described in Schank and Abelson (1977)) allows simple stories to be understood as a CD representation. It is not necessary to go into the details of this program here, but the important point to note is that very many scripts are needed if a program is to behave in a non domain specific way. Furthermore, scripts themselves are not the full answer - combinations of scripts, and scripts interrupted by other scripts occur in real life. Thus a successful script applier must be able to handle variations on a theme, and must be able to spot the start of a new script situation and the point of resumption of a temporarily suspended one. *Plans* and *goals* were introduced to handle such cases, being planning and motivation-capturing devices assigned to the human actors involved. It is not feasible to go into these topics here. However, the division between scripts and plans was never made very clear by the various authors of CD.

In the 1980's Schank reorganised CD/script theory in an attempt to overcome the limitations of fixed scripts. The idea of *memory organisation packets* (MOPs) (Schank (1982)) was introduced. MOPs comprise collections of *scenes*, such as the 'queuing at a checkout' scene, and represent a more flexible approach to scripting. Also introduced were TOPs (thematic organisation points), which were general statements of types of themes, such as 'mutual goal pursuit against outside opposition', as found in *Romeo and Juliet.*

It is clear that the CD representation method embodies deep processing. The method does indeed attempt to fully understand the input text, and it tackles all the hard problems of NLP to some degree. Meaning is, however, extracted at a coarser level of granularity than is actually encoded in the original English text. The methodology is capable of extracting information from texts such as stories, but this involves the use of several large and complex programs together with large MRDs and WK resources. This is not really surprising. Human beings take many years of constant exposure and learning to reach a point at which they can understand simple stories, based upon knowledge of physical law, human motivations and script-like scenarios. It would be surprising if a similar amount of learning were not required for a program attempting to perform the same task.

Is CD a NDS system? The general approach is certainly NDS, since the CD representation is designed to be able to capture models of general real life situations. It is also true that there do not appear to be barriers to the expansion of CD to new domains, e.g. by the addition of new scripts. Thus the structure of CD is indeed *inherently* NDS, since it boils down all actions into a set of universal primitives and it allows the addition of modularised extensions to its WK. This is not to say, however, that CD cannot be used in a DS way: see section 2.4.1.2, which describes the ATRANS system, for such an application.

If CD, a non domain specific system, can also be used in DS situations, then one might ask whether *all* NDS systems can be used in DS circumstances. The answer to this question is tied not just to specific KBs needed for the new domain, but also to the textual variety (genre, sub-language, communication channel) of the specific domain. Where there is an evident difference going from NDS to DS, e.g. from story understanding to telex message IE, it is almost always the case that much tailoring of the system is needed to do the DS extraction well. Thus to say that a NDS system can also "be used in a DS way" does not mean that it can be dropped into the new specific domain without any changes. As will become apparent from forthcoming descriptions of DS systems, these invariably use characteristics of their domains in order to do the job well. Thus when a NDS system is ported to a specific domain, those characteristics peculiar to the new domain need to be catered for if the best possible performance is to be achieved.

2.3.1.2 Preference Semantics

Preference Semantics - Wilks (1975) - is another system based upon semantic primitives, also developed in the early 1970's. It is thus a contemporary of CD. Whereas CD attempts to map several words (verbs) to one of several primitive acts, Preference Semantics (PS) maintains a dictionary of meanings for each word, expressed as *formulas* built out of primitive *elements*. The idea of PS is to understand English sentences by building up possible meanings for them based upon the meanings of the individual words. Then the *preferred* meaning is picked out. Groups of words (clauses, simple sentences) form *templates*. A paragraph of text forms a *semantic block*.

The elements used to build up the formulas include entities such as MAN (human being), THING (physical object), and FOLK (human groups). They also include actions such as CAUSE, type indicators such as HOW, and sorts such as GOOD. Cases are used in the manner of noun declensions i.e. TO (towards), SOUR (source i.e. from), SUBJ (i.e. nominative), OBJ (accusative), IN (containment), POSS (genitive) etc. In addition, classes of elements are used, such as *ANI for the class of animate elements (i.e. MAN, BEAST, FOLK) or *HUM for human elements (MAN and FOLK). These elements are combined to give the formula for a particular word sense. For example, for *policeman* the formula is:

"policeman" -> ((FOLK SOUR)((((NOTGOOD MAN)OBJE)PICK)(SUBJ MAN)))

which means "a person who selects bad persons out of the body of people". The *head* of this formula is the rightmost MAN element. A *bare template* is a pattern of three heads, such as MAN GIVE THING. A list of possible bare templates is held. To understand a piece of text, it is fragmented (in a manner discussed shortly) and each fragment assigned a list of bare templates that match it. For example, MAN GIVE THING matches *John gave Mary the book.*

The different template MAN GIVE MAN also matches *John gave Mary the book.* However, when the formulas attached to the heads in the template are combined and expanded, the *semantic density* of MAN GIVE THING is found to be greater than that of MAN GIVE MAN, and so the former is the preferred template. The preference of GIVE is for a physical object to be the thing given (not a human). So the preferences embodied in the individual formulas are used to indicate the preferred meaning of the whole text fragment. (The preferences are built into the word formulas by having a first section like (*ANI SUBJ), which means that the preferred subject (the agent) is an animate element.)

Pairs of templates are linked together by "TIE" routines into *paraplates*. A *mark* template and a *case* template are linked. For the sentence *He ran the mile in four minutes* a paraplate, attached to *in*, links the mark template on *he ran the mile* to the case template on *four minutes.* The patterns within the two linked templates are used to discover that the linking paraplate is a TIMELOCATION paraplate, rather

than any other sort (such as a CONTAINMENT paraplate, e.g. as in *He ran the mile in the sports hall*). Again the decisions are made on the basis of preferences. This also works with sentences such as *He put the key in the lock* where "lock" can mean part of a canal or part of a door.

The PS TIE routines also handle simple anaphora on a preference basis. For example, in the sentence *I bought the wine, sat on a rock, and drank it* the preference of *drink* for a liquid object is used to tie "it" to *wine* (rather than *rock*). But this does not work for cases where the possible references are both in the same class, as in *The soldiers fired at the women and we saw several of them fall.* For these, a set of *common sense rules* is applied to an *extraction* from the templates. This uses a common sense rule such as "struck things fall" and matches this against the extraction "the women were struck". The latter is condensed from the templates/formulas.

The text is originally fragmented prior to matching against the 3-head bare templates using an extensive list of *key words* to indicate fragmentation points. This list includes "all punctuation marks, subjunctions, conjunctions, and prepositions". The fragmentation process also uses other heuristics where no key words are found, although these are not described in the paper. (Sentence fragmentation is an important topic in this thesis, since the novel pattern-matcher under test (described in Chapter 4) fragments sentences (using key phrases and punctuation) prior to pattern matching.) Where there are alternative fragmentations for a sentence, PS determines its preferred fragmentation using individual preferences of the agents involved. Thus for the alternative fragmentations *I heard an earthquake / singing / in the shower* and *I heard / an earthquake singing / in the shower* the preference of animate agents (such as *I*) over inanimate agents (such as *earthquake*) for notions such as *singing* ensures that the former is preferred.

As is the case with CD, PS seems to have quietly faded from view over recent years, although these systems have left a legacy of new systems incorporating many of their concepts, such as the use of primitive actions within virtual reality systems. Today the areas of interest seem to have shifted towards shallow systems. This may in part be due the realisation of the scope of the NLP problem as revealed by systems such as CD and PS, but it is also undoubtedly the result of pressure to come up with "working" systems within short timescales and budgets. Large-scale deep processing NLP projects do of course still exist, particularly when funded by multi-country organisations such as the European Community, but for now the shallow approach is in the ascendant. In the next section some of these systems are considered.

2.3.2 Shallow Processing NDS Systems

It is very difficult to find any examples of KE or IE systems which are shallow *and* NDS. The KEP system appears to be novel in this respect. Instead, a program is described here which really lies in the content analysis or text summarisation fields, and which has as its application the information retrieval

(IR) field, as defined in Appendix A - Nomenclature of KE-related fields. Also described is a typical dictionary definition processor program, one of several such systems which rely on the fact that dictionaries are in effect collections of definitions (thus removing the need to find instances of definition in running text).

2.3.2.1 The COMMIX system

The COMMIX system of Norris (1996) is not a KE or IE system which extracts individual facts or pieces of information from text, but it does capture a glimpse of what a shallow-approach non domain specific system looks like. The purpose of COMMIX is to produce a single phrase which captures the "aboutness" of a text. It creates a sort of "super abstract" of the given text, condensed into a single phrase (the intended target application being condensation of abstracts, for use by an IR system). For a *deep* system to do this there would have to be the creation of a structure containing the understanding of the abstract (e.g. a CD-like representation). This understanding would then have to be condensed to a single sentence. Both of these stages would require extensive processing involving lexicons, syntactic knowledge, semantic processing, pragmatics, and discourse structure knowledge. The *shallow* approach taken by COMMIX is to use part of speech information together with a MRD and heuristics to produce a compound nominal term. No attempt is made to parse the text (syntax) or to find its whole meaning (semantics), and yet the aim is to produce an output phrase which contains within it an encapsulation of the subject of the text.

An example heuristic is to look for phrases M1 N and M2 N separated in the input text, where M1 and M2 are modifiers, and N is a noun. Then the compound term M1 M2 N is constructed. For example, **big dog** and **black dog** give rise to **big black dog**. This is the simplest sort of rule, for the 'semantic relatedness' of the two nouns is clear (they are the same noun). However, another example occurs where M1 N1 and M2 N2 are seen, with N1 and N2 different, and where N2 appears in the definition of N1 (in a MRD or MR thesaurus). The semantic relatedness here is lower but nevertheless exists. An even weaker overlap occurs where N1 and N2 share a word W which occurs in both of their definitions. These techniques have been suggested in a similar manner by Jobbins and Evett (1995) for use in automatic evaluation of textual cohesion and in Rose and Evett (1993b) for calculation of definitional overlap.

> *The video games industry is growing fast and will dominate the toy market and become an established part of home entertainment. The 1991 computer games market was worth 275 million pounds sterling growing to 500 million in 1992, half the toy market. Hardware sales will rise from 261 to 635 million pounds sterling in 1994. Associated software sales are forecast at 645 million pounds sterling in 1993. The compact disc market is worth 345 million pounds sterling. The main competitors in the market are Sega and Nintendo. Nintendo will spend 15 million pounds sterling on advertising over Oct-Dec 1992.*

Figure 3. Sample input text for COMMIX (from Norris (1996))

Figure 3 shows a test input for COMMIX. Processing proceeds as follows. Firstly, all closed class words (determiners, auxiliaries, prepositions, conjunctions etc) are deleted and all open class words labelled with their syntactic classes (as listed in WordNet, Miller et al. (1990)). Then all existing compound nominals are identified and counts of nouns occurring more than once are made. This yields compounds such as *video games industry*, *growing fast*, *toy market*, *associated software sales*, *hardware sales* etc. and the noun count list:

games 2 toy 2 market 5 worth 2 million 6 pounds_sterling 5 sales 2

The definitions of these nouns are then retrieved and processed to remove closed class words and words there merely to aid in description, and the open class words are labelled. Matching can then be done between the definitions in the ways described above. For example, *hardware sales* and *associated software sales* give rise to *hardware associated software sales*. Links so created are weighted based upon the degrees of semantic relatedness, so that the correct pairs of phrases can be combined in the correct order. The final nominal phrase is constructed by linking salient nouns in such a way that the most salient is used as the head term. For example, *hardware associated software* ***sales*** and *toy computer games compact disc* ***market*** would be linked as *toy computer games hardware associated software compact disc* ***market***, because **market** has a higher count than **sales** (see above). This forms the final output phrase available to IR systems.

The paper Norris (1996) does not report any systematic evaluation of COMMIX, merely stating that its performance is "encouraging". Although the processing of one test text is followed in some detail, it is difficult to know how many other texts have been used and whether the outputs produced by COMMIX were deemed worthwhile. Given that the output statements are in the form of one large noun phrase, it is difficult to know how one could score each output for correctness. Norris does not discuss this problem.

Note that the reliance on WordNet reduces the NDS-ness of COMMIX. This means that COMMIX would not work effectively on very specialised domains having terms not present in WordNet. However, given that WordNet is a general resource which does currently exist, it is justifiable to label COMMIX as an NDS system. On the other hand, one should be aware of the counter-argument, which is that it could be argued that **any** system which *could* become NDS if the relevant lexicons etc were added *should* be regarded as NDS.

It is also debatable as to whether COMMIX should be labelled as a shallow processing system. Although no attempt is made to understand the input text, comparative semantic knowledge is used, i.e. *meaning* is used in the system. However, the process does not attempt to find the meaning of the whole text. What is more, some of the operations even dispense with the WordNet look-up, e.g. in the M1 N, M2 N combination heuristic. Thus the system is indeed quite shallow.

2.3.2.2 Alshawi's Definition Analyser

Alshawi is just one of the various researchers who have attempted the processing of definitions from machine readable dictionaries (MRDs). Other attempts include that of Martin (1992), (although this system handled only medical definitions, and so was DS), Zhu and Shadbolt (1995) and Chodorow (1985). In the sense that a non-specialist dictionary is a collection of definitions on *any* topic, this is a NDS field. MRDs have also been used for purposes other than definition processing, such as with Jensen and Binot (1987 and 1988) who used an MRD to resolve ambiguity of prepositional phrase attachment.

Alshawi (1987) describes a program which reads Longman Dictionary of Contemporary English (LDOCE) entries and builds semantic structures from them to extract the *semantic head* and other information. For example, for the noun **mug** in its 'gullible person' sense, the semantic head would be *person* rather than drinking vessel. Other pieces of information attached to 'mug' by the program include the property *foolish* and object-of class *deceive*.

The algorithm is described as follows. The analysis mechanism "has the flavour of a pattern-based phrasal analyser" which "...was designed to overcome some of the more obvious difficulties of applying a simple pattern matching approach to robust phrasal analysis". The approach is to use a hierarchy of phrasal analysis patterns in which less specific patterns dominate more specific ones. This is done as follows. The input definition is taken and attempts are made to match patterns with it. If a match with a pattern occurs, attempts are made to match the input definition with each of the matched-pattern's daughter patterns. These daughter patterns are more specific (detailed) forms of the successful top-level pattern. This process continues until the most specific matches against the input pattern are reached. This progression towards more-specific "parsing" is the reverse approach to that taken by many other robust parsers, which try to start with the most detailed parse, and relax these to more general parses if need be.

The hierarchy of patterns is stored as a list of analysis rules. Each analysis rule has the form:

(rule_id (pattern) rule_id, rule_id ...)

The leftmost rule_id is the name of the rule, e.g. n-100. (n-100 is the topmost i.e. most general noun phrase rule). The middle part is the pattern to be matched against the input text. The rightmost list of rule_ids is the list of the daughter (more specific) rules, which are applied if the leftmost rule_id is found to match. These daughter rules have the same format as above. Here is an example pattern for a rule (this is for n-100):

(n && +0det && &0adj &noun &&)

The meanings of the elements in the above are as follows:

n	noun pattern
&&	one or more words (equivalent to KEP's 'X' token - see section 4.6.10.1)
+0det	zero or one determiner
&0adj	zero or more adjectives
&noun	one or more nouns

Thus rule n-100's pattern would match phrases such as *the large black dog*, *a small but brightly-coloured buzzing insect*, *Big mountains by the sea* etc.

Now consider rule n-110, one of the daughter rules in the RHS list of n-100:-

(n-110 (n +0det +0intens &0adj &noun *0pp-mod &&))

In this rule, *0pp-mod means zero or one PP modifier; the asterisk indicates that this element is itself expandable into sub-elements. The last of the three example phrases above fits the pattern for this rule; obviously it is a more specific pattern-match than that given by n-100. Also, in the case of n-110, there are no daughters to match. Note that individual words can be part of a pattern, too.

Once the pattern matching has gone as far as possible, the analysis rule's associated *structure-building rule* is activated. Here is n-100's structure building rule:

(n-100 ((compound-class &noun) (properties &0adj)))

A phrase which matched n-100's analysis rule but none of its daughter rules would use the above to bind a noun (or nouns) to &noun, and adjective(s) to &0adj. That would be the end of the semantic structure building process. But if the matching process finished at a more detailed level, the relevant building rule would extract more information. (Note: This binding process is equivalent to KEP's matching of C (Concept) to X (some words) and 0,1,2... (examples etc) to X (some words). This is explained in Chapter 4.) In fact, all paths are followed i.e. there might be more than one extraction, although this rarely happens to a deep level. This is equivalent to KEP's use of all tokenisation combinations. In Alshawi's program the method used to select the correct extraction is simply to pick the one that accounts for most words in the input.

The Alshawi paper does not explain in detail how the patterns are matched against a text fragment. But given that this pattern matching can be achieved, the method demonstrates that it is indeed possible to do knowledge extraction with a pattern matching approach. The output data structures could presumably be used to build a semantic net. Thus this system appears to be able to read LDOCE and "understand" the definitions to a level from which a large interconnected knowledge base could in theory be constructed.

What is more, it is robust in that even if very detailed knowledge cannot be extracted, at least *some* information can be retrieved. The paper states that 77% of semantic heads were extracted correctly, and 88% of other information.

The above illustrates the power of pattern matching techniques, and demonstrates how a system may avoid parsing and yet still give high precision rates. The task is made easier here because of the nature of the communication channel i.e. because of the lack of the need to *find* the definitions in running text. It nevertheless demonstrates a shallow technique in action.

2.4 Domain Specific Systems

2.4.1 Deep Processing DS Systems

2.4.1.1 The MEDLEE System

MEDLEE is an IE system designed to extract information from medical (radiology) reports, and was developed at Columbia-Presbyterian Medical Center (CPMC) (Friedman et al. (1995)). MEDLEE stands for **MED**ical **L**anguage **E**xtraction and **E**ncoding System and is integrated with the clinical information system at CPMC. It is thus a working system. Written in Prolog, MEDLEE performs syntactic and semantic analyses on sentences taken from radiology reports (chest X-rays and mammograms). The input is a short machine readable report derived from dictated comments which includes some fixed fields and some free NL fields. However, within these the terminology used by the medical practitioners is constrained in the sense that certain phrases are used consistently (e.g. *cannot be excluded*, which means 'low possibility'). In addition, technical terms specific to the domain abound (e.g. cardiomegaly, retrocardiac opacity etc). Output is into a structure designed to hold clinically salient information, based on information formats of the Linguistic String Project (Sager, Friedman and Lyman (1987)). This **Rad Finding Structure** comprises various slots such as **Central Finding** (the main finding of the report), **Bodyloc Mod** (body location information), **Certainty Mod** (certainty rating of central finding, which can be one of **no**, **low**, **moderate**, **high**, **cannot evaluate**, and **rule out**) etc.

The first stage uses a definite clause grammar (DCG) to zone the document, so that the free text sections may be identified. There are four such sections: *Report*, *Clinical Information*, *Description* and *Impression*. It is these sections which are of interest here. The first act is to chop them into their sentences (by a method not described in the paper). Clinically nonrelevant phrases are then removed (e.g. *on the basis of this exam*, which adds nothing to the findings). This is done using a list of commonly-used phrases. Phrases which should be regarded as atomic, such as *no conclusive evidence of*, are then bracketed. The output of this first stage of processing is a form suitable for input to the next stage, the parser. For example, the *Impression* field may contain the sentence *An infiltrate cannot be excluded on the basis of this examination*, and the output of the first stage for this sentence would be:

[infiltrate, [cannot, be, excluded],.]

The parser stage is now applied. This is mostly a semantic parser rather than a syntactic parser, since the first stage effectively performed the latter. Using the semantic categories of the words in the sentence, the parser firstly determines which types of high-level production rules to apply. For example, if *cardiomegaly* is spotted in the sentence, then the parser will look for phrases such as *increasing cardiomegaly*, *increase in cardiomegaly*, *cardiomegaly increased*, *cardiomegaly has increased* etc. These are syntactically different but semantically equivalent. The semantic head is *cardiomegaly*, and this is modified by a temporal qualifier that is a form of *increase*.

The parser uses a semantic grammar which resembles a DCG but is in direct Prolog form. In order for a parse to be successful, the parser must be able to identify all the words in the sentence and the sentence must match one of the allowed semantic forms. If a sentence cannot be parsed, it is segmented around connectives such as the phrase 'may represent' in the sentence *Opacity may represent effusion*. The smaller segments are then parsed. Note then that the parser is designed to handle fragments of sentences, and not just whole grammatical sentences. This is important in a domain where the dictated reports often contain fragments of text, such as verbless phrases, due to the production process.

The next stage is a compositional regularizer. This ensures that phrases such as *enlarged heart* and *heart appears to be enlarged* are both reduced to **enlarged heart**. A database of mappings of multi-word phrases is used to do this. However, in order that the output of MEDLEE is compatible with terminology used in other clinical information systems, an *encoder* stage is now performed. This maps synonym terms to a preferred single term found in a medical entities dictionary (MED), using couplets such as:

synonym('enlarged heart', cardiomegaly)

Thus in the final output the **Central Finding** slot is set to **cardiomegaly**, whatever the wording used in the report to indicate the fact that the patient's heart was enlarged.

In the words of Friedman et al. (1995), "MEDLEE became an integral part of the operational clinical information system after two independent evaluation studies demonstrated that it performed comparably to experts under certain circumstances". Thus MEDLEE is a demonstration of an effective IE system which is actually being used on a routine basis.

Clearly, MEDLEE is a DS system. The various processing stages described above require medical lexicons, MRDs and databases, more specifically those relating to radiology and radiological examinations. The "nonrelevant phrases" removed in the first stage of processing are also domain

specific (the domain being that of medical examinations). Much of the processing relies upon the fact that the medical practitioners dictating the reports use a common sublanguage with great regularity. The various symptoms that may occur and the medical conditions they suggest form small sets.

MEDLEE has been categorised here as a "deep processing" system because the parser is a semantic parser capable of extracting *meanings* from sentences having varying syntaxes. Although a full syntactic parser is not used on the initial sentences, the processing does reach the semantic level, and it does involve large amounts of knowledge in order to come to an understanding of the text.

2.4.1.2 The ATRANS System

The ATRANS program is an information extractor for interbank money transfer telex messages (Lytinen and Gershman (1986)). The program is domain specific, and robust. It uses a 'semantically-based predictive conceptual analyzer' to produce a CD-style representation of the message content. ATRANS is a large and complex system and it is therefore not possible to detail its operation here. However, certain aspects of the system are of interest, such as its use of stacked mini-scripts as described shortly.

Interbank money transfers usually follow a simple script or a variation on this (of the form *Customer C asks Bank A to send money M to beneficiary X, Bank A requests Bank B to transfer M to local Bank L* and so on), and so the first stage of processing (the *message clarifier*) is to choose the correct script version based upon various clues including document layout. The telexes used in interbank transfers do not have to conform to well-defined layouts, although as in any domain, specialist terminology has arisen and certain fields are obligatory, such as the sender, target recipient, amount to be transferred etc. Thus the message clarifier must be able to decide upon the particular script version despite this uncertainty in layout. When this has been done, the *text analyser* stage is brought to bear.

The task of the text analyser, the heart of the system, is to process each telex to produce a CD representation of it. An interesting feature of the text analyser is the use of local context for lexical disambiguation. ATRANS uses a hierarchy of lexicons/scripts, with the most specific lexicon made active (by local context) and then used to disambiguate problem words. For example, if the activated lexicon is the one for "sender details", then any number encountered will be interpreted as a *date* (rather than as a monetary amount) because the sender lexicon has no entries relating to money but does have entries such as "sent date" etc.

Thus the lexical disambiguation problem has been transformed into a different problem: how to activate and end local contexts. This is a problem which has been studied by NLP practitioners and linguists interested in the structure of discourse, such as Grosz and Sidner (1986), who suggest that texts contain three structures: the actual utterances (linguistic structure), an intentional structure (a structure of

purposes), and an attentional structure (a state of focus of attention). These overlapping structures allow text to be segmented accordingly. However, this is at present a manual technique. Automatic techniques include that of Crowe (1996) in the CONTESS system. Here, a clause-event grid is constructed for news reports, where *events* are incidents as specified by the MUC guidelines. Three parallel-runnable analysis modules are used to segment the text into events: temporal phrase analysis, cue phrase analysis, and location phrase analysis. Each of these finds pairs of clauses which cannot refer to the same event. An event manager stage then produces a single grid representation which shows the event nesting structure of the input text. Another automatic technique is that of Hearst (1994), who presents a method which segments a text at paragraph boundaries based upon subtopic structure based upon term repetition alone.

The method adopted in ATRANS is relatively simple. Whereas a lexicon can be made active by the detection of certain key words and phrases, its deactivation is triggered by the detection of words *not* in the current context. For example, to disambiguate *beat* in *John raced Mary; Mary won. John got angry and beat her* we need to know that the 'race' context ended after the word *won*, where the 'conflict' context starts, as triggered by the word *angry*. The method used is to stack contexts, so that when an encountered word is not related to the current context (i.e. not in its lexicon) but is related to a previous context, then the current context is ended and the previous one popped off the stack to become the new current context.

The third stage of processing is the *message interpreter*. This verifies the extracted information (in the CD construct) and checks it for consistency. Databases containing banking knowledge are used to check bank names, customer names and account numbers. The output from this stage goes to an *output formatter* stage.

Lytinen and Gershman state that "in contrast to other message-parsing systems such as FRUMP[3] or TESS[4]..., ATRANS carefully analyses every word in a message, producing a highly-detailed representation of its content." Thus ATRANS falls firmly into the deep processing camp. The lexicons used in the hierarchy and the banking KBs are clearly very domain specific, so ATRANS is DS system. The communication channel in this case is that of a telex message which contains certain mandatory fields, but not in any defined format or order. ATRANS does not have to parse full sentences, but all of the words in a message are considered.

The paper Lytinen and Gershman (1986) does not say much about evaluation of ATRANS, other than that "ATRANS is currently undergoing live testing at a major international bank" and that "the average

[3] FRUMP is a newswire IE system; see DeJong (1979)

[4] TESS is a banking telex summariser; see Young and Hayes (1985)

processing time on a VAX 11/785 is under 20 seconds per telex". It is difficult to form an impression of the effectiveness of the ATRANS system without precision and recall figures.

2.4.1.3 The SCISOR System

The SCISOR system described in Rau and Jacobs (1988), Rau, Jacobs and Zernik (1989), Jacobs (1990) is a deep processing DS system which extracts information from newspaper stories about corporate mergers and acquisitions. Figure 4 shows a typical input text and a User/System dialogue concerning it (source: Rau and Jacobs (1988)).

```
      W ACQUISITION UPS BID FOR WARNACO
Warnaco received another merger offer, valued at $36 a share, or $360
million. The buyout offer for the apparel maker was made by the W
Acquisition Corporation of Delaware.

User: Who took over Warnaco?
System: W Acquisition offered $36 per share for Warnaco
```

Figure 4. Example SCISOR input and dialogue (from Rau and Jacobs (1988))

The first point to note is that SCISOR uses both bottom-up (parsing) and top-down (conceptual) methods in attempting to extract information from texts. The conceptual approach is valid because, in a limited subject domain, certain types of information (concepts) can be expected. SCISOR actually uses four sources of knowledge in order to extract information. Firstly, role-filler expectations are used. In a takeover, there will always be the roles of suitor and target. These roles can be filled by companies, but not by other entities. This role-filler expectation knowledge can be used when there is ambiguity in the other sources of information, described shortly. The second source of knowledge used by SCISOR is that of Event Expectations. Using a script-like approach the output from one text is used to partially fill script instantiations which processing of later texts can call upon. Thus if one text was about a rumoured takeover bid of ACME by Universal Widgets, then a later story about ACME's acquisition by Universal Widgets can call upon the role-filler knowledge already stored. Thus SCISOR has a temporal aspect.

The third source of knowledge for SCISOR is linguistic, i.e. lexical and grammatical. The TRUMP parser (detailed in Jacobs (1987)) is used for bottom-up processing of text. It is a flexible language analyser consisting of a syntactic processor and a semantic interpreter. Within SCISOR, TRUMP identifies linguistic relationships in the input, using lexical and syntactic knowledge. Knowledge structures so produced are improved by the expectations employed by SCISOR, e.g. expected role fillers. Thus full syntactic/semantic parsing is performed, one of the reasons for classifying SCISOR as a deep system. It is not feasible to describe TRUMP in detail here, but Figure 5 shows the final TRUMP output for the given input sentence.

```
W Acquisition offered $36 a share for Warnaco.

(offer
        (offerer W_Acquisition)
        (offeree Warnaco)
        (offer (dollars (quantity 36)
                (denominator share))))
```

Figure 5. Example TRUMP output (from Rau and Jacobs (1988))

Lastly, SCISOR uses domain knowledge. This is encoded in heuristics of the form "If A then B". For example: If it is ambiguous whether ACME is taking over UW, or UW taking over ACME, then choose the larger company as the suitor and the smaller as the target. This is obviously very useful in disambiguating certain situations. In the words of the authors, "...there is a great deal of 'common sense' information that can increase an understanding mechanism's ability to extract meaning."

The process of using the four sources of knowledge to extract information from text works as follows. The text is scanned for apparent events (e.g. rumour of a takeover), role-fillers obtained where obvious, and event expectations set up whenever possible. SCISOR then processes the text in detail, using linguistic and domain knowledge to fill in roles and close events. Linguistic knowledge can be used to determine which company is the suitor and which the target, or if this fails, domain heuristics are applied.

The initial **skimming** phase used by SCISOR has been described in Jacobs (1990). It has strong echoes to KEP which skims (scans) text looking for exemplifications etc (described in a later section). The skimmer described effectively scans for words and phrases (lexical items) which indicate the presence of an occurrence of a concept, e.g. the *corporate takeover* concept. Sections of text which do not contain concepts cannot be fully processed by SCISOR, so they are discarded. (This is also what KEP does when looking for exemplifications, definitions etc.) Once an information-rich piece of text is encountered, the various roles in the concept are filled out on an initial basis. This is done by categorising the lexical items either as *triggers* or as *role fillers*. Note that the triggers do not have to be single words - combinations of separated words may be used.

The discarding of unwanted parts of the text, and "attachment" (who does what, who suffers what etc) are actually performed in a bottom-up (i.e. parsing) way. Thus this skimming system is not just something which highlights interesting text; it actually starts some of the analysis of the relevant sections. The interesting aspect of this processing is the order in which it is done. Parts of the text are discarded *before* the attachment is performed. Thus the discarder must perform limited parsing so as not to throw away text which is needed later.

Jacobs makes an interesting point when he says that pure template-based approaches (for top-down i.e. conceptual processing) fail if the conceptual information cannot distinguish between roles. For example, in company takeovers, both the suitor and the target are companies. Thus one must use syntactic clues to determine which company is the suitor.

Clearly SCISOR is a deep system because (although it does not parse the whole of a text) it does do full parsing on those extracted parts. It also applies WK in order to understand correctly the roles of entities taking part in the events described in the input texts. SCISOR is also DS, because domain-dependent WK is used.

The paper Rau and Jacobs (1988) does not include any discussion of evaluation of SCISOR. However, in the intervening two years between the publication of this paper and the Jacobs (1990) paper a certain amount of evaluation must have been performed, since Jacobs (1990) states that SCISOR is "... a completed prototype that reads news stories at the rate of about 500 per hour. It extracts certain key information..., identifying target, suitor, purchase price, and other information with about 90% accuracy." However, no recall figure is given.

2.4.2 Shallow Processing DS Systems

2.4.2.1 The JASPER System

JASPER (Journalist's Assistant for Preparing Earnings Reports) is a "fact extraction system" developed by Carnegie Group for Reuters (described in Andersen et al. (1992)). In the words of the authors, "JASPER uses a template-driven approach and partial understanding techniques to extract certain key pieces of information from a limited range of text". Thus, like ATRANS, and indeed SCISOR, JASPER is a DS system aimed at automating a specific task for a commercial organisation. Having said this, JASPER is distinguished by the fact that it has a domain independent (i.e. NDS) core module to which DS modules are attached. Thus it could be argued that JASPER is potentially NDS.

The input to JASPER is a live feed of company press releases (PR). Only those press releases containing details of company earnings or dividends are selected (selection stage), and these are processed to extract a predetermined set of facts (extraction stage). These facts are then reformatted as a candidate Reuters news story which is passed to a financial journalist for validation and completion. It is claimed that JASPER thus "improves both the speed and accuracy of producing Reuters stories and hence provides a significant competitive advantage in the fast-paced world of financial journalism". This is a bold claim; if a human reader is to trust the system, i.e. if he is not to have to check the original text every time, then precision should be very high. Also, high recall rates are desirable if the system is not to miss out completely on relevant stories buried in the newswire feed.

For the selection stage, both recall and precision are given as between 95 and 97%. Thus JASPER is very good at detecting the stories of interest. For the extraction stage, Andersen et al. use the terms *completeness* and *correctness*, which correspond closely to recall and precision, and which are based upon the individual targeted pieces of information (facts) which are potentially present in a given story. In this stage, completeness hovers around the 75% mark, and correctness around 90%. The emphasis was deliberately placed on correctness rather than completeness by the system's designers because it was thought that reporters were less likely to overlook gaps in the output than errors. Although these figures are very good, clearly human input is required to check the extractions. This is brought home by the fact that only 21% of earnings stories and 82% of dividend stories are handled perfectly (with a 33% perfect-rate overall). Thus although JASPER extracts the majority of target facts from the newswire, if used as a completely automatic system its reports would contain errors in two cases out of three.

JASPER achieves the above performance using frame-based knowledge representation, object-oriented processing, pattern matching, and heuristics. The heuristics take advantage of stylistic, lexical, syntactic, semantic and pragmatic regularities observed in the input stream. Andersen et al. describe the approach as "shallow, localized processing". They also mention that although JASPER has a NDS core processor, the DS knowledge base required for a specific application involves a significant knowledge engineering effort (about 8 man-months in the case of the company earnings/dividends applications). For this reason JASPER has been classified as essentially a DS system. Furthermore, the communication channel is restricted to newswire input (JASPER does not read newspaper articles, or books) and this further reinforces the categorisation of JASPER as DS.

The newswire story selection stage utlised by JASPER is Carnegie Group's Text Categorization Shell (TCS) (Hayes et al. (1990)), which is itself a shallow system and which is over 90% accurate in its categorisation decisions. TCS works by pattern-matching for concepts in the text, applying rules to assign a text to predefined categories. Rules and concepts are domain specific and require some knowledge engineering. The selection stage of JASPER selects about 1 in 5 of the newswire stories as relevant, and generates some of the time savings provided by JASPER. Other time savings are made by the automatic creation of a standard format Reuters story ready to be edited. Andersen et al. report an average creation time of 25s for each ready-to-edit story, which is extremely fast compared with likely processing times of systems utilising full parsers.

In the extraction stage, the selected stories are matched against a frame of slots, the slots defining both the fact to be searched for and the method of processing to achieve this. Each sentence in the selected story is processed for each slot, by attempted word-pattern matching. (Thus JASPER is similar to KEP in that the basic pattern-matching unit is the sentence. KEP's approach is described in Chapter 4 of this thesis.) If a pattern match is found, the process associated with the slot is called in order to assign a value to the slot, or reject the match as not useful. Once all sentences in the story have been processed for all

the slots in the frame, the partially filled frame is used as the basis of the editable news story. (It is rare for all slots to be filled since the complete set of relevant facts is rarely present in the source - the earnings application, for example, has 56 target facts.)

The frame with its slots, slot patterns and slot processing methods make up the DS part of JASPER, termed a *rulebase*. The pattern matching mechanism is able to match several patterns to one sentence (and vice versa) and includes the notions of optional elements, skipping over runs of words, automatically generated morphological forms (for nouns and verbs), and negation amongst other things. Let us consider an example pattern:

```
(profit +N ! earnings)
(&skip 8 ($n ?million dollar +N) )
(&n (per share))
```

This pattern is interpreted as follows. Note firstly that the pattern comprises three parts (one part on each line of the pattern, bracketed), and that it is intended to match text having three parts in the same order. The first part is to match the word *profit* or *profits* (the +N signifying that *profit* is a noun and so may be present in its plural form) or the word *earnings*, the exclamation mark being the OR operator. The second part means that this is to be followed within 8 words (&skip 8) by any number ($n), optionally the word *million* (?million), and then the word *dollar* or *dollars*. The third part says that this is not to be followed (&n) by the phrase *per share*. The pattern may be matched against a whole sentence or part of a sentence by the pattern-matcher. Thus it would match the sentence:

ABC Company announced profits of more than 50 million dollars last year.

However, it would not match:

XYZ Company's profits will be 2.25 dollars per share.

The pattern matcher is sophisticated enough to recognise different morphological forms for nouns and verbs and so removes the need for the person specifying the patterns to list all possible forms. The negative-match facility is a powerful feature which allows many unwanted forms to be rejected early (rather than in a post pattern-matching stage) without the need to list them all. The skipping facility is also a very useful feature, and corresponds to the X-token mechanism in KEP's pattern matcher (described in Chapter 4).

A successful pattern match results in the binding of a phrase from the text to a pattern variable, thus extracting the bound item. This can be done in such a way as to allow different items in the text to set the pattern variable to the same value. For example, *fourth* and *4th* in the text can both cause a pattern variable called `%quarter` to be set to the value 4. This means that slot fillers can be written in a

consistent way, e.g. so that the Quarter slot is always set to one of the numbers 1,2,3,4. Slots also have allowed filler classes and are grouped together accordingly. For example, several slots may take a value which is a net income figure. These slots would be labelled as taking a <net-income-object> value, and all such slots would be classed as <net-income-group-object>. Thus a frame is not simply a collection of slots, but can contain hierarchies of types of slot.

JASPER handles multiple pattern matches to a sentence using "a heuristic procedure", which unfortunately is not described in the paper Andersen et al. (1992). Presumably the heuristics encode domain knowledge (such as when earnings rises, the final number is larger than the starting number). The resolution of multiple pattern matches to a single sentence has proved an important process for the KEP program described in Chapter 4. As will be seen, since KEP is not DS it is not possible to use DS heuristics. Thus the problem is actually more difficult for KEP. The techniques used by KEP are described later.

JASPER also attempts to overcome time context problems arising through its focus on separate sentences. For example, in the following couplet the Sales referred to in the second sentence are 4th Quarter Sales:

> *Earnings during the fourth quarter of 1990 were 50.5 million dollars. Sales were 74.3 million dollars.*

The above illustrates a form of indirect anaphora (see the following chapter for a discussion of anaphora) and, as with all instances of endophors, will cause problems for any KE/IE program which works on a per-sentence basis (such as KEP). JASPER attempts to solve the time context problem by using a persistent time context which is changed when certain syntactical or semantic clues are found in the text stream. However, Andersen et al. report that this method is not always successful (although no figures are given for error rates caused by this).

JASPER is a good example of a very successful, fast, shallow IE/KE system, based on pattern matching. It demonstrates what can be achieved without full syntactic and semantic processing stages. Although it requires a certain amount of human supervision (acting more as an assistant than as a completely automatic journalist), it is clearly a system which is cost effective for the task of sifting huge amounts of text for specific types of information. Systems like JASPER will undoubtedly become standard tools in the MU task domain, where one can envisage many copies of the system simultaneously scanning the incoming text stream for information pertinent to their own target domains.

2.4.2.2 The FASTUS system

The FASTUS system (Myers and Mulgaonkar (1995), Appelt et al. (1993)) performs the information extraction function in a printed document IE system developed by SRI International. Based on a series of cascaded finite state automata, NL text is searched for specific logical structures of interest. FASTUS is domain specific; terrorist incidents form the target subject. However, the communication channel is not restricted (to e.g. telex messages), because full NL text, as found in newspaper reports etc is processed. (It might be argued that the newspaper medium does represent a distinct communication channel, but even so this channel is not as constrained linguistically as that of telexes etc.) FASTUS took part in MUC-4 (the fourth Message Understanding Conference) and achieved 44% recall and 55% precision figures.

Four stages of processing are performed: (1) triggering, (2) phrase recognition, (3) domain pattern recognition, (4) incident merging. Triggering involves scanning sentences for keywords, such as 'terrorist', 'killed' etc. There is a similar stage in KEP (see section 4.6.7). Phrase recognition then cuts the triggered sentences into noun groups, verb groups and particles. Domain pattern recognition involves scanning phrases output from the previous stage for patterns of interest, so that *incident structures* can be built from them. This stage represents pattern matching at the semantic level. For example, one pattern looked for is:

<Perpetrator> <Killing> of **<Human Target>**

Finally, the incident merging stage attempts to recognise different descriptions of the same incident separated in the input text.

FASTUS has been placed in the "shallow processing" category here because it uses pattern matching in essentially shallow ways. Phrases are recognised using syntactic information and domain knowledge, and patterns of those phrases are then looked for. No parsing is performed on the input text. Furthermore, only sentences thought likely to contain useful information are processed - the rest are discarded. The fourth stage, that of incident merging, is an ingenious idea which allows two separate extractions to create one single, more reliable, extraction. This helps to overcome the complete lack of semantic processing (text understanding). The fast, shallow approach was a deliberate stance taken after experiments with TACITUS (a previously developed deep system) showed that "it was wrong to approach the information extraction task as a 'traditional' computational linguistics problem" - Appelt et al. (1993).

It is the contention of Appelt et al. that finite-state models (regular grammars) can achieve more than was previously thought possible, despite the fact that English has constructs (such as centre embedding) that

cannot be described by a finite state grammar. A 37-state nondeterministic finite state automaton is used in FASTUS's phrase recognition stage to identify noun groups. By noun group is meant the head noun of a noun phrase together with its determiners and other left modifiers. This allows noun phrases such as the following to be recognised:

approximately 5kg
more than 30 peasants
the newly elected president
the largest leftist political force
a government and military reaction

Note the similarity of these phrases to those found by rule n-100 in Alshawi's program. Since regular languages (those generated by regular expressions, as used by Alshawi) can be accepted by FSAs (Kleene's theorem[5], see Cohen (1986)) this is not surprising.

Similarly, verb groups (the verb together with its auxiliaries and any intervening adverbs) are handled by an 18-state FS machine. All verbs are tagged as Active, Passive, Gerund or Infinitive, although in some cases the automaton is not able to distinguish Active/Passive, as in the sentence *Several men kidnapped yesterday were released today*. In such cases, the pattern recognition stage is left to make the decision. Although some relevant adjectives and adverbs are recognised, most are simply ignored.

Recognised phrases are passed onto the pattern recognition stage, which processes them in the order they occur. For example, the sentence *Guerrillas attacked Merino's home in San Salvador 5 days ago with explosives* is turned into the string of phrases:

(Guerrillas) (attacked) (Merino)('s) (home) (in) (San Salvador) (5 days ago) (with) (explosives)

This matches the pattern:

<Perp> attacked **<HumanTarget>**'s **<PhysicalTarget>** in **<Location>** **<Date>** with **<Device>**

In order for this match to be recognised, the DS knowledge must include lists of perpetrators, human targets etc. Although some research has been done on the automatic recognition of proper names in text (see e.g. McDonald (1992), which describes a system (SPARSER) which extracts incidents of job appointments reported in the Wall Street Journal), it is not clear whether FASTUS uses any such system. FASTUS does however link bare surnames to full names occurring in previous text, thus allowing full names to be used in the output incident structure if they are ever given in the text.

[5] Any language that can be defined by a regular expression, or a finite automaton, or a transition graph, can be defined by all three methods.

The above pattern match eventually leads to the extraction:

Incident:	ATTACK/BOMBING
Date:	14 Apr 89
Location:	El Salvador: San Salvador
Instr:	"explosives"
Perp:	"guerrillas"
PTarg:	"Merino's home"
HTarg:	"Merino"

FASTUS also carries out a certain amount of "pseudo-syntax" to skip over prepositional phrases when necessary. As the above example shows, it also uses WK to de-reference times and dates and to expand locations. A rudimentary sort of pronoun resolution is also performed: where a pronoun occurs as a human target, an antecedent is sought. The algorithm is simple, amounting to a backwards search (as far as the last paragraph break) for a suitable noun group which agrees in number with the pronoun. The authors claim a near 100% correctness rate with this method, although this seems surprisingly high.

Note that FASTUS, like JASPER (and KEP, described in Chapter 4) essentially processes single sentences. Where this causes problems, i.e. due to anaphoric links, both FASTUS and JASPER use bolt-on heuristics to attempt to retrieve the antecedent. As will be seen later, KEP currently detects anaphoric links starting on demonstrative pronouns which may stand in for a concept (*this*, *these*) but does not yet have a function to follow the links to their antecedents. Neither the JASPER nor FASTUS papers report the percentage of sentences in their specific applications which require antecedent finding.

FASTUS again demonstrates how shallow systems can be successful for IE applications. In the words of Appelt et al. (1993), "Although the full linguistic complexity of the MUC texts is very high... the relative simplicity of the information extraction task allows much of this linguistic complexity to be bypassed." The MUC-4 precision (55%) and recall (44%) figures demonstrate that real, useful systems may be constructed using the shallow processing philosophy in a domain specific application.

2.4.2.3 The 'wit' system

The 'wit' KE system described in Reimer (1989) builds a semantic net as a result of exposure to large amounts of explanatory text. However, this net contains only "is a" links, used to indicate "is a type of" as well as "is an instance of". Furthermore, the concept nodes joined by these links contain slots (created by 'wit' dynamically) which allow other information to be held within the nodes rather than as external links. Thus *printer* has a *manufacturer* slot, say. Partitions (descriptions of sub-parts of objects) are also held as slots rather than as has-part links.

The 'wit' system uses a small amount of domain specific knowledge to commence on a new domain. This is referred to as "knowledge bootstrapping" or closed-loop learning, because newly acquired knowledge is immediately available to aid in the extraction of yet more knowledge. This knowledge

allows 'wit' to "focus its attention" on interesting parts of the text (a sort of triggering, but *semantic* triggering, not syntactic/lexical as used by KEP).

The approach is one which uses term acquisition techniques (see also section 4.6.4). Two major ways of finding new concepts based upon phrasal patterns are used, as illustrated these by examples:

(1) If concept **cartridge** is already known, and **ink cartridge** is repeatedly seen, add **ink cartridge** as a new concept (new hyponym).

(2) If **cpu board** and **memory board** are repeatedly seen, then add the concept **board** as a hypernym of both.

These hyponym/hypernym techniques have been suggested by various researchers, and one of the two techniques (the latter) is used by KEP. In addition to the above two methods, a separate function in 'wit' looks at the results of past parses (of many documents, as held in the growing semantic net) and decides whether to make a new slot within a concept node. This process is given a "certainty" rating, in a manner which is not relevant here. It also merges parts of the network based on is-a links where syntactic information is lacking but semantic information allows it. For example, if *Courier* is-a **font**, and *Helvetica* is-a **font**, then the two **font** concept nodes can be merged into one single node. Slots may also be promoted to entirely new concept nodes in their own right, and this is done when a statement about a slot or its entry has been found in a text.

It is claimed that 'wit' can avoid parsing errors by processing very many texts, so that erroneous parses are eventually dropped. This situation arises almost naturally because the KB takes form as the weight of evidence gradually builds up. Thus 'wit' has a module for extracting possible facts (the parser), and other functions for integrating all extracted facts into a semantic net and maintaining that net (something not attempted in KEP). The latter is done in such a way as to effectively throw away bad extractions. In contrast, KEP attempts to get only *good* extractions. Thus 'wit' uses sheer volume of text (i.e. very many separate texts) to find the good facts hidden within a mass of possibly erroneous facts. This is an appealing idea, and may even reflect the way that humans obtain knowledge.

The 'wit' system has been placed here in the 'shallow processing' category, despite the fact that sophisticated functions are provided to maintain knowledge in the semantic net KB. This is because these functions are not part of the actual extraction mechanism, i.e. the parser. This parser (see Hahn (1989)) is a syntactic/semantic partial parser which outputs a set of propositions which are processed by the KB-updating component as described above, and which does not attempt a full parse of every word in the input. It is thus a reasonably shallow approach. The parser's output depends upon the amount of

WK available, but assuming that sufficient knowledge is provided, the text given in Figure 6 (source: Reimer (1989)) would give rise to the proposition structures shown in Figure 7. In the former the bold text indicates concepts (themes) and the italic text closely associated concepts (rhemes).

The **DeskWriter** from HP is a new ink-jet printer. *Ink* is deposited from a disposable *ink cartridge* at a *resolution* of 300 dpi. The *cartridges* are priced at a reasonable $18.95. The **DeskWriter** comes with four *fonts*: *Courier*, *Times*, *Helvetica* and *Symbol*.

Figure 6. Theme-rheme patterns in "wit" parser sample input (from Reimer (1989))

DeskWriter	**is_a**	ink-jet printer	**is_a**	printer
<manufacturer>		<manufacturer>		<manufacturer>
HP 0.8				
<ink>				
<ink cartridge>				
ink cartridge-1		ink cartridge-1	**is_a**	ink cartridge
<resolution>		<price>		
300 dpi		$18.95		
<font>				
Courier		Courier	**is_a**	font
Times		Times	**is_a**	font
Helvetica		Helvetica	**is_a**	font
Symbol		Symbol	**is_a**	font

Figure 7. "wit" parser output for text in Figure 6 (from Reimer (1989))

In Figure 7, angle brackets <thus> represent slot names and the text on the following line the slot contents. Where a number such as 0.8 occurs, this is a certainty factor attached by the parser to indicate how sure it is about the given knowledge. The 'wit' parser utilises lexical knowledge, syntactic knowledge, knowledge of text coherence and domain-independent world knowledge e.g. about part-whole relationships.

Note that 'wit' is a KE system rather than an IE or MU system. Its aims are very similar to those of the KEP system. However, it is not clear how successful 'wit' actually is in practice, since the papers referred to above make it clear that the parser function was still being written at the time of publication. Furthermore, no references to the 'wit' system dated later than Reimer (1989) have been discovered by the author of this thesis. Other authors closely associated with Reimer (e.g Hahn, who wrote the parser paper referenced above) also do not refer to any more recent papers. For example, in a paper on knowledge acquisition from text dated 1996 (Hahn, Klenner and Schnattinger (1996)) there is no reference to any later paper on the 'wit' system. This suggests that perhaps the 'wit' parser, and indeed the total 'wit' system, did not live up to early expectations as described above.

2.5 A Note on Evaluation

Before concluding this chapter, it is worth making an observation concerning the state of 'reality' of the systems just described. Do all these systems really exist, or are they little more than 'paper' designs? Unfortunately, the latter appears to be the case all too often in the KE field. Many papers describe systems under construction, and hence contain few results of evaluations of their performances. The 'wit' system above seems to be one such design. Other systems, such as FASTUS, are clearly further down the road to real-world application, and one reported system (JASPER) is already in commercial use. The real test must be the existence or otherwise of recall and precision figures - one cannot evaluate a system that has not yet been coded, but if a system has been evaluated then it must be runnable.

2.6 Concluding Remarks

What are the lessons to be learned from the examples of MU, KE and IE systems described above? Although only a small set of programs was described, it is possible to draw out the important trends. These are as follows:

(1) Shallow systems can be successful. This has been demonstrated by the JASPER and FASTUS systems (which are domain specific), and Alshawi's MRD analyser (which is NDS). Many IE systems are shallow systems (see review paper Cardie (1997)).

(2) Domain specific systems can be successful. See MEDLEE, ATRANS, SCISOR, JASPER and FASTUS. Such systems rely on DS WK to facilitate extraction, largely by providing expectations of the information to be extracted. Most IE systems are DS systems (see review paper Cardie (1997)).

(3) NDS systems are usually large systems. This is demonstrated by the scale of Conceptual Dependency and Preference Semantics. NDS systems are only small when the communication channel looked at is tightly constrained (e.g. Alshawi's MRD analyser).

(4) Shallow NDS systems are rare. This is evidenced by the difficulty the author had in finding examples of them in the KE/IE/MU fields. Where shallow NDS systems do occur, they usually reduce the scope of the problem by looking only for specific types of information (such as definitions), or by restricting the communication channel used (e.g. dictionaries), or both of these.

(5) Pattern matching techniques are useful in shallow systems. See COMMIX, Alshawi, MEDLEE, JASPER, and FASTUS. Pattern matching may occur at the semantic level in DS systems, although in NDS systems (e.g. COMMIX) it usually occurs at the lexico-syntactic level.

(6) Part of speech information is useful in shallow systems. Automatic part-of-speech tagging permits lexical/syntactic pattern matching. Other methods of identifying word categories are also used (e.g. MRDs, WordNet).

(7) KE systems are rare. The 'wit' system is one example of a KE as opposed to an IE system. Note, however, that even this one example does not appear to have been fully evaluated.

Throughout this thesis other systems are also mentioned and briefly described. These also bear out the above conclusions.

The examples of KE and IE systems described in this chapter have been taken from a wider set of such programs, but those chosen are representative of these fields. Having examined these examples and drawn the above conclusions, it is now possible to discuss the novel KE program being developed by the author of this thesis. In particular, the design rationale may be considered in the light of the above points. In Chapter 4, the KEP program is described. KEP aims to be shallow and NDS. It does this by looking for only specific types of information (see the fourth point above). A pattern matching approach is used (fifth point) which relies on part-of-speech tagging (sixth point). The rationale for KEP's approach is put forward in Chapter 4, where it is shown that for the intended application and given the available timescale, a shallow-processing approach which looks for specific types of knowledge is the route most likely to yield useful results. However, the above descriptions have thrown up a number of linguistic issues, such as the problems caused by anaphoric links between sentences and the nature of the text to be processed. Therefore before describing the KEP program in detail it is useful to explore linguistic issues which have affected the design and development of KEP. Chapter 3, which follows, introduces discourse-level concepts such as the nature of the text (is it explanatory, historical, fictional? etc) and the functions of sentences (informational vs. presentational). Categorisations such as the latter are important because in a shallow IE/KE system if one can filter out "the wrong type of sentence" then much fruitless processing may be avoided.

3. Linguistic Issues Relevant to KE

3.1 Introduction

In the previous chapter several real message understanding (MU), information extraction (IE) and knowledge extraction (KE) systems were described. For domain specific shallow systems in particular it became apparent that pattern matching techniques on a sentence-by-sentence basis are often used. It also became evident that there are features of natural language which may partially frustrate a sentence-by-sentence approach, whatever extraction method is proposed. One such feature was the existence of indirect anaphora relating to time context, as discussed with respect to the FASTUS system. Direct anaphora also posed a problem (e.g. finding the antecedent of a pronoun such as *he*).

It is the purpose of this chapter to introduce linguistic issues particularly relevant to shallow systems which process text a sentence at a time. This is the approach taken by KEP (described fully in the next chapter). By discussing linguistic issues which might affect KEP at this point, the need to parenthesize grammatical explanations in the following chapter will be avoided. Thus when these linguistic issues arise in the following chapter, the solutions adopted (if any) may be described without further preamble.

There are two main areas in which linguistic issues may affect shallow sentence-by-sentence KE: the first relates to the type of text to be processed, and the second to grammatical features which must be correctly handled for good IE/KE. To some extent these categorisations apply to all types of KE (DS and NDS, deep or shallow) but the discussions which follow are biassed towards shallow systems, be they DS or NDS, which essentially pick out individual sentences for processing (such as JASPER, FASTUS and KEP). The following discussion will cover only text types; it is assumed that the reader is familiar with grammatical constructions such as anaphora, ellipsis, apposition etc.

3.2 Types of Text

In this section the properties of texts as a whole which may be of concern to a KE system are considered. These properties relate to the suitability of individual texts for KE input. The properties are concerned with the purpose of the texts, the intended readerships, the style of writing etc.

3.2.1 Fictional vs. Non-Fictional Texts

The programs discussed in this thesis are not intended to process works of fiction. Although fictional texts may well contain facts, their main purpose is not to convey those facts to the reader - the facts are incidental to the plot. Fictional texts do indeed contain information regarding how language is used (lexical, grammatical, and stylistic information) but this is not relevant to the research reported upon in this thesis. Works of fiction will not therefore be considered further.

3.2.2 Explanatory vs. Historical Texts

Not all non-fictional texts are of interest to KE programs, although they might be of interest to IE systems. Non-fiction may be classified as historical non-fiction or explanatory non-fiction. The term expository text is also used, although it can also refer to the *setting out* of an argument, where the *purpose* of the text is not so much the conveyance of knowledge but the arguing for a particular viewpoint. Expository texts of this type may well use factual knowledge to bolster their case, even though the transferral of knowledge is not the prime aim (the prime aim being to persuade the reader to adopt a certain viewpoint). Despite these motivational differences, the terms explanatory text and expository text are often used interchangeably in the NLP community (see e.g. Hearst (1994), Tucker, Nirenburg and Raskin (1986)).

Historical texts, which are descriptions of chains of events, are not usually designed to convey individual facts to a reader or to introduce a topic. The medical reports processed by the MEDLEE system (see section 2.4.1.1) are historical reports - they are not intended to teach radiology to the reader. Although they are of interest to the MEDLEE system, which is an IE system, they would not be of interest to a KE system attempting to extract general medical facts. Thus for systems looking to extract facts as defined in the first chapter of this thesis, historical reports are not valid input texts. Newspaper reports and newswire stories likewise tend to be of more interest to IE systems. This topic has been discussed in the first chapter, in which *episodic knowledge* was introduced (Burkert (1995)).

Explanatory texts are on the other hand precisely those texts designed to convey knowledge to the reader. The purpose of a textbook on chemistry is to convey facts to the reader, in a structured way which builds up the reader's knowledge of the subject. Such texts are the prime inputs for KE systems.

There will, of course, always be texts which are difficult to place into one or other of the above categories. Technical reports may be a mixture of historical narrative and factual content. Indeed, this thesis probably falls into this mixed category. Such texts are still useful as KE inputs. However, the KE system will then be faced with the extra burden of deciding which parts of the text are fact-bearing.

3.2.3 Informational vs. Presentational Sections of Text

Not all sentences in a text are designed to convey information directly. Many sentences exist to smooth the flow of reading or point the reader to other parts of the text. Presentational sentences may start with phrases such as "The example given above..." and then go on to discuss the example rather than actually give an example of a concept. An informational sentence might take the form "An example of an X is the Y". For a shallow KE system which triggers likely sentences based on keyword searching, both sentences could be triggered (i.e. by the word *example*). However, the presentational sentence cannot give rise to an extracted example, and any subsequent processing done on it by the KE program is a

waste of effort. Worse still, it might result in a false extraction. Thus it is desirable for a KE system such as KEP to be able to detect, and hence ignore, presentational sentences.

3.2.4 Generic vs. Specific Sections of Text

It was stated above that historical texts are not generally of interest to KE systems. One of the main reasons for this is that sentences and larger units of text may refer to specific items rather than to general classes of items. Episodic knowledge about a specific dog Fido is of less interest than facts about dogs in general. Individual sentences in a text may refer to either specific or generic items and so for a KE system which works on a sentence-by-sentence basis it is desirable for the system to be able to distinguish between the two types of sentence. This topic has also been introduced in the first chapter.

3.2.5 Fact-Rich vs. Fact-Poor Texts

For NDS KE systems working on texts from widely varying topics it is likely that extractable facts are more numerous in certain domains than others. This may be the case irrespective of author style differences or medium variations. For example, in texts from some disciplines such as sociology it is unusual to find baldly stated facts such as "an increase in truancy causes an increase in crime". Instead, the entire text may argue for or against a view such as the above. The complex nature of human society is such that in this domain causal relationships are much more difficult to identify than in subjects such as computing. Thus one should expect "fact density" to vary with domain, and indeed this appears to be KEP's experience.

Clearly fact density is related to writing style. Although KE systems are not designed as style investigators, metrics such as the number of extracted facts per 100 words, as a function of domain, might prove of interest to stylistics researchers.

3.2.6 Declarative vs. Procedural Texts

Texts such as instruction manuals contain knowledge/information on *how to do* things. Skuce et al. (1985) distinguish this procedural knowledge from declarative (factual) knowledge. A single "piece" of procedural knowledge is likely to be spread over more than one sentence, since it may contain numbered steps etc. A piece of procedural knowledge may be regarded as an elaboration of a concept *how to do X*, where X is an action or a goal state, rather than as a definition of concept *how to do X*. To illustrate the difference, consider the concept *peeling an apple*. This concept is indeed an action. However, its elaboration would be something like: *take a knife, pick up the apple, and starting at the stalk cut the skin off in a single spiral.* The definition of *peeling an apple* on the other hand might be*: the action of removing the skin from an apple.* KEP is not designed to extract procedural knowledge. Although actions are valid target concepts, elaborations of such concepts in the above sense are not attempted by KEP.

However, there may be times when a KE/IE program is unable to distinguish between the two types of knowledge, so the designer of such systems must be aware of this issue.

3.2.7 Complex vs. Simple Texts

As was mentioned in the Introduction, a text is not just a set of standalone sentences (see Halliday and Hasan (1976)). The sentences in a coherent text are correctly ordered and refer to the other sentences or to concepts created by the body of text up to the current point. We may regard a text which has many such cross-links as a complex text. We may envisage many types of cross-links (connections, relations) at both syntactic and semantic levels. Taking just one example, an abundance of anaphoric links will clearly have an effect upon a sentence-by-sentence based KE system, especially if that system does not have the ability to de-reference such links. Even if the KE program is good at resolving anaphora, there may be facts which are not extractable because they are generated by an understanding of the text as a whole, an understanding which requires the identification of other types of connection.

3.2.8 Technical Text vs. General Text

Explanatory texts confined to a single domain which is specialist in nature invariably use specialist terms, or *technical terms* (TT). TTs are present in legal documents, medical texts, technical reports, scientific papers, trade journals, professional newspapers etc. The word "technical" does not imply "scientific" or "engineering"; it means specialised to the group of practitioners who need to communicate *within* their group concerning their specialism, whatever that might be. Thus estate agents, plumbers, patent agents, and astrophysicists all use TTs. The ability to detect TTs in texts is useful for KE systems, especially if those terms are the concepts being elucidated. For a NDS system, automatic detection of TTs is desirable, if not mandatory in practice (i.e. it removes the requirement to store many lists of TTs, costly to create and maintain). In section 4.6.4 the TT acquisition method used by KEP is described.

Such texts also often make use of abbreviations, which play the part of technical terms. Where an explanatory text uses acronyms then the full text for the acronym may be regarded as a useful fact to be extracted (e.g. "FSA stands for Finite State Automaton" is itself a useful fact). In section 4.6.5 the acronym acquisition method used by KEP is described.

General texts, whose target readership is not the closed class of a specialist group, are likely to use TTs and abbreviations less frequently. However, where a text has been written by a specialist for a general audience, it is more likely that any TTs and abbreviations that do occur are explained to the reader, at least on the first occurrence. Thus general texts also provide valuable input for KE systems. In fact, such texts may actually prove more valuable than highly specialised texts because they are aware of the need to teach in this way.

3.3 Conceptual Relations

This section introduces the knowledge-bearing constructs which are used by KEP to construct the explanations found in the third column of the glossary output. These are *conceptual relations*.

A KE system which attempts to use world knowledge to be non domain specific is currently not feasible on practical grounds - as evidenced by the efforts of Lenat in the CYC project (Lenat (1995a), Lenat (1995b), Lenat and Guha (1990)). A different approach must be taken to achieve generality (NDS-ness). Since factors external to the text are impractical, devices within text must be used. One way is to use knowledge about *how* information is encoded within a piece of writing. The study of textual discourse structure provides insights into such devices. Specifically, the structures variously known as *conceptual relations*, *coherence relations* (Hobbs (1979)), or *semantic relations* (Ahmad and Fulford (1992)) are available[6]. The preferred term in this thesis is **conceptual relation**, since the various instances all involve a concept which is elucidated in some fashion.

However, the term *coherence relation* also carries some weight since these textual structures tend to make a text coherent and, via anaphoric links and other mechanisms, cohesive. Morris and Hirst (1990) have pointed out that the terms *coherence* and *cohesion* are often (incorrectly) used interchangeably. Whereas cohesive links are all about the sticking-together of words in a text, coherence is concerned with relations among sentences and clauses, such as *elaboration*, *cause*, and *exemplification*. Furthermore, whereas cohesion lends itself to computational identification, there does not exist a general computationally feasible mechanism for identifying coherence relations. The KEP program represents an attempt to rectify this omission, albeit by looking for specific types of relation. Hoey (1991) also distinguishes coherence from cohesion; for Hoey, coherence is that (somewhat subjective) property of a text which arises due to cohesive links between sentences - a coherent text is one which makes sense as a text. Hoey argues that those texts containing many *bonds* between sentences (formed from cohesive links) are likely to be coherent.

Many researchers have attempted to categorise the various conceptual relations, or to examine a particular relation in some depth. For example, Vander Linden and Martin (1995) have made a study of the forms of the *purpose* relation as used in instruction manuals, from a Natural Language Generation (NLG) perspective. Ahmad and Fulford (1992) used both manual lookup and corpus-based methods to compile lists of lexical forms used in the *synonym*, *hyponym*, *partitive*, *causal* and *material* relations. Broad studies include that of Cruse (1986), who considers taxonomies (hyponymy), meronomies (parts and pieces), opposites (including antonyms) and synonyms. Lyons (1977) considers conceptual relations as part of a wider treatise on semantics, where the term *formulae* is used to describe lexical patterns used

[6] These terms are *used* by the references given, although may not have been coined by them.

to evince conceptual relations. Attempts have also been made to standardise terminology for particular applications, such as the British Standard Guide to the Establishment and Development of Monolingual Thesauri (published by the British Standards Institute, BSI) which considers hierarchical relationships (generic, whole-part, and instance), and the associative relationship (the *related term* relationship, for example as between "birds" and "ornithology").

Four major conceptual relations are definition, exemplification, partition and hyponymy. These are the four relations targeted by the KEP system described in the following chapter. It was not possible to test all conceivable relation types within the scope of the research reported upon in this thesis, and so it was necessary to decide upon a small set of important and interesting ones. The above four types were chosen out of all the possible relation types because they are commonly found in glossaries (this being determined by manual inspection of a small random sample of glossaries from technical reports, theses and textbooks).

Examples, part-whole descriptions and class inclusion statements are clearly useful pieces of knowledge which help a reader to understand a new concept, and definitions provide a statement of what a specific concept *is* or what it *means*. These four relation types seem more frequent and more fundamental than some of the others (appellation, causation, characterisation etc). In addition, they are somewhat different from each other in terms of their main purpose or emphasis, and so allow a KE approach to be tested for varying textual aspects. Definitions are given to tell the reader what is meant by a concept, examples seem to be given to aid in reaching an understanding of complex or subtle concepts, hypernyms place a concept into a tree-like categorisation scheme and hence allow the description of a new concept based upon differences from existing concepts, and partitions describe concepts as aggregates of components (which might already be familiar to the reader). Of the four chosen relations, one appears to be qualitatively different: definitions appear to be driven mainly by their purpose; they may in fact borrow the other three types in order to achieve this. Thus the set of four relation types chosen provides variety whilst being likely to cover the more important and most frequently found facts.

The author defines the four chosen relation types as follows (definitions in larger point size with explanatory text following each):

> A **definition** is a description of a concept in such a way and to such a depth that it presents the essential features of that concept.

Definitions play a crucial role in subjects requiring specific technical terms, such as engineering, mathematics and law. Note that the word "essential" within the above is being used in the sense of *captures the essence of*. Definitions have also been considered by other researchers, such as Swales

(1981), who states that the purpose of a definition is "to carry the reader from the known to the unknown either by explaining new terms or by re-defining old ones." Perhaps a more useful explanation is given by Skuce et al. (1985), who define a definition of a concept as the answer to the question *What is the meaning of <concept>?* This provides a useful working test for the presence of a definition in text. If the text is considered to correctly answer the question *What is the meaning of <concept>?*, then it can be considered to contain a definition of *<concept>*. It is important to be able to identify definitions in text in some standard way because the recall metric requires it (see Chapter 5).

Clearly, even with a test such as the above, there will be an element of subjectivity involved in the manual identification of definitions. The combination of the author's definition and Skuce et al.'s definition, which are conceptually very similar, should reduce this subjectivity. The usefulness of this approach will be illustrated in a later section. Note however that it would successfully have identified a definition of an action concept such as *peeling an apple* (i.e. *What is the meaning of peeling an apple?* is answered not by a list of steps required, but by some statement such as *the action of removing the outer skin of an apple*).

Definitions have also been considered in great depth by Flowerdew, and no discussion of this relation would be complete without mentioning this work (see, for example, Flowerdew (1991), Flowerdew (1992a), Flowerdew (1992b)). However, the focus of the Flowerdew studies is spoken text (specifically, science lectures) and so much of the discussion is not relevant to the work reported upon in this thesis, since it covers features not often used in written text, such as *repetition*, and also paralinguistic features (e.g. emphatic stress, hand-waving, body language etc). However, much of it might indeed prove useful, especially for the *detection* of a definition prior to extraction. For example, in Flowerdew (1992b) the use of *grounders* is considered, these being sentences or phrases preceding a definition in order to pre-introduce the concept about to be defined. For example, a speaker/writer might say *Let us now turn to X. An X is <definition>*. KEP does not currently search for clues as to whether a definition is imminent; clearly this would be a useful feature in an improved triggering function (i.e. the function which signals the possible presence of a definition in a sentence; the forthcoming section 4.6.7 discusses the current triggering method). Such work is outside the scope of the present research, although it is indeed in the author's long-term plan to utilise it.

> An **exemplification** is an elaboration of a concept using a specific instance (or instances) of that concept.

Examples are a valuable tool in instructional text and their roles in pedagogical applications have been much studied. Almost all explanatory texts make use of exemplifications, this thesis itself being no exception.

Mittal and Paris (1993) attempt to categorize *exemplifications* as found in instructional texts. The authors start by describing the example categorisations of Polya and of Michener, which are as follows. Polya (1945) gave three categories (i) leading examples, (ii) suggestive examples, (iii) counter examples. Michener (1978) gave five categories (i) introductory examples, (ii) model examples, (iii) reference examples, (iv) counter examples, (v) anomalous examples. These schemes are criticised by Mittal and Paris on the grounds that because the *context* of the example is not looked at, the same example can be categorised differently in different contexts. Also, because the categories are somewhat fuzzy, it is difficult to implement a computational system to pick them out.

The method preferred by Mittal and Paris is based on the well-supported hypothesis that context is important when assessing the effectiveness of exemplification in instructional texts. To this end, three *dimensions* are used:

- the relationship between the information in the example and that in the context
- the intended audience of the example
- the knowledge type being communicated by the example

The first dimension is categorised three ways. *Positive examples* are instances of the concept being described in the accompanying text (context) and they satisfy all the properties of the concept as described. Such examples play an *elaborative* role. *Negative examples* are counter-examples and as such are deliberate non-instances of the concept being described; they play a *contrastive* role. Negative examples allow *necessary* concept features to be highlighted (the so-called *critical features.*) Finally, *anomalous examples* are irregular or exceptional cases which are either positive instances of the concept even though the accompanying description of the concept does not directly imply this, or negative instances which readers often misclassify as positive (and hence which require highlighting). Such anomalous examples will always require appropriate explanatory text, and are often presented apart from the other two types (e.g. towards the end of the discussion).

The second dimension concerns the intended readership of the text. Mittal and Paris suggest three levels of example as follows. *Introductory* examples are aimed at novices who want to *learn about the concept. Intermediate* examples are aimed at users with moderate previous exposure and who want to *learn to make use of the concept. Advanced* examples are aimed at users with extensive knowledge who want to *have some point about the concept clarified.* It is worth pointing out that KEP is designed only to consider the first of these levels. The latter two categories imply the updating of pre-existing nodes in the knowledge base, rather than the creation of new nodes.

The third dimension is the knowledge type. Here, Mittal and Paris categorise the concept being exemplified into one of the three types *concepts*, *relations* or *processes*. By *concept* they mean "thing" e.g. "a list". By *relation* they mean something which relates one thing to another, such as the input of a

function to its output. *Processes* are chains of events. Thus processes correspond to the procedures of Skuce et al. (1985) discussed earlier.

Mittal and Paris state that they have used their example categorisation scheme in a planning system that suggests the required places and types of examples needed in a text. The input is a top-level goal such as *describe "list"*. The output is a plan of the tutorial required. This output is in the form of a "discourse tree", which is a hierarchical representation of the sub-goals within the plan, the leaves being primitive statements such as INFORM... i.e. which could be turned into English language statements. It is claimed that a "grammar interface" converts the discourse tree into a form suitable for input to PENMAN, an NLG system.

> A **partition** is an elaboration of a concept using a list (or partial list) of its component parts.

The partition relation is signalled by keyphrases such as *is made up of three parts*, *comprises*, *has the following components* etc. It is not to be confused with the material relation, which describes the substance from which a concept (usually a physical object) is made. The partition relation has been investigated widely largely because of its informational (rather than presentational) content. It appears to be one of the fundamental semantic relations used within text to say something about how things are in the real world. It is an ideal starting point for a KE program because it presents *facts*. Phrases such as (1a) to (1f) below demonstrate this.

(1a) The handlebars are part of a bicycle.
(1b) A computer has three components: a monitor, a keyboard, and a CPU unit.
(1c) A jug has a handle and a spout.
(1d) Germany is part of the EC.
(1e) A suit comprises jacket and trousers.
(1f) A tree is part of a forest.

Note that these examples show various lexical forms available to the partition relation. They also reveal sub-divisions within the partition relation relating to the nature of the object being partitioned and the manner of that partition. In order to detect instances of the partition relation within text it is necessary to define precisely what is meant by this term, and what is to be excluded.

A detailed study of the taxonomy of the partition relation has been made by Winston, Chaffin and Herrmann (1987). The taxonomy describes six types of partition relation, as follows:

Component/Integral Object An object is characterised by its components, as in (1a), (1b), (1c) and (1e) above. The components are functional parts of the object, can in theory be separated from it, but are not homeomerous i.e. they are not similar to one another and to the whole object. Thus (1d) also falls into this category. This is the type of partition relation which most readily springs to mind. Note that the wholes do not have to be physical objects, and that their components are not the same thing as *pieces* of the whole (a broken jug's pieces are not the same as its components, as in (1c).)

Member/Collection Members of a collection are indeed parts of it but do not perform a particular function or possess a particular temporo-spatial arrangement with respect to it. Sentence (1f) provides an example. Winston, Chaffin and Herrmann make the point that classes differ from collections in that membership in a class is determined on the basis of similarities to other members, but collection membership is determined by spatial proximity or social connection. The special term *groups* is used when social connection is the binding factor. Note that this type of partition is not the same as the Member/Set relation. A *set* is not the same thing as a collection; sets have names which reflect their membership, such as *the set of all positive integers*. Here, we could not say that 198 *is part of* the set of all positive integers. A set is essentially a collection of individual items, not a whole thing, such as a forest.

Portion/Mass A portion of a pie is a part of that pie, but it is not a component of it. In this type the parts are homeomerous (the piece of pie is "pie", and so is the whole), non-functional, and separable. The lexical form "some of" indicates this partition type if it may replace "part of" without altering the meaning. This is the mass sense of "some of", but since this phrase also has a count sense (as in *some of my friends are boring*) it cannot be relied upon to exclusively indicate the portion/mass relation.

Stuff/Object Winston, Chaffin and Herrmann argue that in a sentence such as *A martini is partly alcohol* the alcohol is to be regarded as a part of the whole drink, as evidenced by the term "is partly". Clearly this is a troublesome area because they also discuss the need to use "is made of" when the whole object is composed of the same substance, as in *A lens is made of glass*. The author prefers to regard both these examples as belonging to the *material* relation. In the first sentence, the term "is partly" has the intended meaning "is partly made of", where the emphasis is actually on the elided "made of". Thus the main intention of the sentence is not to list one or more of the *parts* of a martini, but to state that a martini is (partly) composed of a particular *material* (alcohol). It is suggested that the parts of a martini are vermouth, gin, ice etc, i.e. its ingredients. However, one might argue that this still comes under the material relation, and so clearly this is an area open to debate.

Feature/Activity Here the whole is an activity, such as shopping, and its part a stage (feature) of that activity. Thus *Paying is part of shopping* contains meronymic information. The components in this type are functional, but not homeomerous or separable. Interestingly, this is the only relation type in the taxonomy which cannot be expressed in the form *X has Y*. (Bicycles have handlebars, forests have trees, pies have slices, ?shopping has paying). Since we are ultimately interested in partition relation KE from texts, forbidden forms such as these are of importance.

Place/Area This is the relation between areas and special places within them. For example, *The Everglades are part of Florida.* This type appears to be similar to the portion/mass type, but here the part is not in theory separable from the whole. Parts are, however, homeomerous (the Everglades are "Florida", as is Florida).

Winston et al. use the above taxonomy to explain apparent failures in transitivity within syllogisms, such as in the following triplet:

(2a) Simpson's arm is part of Simpson
(2b) Simpson is part of the Philosophy department
(2c) Simpson's arm is part of the Philosophy department

Sentence (2c) apparently follows logically from (2a) and (2b), although it is obviously odd. This is explained by the recognition that in (2b) the semantic relation is in fact that of meronymic member/collection, whereas in (2a) it is meronymic component/integral object. Sentence (2c) is thus invalid due to the mixing of different semantic relations in the preceding assertions. This becomes clearer if "part of" in (2b) is replaced by "a member of". Thus it is the *vagueness* of the phrase "part of" which allows the apparent failure of transitivity. To put it another way, the phrase "part of" can be used to indicate more than one type of partition, if not other non-meronymic relations as well.

Further discussions relating to these issues are presented in the paper Bowden, Evett and Halstead (in preparation).

A **hypernym** is a categorisation of a concept achieved by stating the parent class of that concept.

The hypernym relation is signalled by phrases such as *is a type of* e.g. *a dog is a type of mammal.* The hypernym relation is the other facet of the **hyponym** relation, since hypernyms and hyponyms exist as pairs. In the hyponym relation, the concept is the hypernym (parent class) and the elaboration is a member of that class; this relation is signalled by phrases such as *include* e.g. *mammals include humans.*

The hyponym relation, also called **class inclusion**, is a natural link type for semantic net knowledge bases. Hypernym/hyponym pairs (such as mammal/dog, computer/mainframe etc) allow the construction of a hierarchy of concept nodes. The most commonly used link type for this relation is *is a type of*, where the link goes from the hyponym (lower class) to the hypernym (higher class). However, the link wording *is a* is also often used, e.g. as in WordNet (Miller et al. (1990)). Thus *a dog is a type of mammal*, or *a dog is a mammal*. The relation should not be confused with the **instance** relation, which attaches a terminal node to a category, such as in *Fido is an instance of a dog*. (Confusingly, *is a* is sometimes used for this relation too.)

The hyponym relation underpins the currently fashionable object-oriented (OO) programming philosophy and it has been extensively studied for this reason alone. Within the KE arena the hyponym relation has been the target of attempts to automate hierarchy creation from dictionaries (e.g. by Chodorow (1985), Markowitz, Ahlswede and Evens (1986), Alshawi (1987), Zhu and Shadbolt (1995)), and from other texts (e.g. Hearst (1992)). Hyponym extraction from MRDs has been found to be especially tractable, due in part to the nature of dictionary entries (i.e. they are definitional by nature, and easy to find because they are not surrounded by much irrelevant text) and also because of the rigidly constrained syntax used (e.g. nouns are explicitly marked as such). For example, Chodorow (1985) claims a 98% accuracy rate on extractions from Websters 7th Collegiate Dictionary, using the simple heuristic of looking for the head of a defining noun phrase.

Hyponym KE from running text is not so straightforward, but has already been suggested using pattern-matching techniques. Hearst (1992) claims that hyponym lexico-syntactic patterns are easily recognisable, occur across text genres, and "indisputably indicate the relation of interest". Hearst also states that "for finding simple semantic relations, pattern recognition is far more accurate and efficient than parsing", and goes on to suggest an algorithm for discovering the patterns. The idea put forward is to use already-known hypernym-hyponym pairs (such as Country-England) to discover new lexico-syntactic patterns used to hold hyponymic knowledge. This method would require large amounts of pre-existing knowledge accessible to the current run of a KE program, and very large amounts of textual input. It is not the method used by KEP.

Note that for all four relation types described above, the relation has been named after what is extracted for the concept, rather than for the concept itself. Thus in the case of the hypernym relation, the concept being elucidated is the hyponym (e.g. *dog*), and the elucidation (extracted solution) its hypernym (e.g. *mammal*). Thus although the concept to be elucidated may be the grammatical subject of the sentence (*a dog is a type of mammal*), the conceptual relation is named after the relation of the extracted part *to* the concept (mammal is the hypernym of dog). This convention is arbitrary and not all authors stick to it, but it seems logical given that it results in consistent concept/elucidation pairs:

DEFINITION RELATION = concept + its definition (a definition of the concept)
EXEMPLIFICATION RELATION = concept + its exemplification (example(s) of the concept)
PARTITION RELATION = concept + its partition (what the parts of the concept are)
MATERIAL RELATION = concept + its material (what the concept is made of)
HYPERNYM RELATION = concept + its hypernym (what the parent class of the concept is)
HYPONYM RELATION = concept + its hyponym (member(s) of the concept's class)
INSTANCE RELATION = concept + its instance (instance(s) of the concept's class)
CAUSATION RELATION = concept + its cause (what caused the concept usually an action or a state)
NOMINATION RELATION = concept + its name (what it is called) (Also called APELLATION)
CHARACTERISATION RELATION = concept + its characteristics (properties)
etc.

The concept need not be the grammatical subject of the sentence; the relations above are concerned with thematic rather than grammatical roles. For this reason relation types such as the above may overlap each other. The hyponym relation usually gives a *class* of items which exists within the concept's class (e.g. Dogs *(concept)* include hunting dogs *(hyponym)*), whereas the instance relation gives a named object in the concept's class (e.g. Dogs *(concept)* include Fido *(instance)*). The exemplification relation is similar to the instance relation, and in many cases it may be difficult to distinguish the two. For example, in *PASCAL is a high-level language*, is the relation an exemplification (of *high-level language*) or an instance? Or both? Or is it a definition of the *PASCAL* concept? To answer these questions it may be necessary to examine the surrounding text to find the motivation for the statement. The definition relation may also have some overlap with other relations, since the definition of the concept may actually be made using a hypernym. Thus for an essentially arbitrary and open-ended set of conceptual relations, deciding whether a given relation is present in the text, or deciding whether it is the sole relation present, are important issues. Are there in fact any trigger phrases which unambiguously confirm the presence of a given relation in text? This question is returned to in later discussions.

4. The Knowledge Extraction Program (KEP)

4.1 Introduction

In this chapter the novel KE system developed by the author is described. KEP (Knowledge Extraction Program) aims to be a non domain specific system which can extract factual knowledge from explanatory texts. Possible applications have been mentioned in section 1.4 and are returned to in section 6.3. An earlier version of the KEP system has also been described in Bowden, Halstead and Rose (1996a) and (1996b).

4.2 Avoiding Deep Processing

4.2.1 Motivation for a Shallow Processing Approach

The deep processing programs described in Chapter 2 demonstrate clearly the difficulty of attempting NDS KE using a deep processing approach. Most of the deep approach examples are domain specific. The only deep processing NDS systems described earlier are Conceptual Dependency and Preference Semantics. The former is a huge suite of software developed over several years by several people, and the latter is hardly less broad in scope. Since the NDS approach is desired for KEP, a shallow processing approach is the only practical route given the resources involved in this research. However, practical considerations are not the sole factors here. One of the major motivating factors for this research was the desire to discover to what extent a shallow technique could be used for the KE task, and in particular for the *NDS* KE task. This theme is expanded upon in a forthcoming discussion section, but it has already been discussed in the first chapter. The idea is to reveal how far a shallow approach could go whilst at the same time creating a useful system, if indeed the latter turns out to be possible. Along the way, it is hoped that some interesting *linguistic* discoveries might be made.

4.2.2 Pattern Matching for a Shallow Approach

If full syntactic/semantic parsing is not to be performed, what are the alternatives? Examination of the examples of systems given previously shows that there is really only one approach - pattern matching. This is a broad term and can cover everything from looking for certain exact phrases (very shallow), through using pattern templates of parts of speech, to patterns of functional parts (subject, object etc). For domain specific systems, such as SCISOR, the elements in the pattern may be at a high-level semantically, including items such as dates, companies, specific events, people's roles etc. Thus at the highest level a pattern might be something like **<company take-over company>**. Actually matching this pattern to a piece of text involves linguistic knowledge (e.g. so that *took over* is recognised) as well as world knowledge (e.g. so that *Acme Widgets* is recognised as a company). Thus the term *pattern*

matching in reality covers a whole set of techniques which operate at different levels syntactically and semantically. The pattern matching approach used by KEP is described in section 4.6.10.

4.3 Avoiding Domain Specificity

4.3.1 Motivation for NDS system

Why attempt to create a non domain specific system? The first answer is so that the program can be used in many domains without the need to spend a lot of time and effort compiling domain knowledge. Such knowledge is not likely to be static, especially when the domain is a fast-moving field such as computing. Thus there is not only an initial set-up effort required; any KB will require continuous updating and maintenance. This problem has been recognised by researchers in various fields, such as those trying to devise systems aimed at extracting technical terms from documents (see e.g. Justeson and Katz (1995)). It is of course true that domain knowledge must be compiled and maintained for a DS system too. But the difference is one of scale; in practical terms the KB needed for a DS system may be finite, whereas that needed for the NDS system may be very many times larger, if not open-ended in scale.

A second reason for attempting an NDS system is that it may tell us something about the nature of language, particularly about the ways in which English is used to hold knowledge. Such knowledge may be interesting in its own right, but it may also have practical applications. The corpus studies described later in this document revealed systematic data concerning the ways in which definitions, exemplifications, partitions, hypernyms etc are structured in English, and hence allowed moderately successful methods for automatic extraction. Other researchers have already performed similar corpus studies, motivated by various applications. For example, for TEFL (Teaching English as a Foreign Language), Xuelan and Kennedy (1992) report on ways of expressing *causation* in English, their motivation being the desire to ease the learning of ways of expressing this concept in English for second-language students (see section 6.2.11).

Thirdly, NDS systems may tell us something about the way to create systems that learn. This is an important research area which has frequently arisen within NLP (see the review paper Collier (1994)). Learning is more important in NDS systems than in DS systems because with the former it is often impossible to *manually* encode all the necessary world knowledge at the outset, thus creating the requirement for automatic learning.

The above is all very well, but what is it that makes the author of this thesis believe that an NDS system is indeed possible? This question is especially important because the system is also to be shallow, and (as has been pointed out in the Concluding Remarks of the second chapter) therefore represents a rare combination of these two dimensions. The answer lies in the nature of the communication channel

chosen i.e. in the narrower task attempted by KEP. Instead of attempting to extract *all* knowledge, something which would undoubtedly require large amounts of WK, only certain specific types of knowledge are to be uncovered. The approach is to use knowledge types which occur in all explanatory texts, whatever their subject areas. Furthermore, if such knowledge types are often present in the same (finite) set of lexical patterns, then a shallow pattern-matching approach may be used to extract the knowledge. The knowledge types referred to are *conceptual relations*; these were introduced in section 3.3.

4.4 Output formats

This section discusses the appearance of KEP output, and demonstrates what KEP is capable of. The description which follows uses a test text created solely for this purpose; this test text was not part of the formal evaluation of KEP. The test text is given as Figure 8. The advantage of describing the output formats at this stage is that it gives the goal towards which we are aiming. By giving a 'black box' description of the KEP system, i.e. by describing its input and resultant output, the reader will appreciate the nature of the attempted task at the outset. This then allows a top-down description of how that goal is achieved to follow in later sections.

Sorting is the action of arranging data items into some specific order. We can define a sort routine (SR) to be a function which orders a list of items according to some criterion. Examples of SRs include the bubble sort and the quick sort. Sort routines are composed of four elements: input list, output list, sort criterion and sort algorithm. An example of a sort criterion is alphabetical order. A sort routine is a type of data rearrangement algorithm, or DRA. In these, data elements are not themselves altered, but their order of presentation is changed to assist the calling application.

Figure 8. Example of text containing 4 conceptual relations

There are five standard output formats for KEP:

- long format (a file containing a copy of the text read in, the sentence structure of that text, and details of the extraction process as it proceeded, together with the extractions themselves and some counts);
- short format (a file containing only the extractions, of which Figure 9 is an example);
- KEN format (Knowledge Extraction Network, of which Figure 10 is an example);
- glossary format (of which Figure 11 is an example);

- term summaries output (of which Figure 12 is an example).

Figure 9, Figure 10, Figure 11 and Figure 12 correspond to the input text of Figure 8. Whereas the first four bulleted output formats above are essentially the same extractions formatted in alternative ways, the fifth (Figure 12) is different in content - it is a list of technical terms discovered in the document, together with blocks of source text containing those TTs. (The long output file, first bullet point, is too long to include as a figure here and therefore has been included as Appendix D.) The reasons for choosing the output formats given below are given later in this thesis.

```
Concept: sort routine
Definition: a function which orders a list of items according to some criterion
Hypernym: data rearrangement algorithm, or DRA
Example: bubble sort
Example: quick sort
Part: input list
Part: output list
Part: sort criterion
Part: sort algorithm

Concept: sort criterion
Example: alphabetical order
```

Figure 9. Example short KEP output

```
C:sort routine]
{has an acronym}
C:SR]
{has a definition}
C:a function which orders a list of items according to some criterion]
{has an example}
C:quick sort]
{has a part}
C:input list]
{has a part}
C:output list]
{has a part}
C:sort criterion]
{has a part}
C:sort algorithm]
C:sort criterion]
{has an example}
C:alphabetical order]
C: data rearrangement algorithm]
{has an acronym}
C:DRA]
```

Figure 10. Example KEN output

ACRONYM	TERM	EXPLANATION
DRA	data rearrangement algorithm	
	sort criterion	Examples: alphabetical order
SR	sort routine	Definition: a function which orders a list of items according to some criterion. Type of: data rearrangement algorithm, or DRA. Examples: bubble sort and quick sort. Parts: input list, output list, sort criterion and sort algorithm. SEE ALSO sort criterion, data rearrangement algorithm

Figure 11. Example Glossary output

SR	sort routine
	1 We can define a sort routine (SR) to be a function which orders a list of items according to some criterion . 2 Examples of SRs include the bubble sort and the quick sort .
DRA	data rearrangement algorithm
	5 A sort routine is a type of data rearrangement algorithm , or DRA .
	sort criterion
	3 Sort routines are composed of four elements : input list , output list , sort criterion and sort algorithm . 4 An example of a sort criterion is alphabetical order .

Figure 12. Example Term Summaries output

Output may be displayed on screen (piped into the UNIX *more* facility) and/or printed at a laser printer. This is under user control. KEP does not at present provide a graphical output. Figure 22 on page 206 gives an example of such an output (essentially from the above short output, except that the extra *sorting*

node is shown). Note that a graphical output shows explicitly the links between concepts implicit in the short output. For example, in the short output the concept *sort criterion* is linked to the concept *sort routine* since the former is a part of the latter; these relationships are not always easy to spot in a textual output file.

Although the glossary produced is correct as far as it goes, it is worth noting what does not appear in the above output. There are two obvious omissions which a human glossary maker might have provided: (1) there is no glossary entry for *sorting*, and hence no accompanying definition of this concept (first sentence), (2) there is no explanation of a DRA based upon the last sentence in the text. Questionably, a human might also have provided an entry for *data item*, realising that *data item* (first sentence) and *data element* (last sentence) are semantically equivalent in this text. The reasons for these omissions will become clear as the KEP program's *modus operandi* is described in the following sections.

4.5 KE Strategy: An Overview

Before detailing the processing used in KEP, it is useful to outline the major steps. The following paragraph is the outline of processing performed for a full run of KEP started using menu choice 6 (see Table 5 on page 93). The outline need not be read at this point; the reader may return to it after having read the rest of the chapter, or may wish to read it both before and after the full description. Each of the steps in the outline is described in detail in the following sections. The outline itself contains no justification for each step (i.e. why each step is necessary) – it is merely a statement of what KEP does in what order. The need for each step and the rationale for the order of processing are developed in the following sections, but much of the rationale has already been described in earlier sections, where the reasons for choosing a glossary were discussed (section 1.4.5) and the necessary elements of a glossary considered (section 1.1.3.7). To a large extent these determine the required processing steps, although not of course how these are achieved.

OUTLINE OF KEP PROCESSING After a file of text has been part-of-speech tagged and pre-processed to bring it into a common format, it is read in by KEP which copies it into memory and obtains the sentence structure. Each sentence is then examined for technical terms (TT), a process which involves looking ahead to future sentences. Next, each sentence is examined for acronyms and what they stand for and, where acronyms stand for phrases which are TTs, reference links are made between the TT and the acronym. For each of the four conceptual relation types, triggering is performed to find the "interesting" sentences. Presentational sentences are then rejected. A novel pattern matching technique is then brought to bear on these sentences, to obtain fragments of text which hold knowledge. Fragments are validated to confirm that the pattern match was a useful one. Validated fragments are then presented as useful extractions. The various output formats are then constructed, including the main output format – a glossary for the input text.

These stages will now be considered in some detail. An overview of processing is also given in the architecture diagram (Figure 13), in which arrows represent text flows and boxes represent processes performed on these.

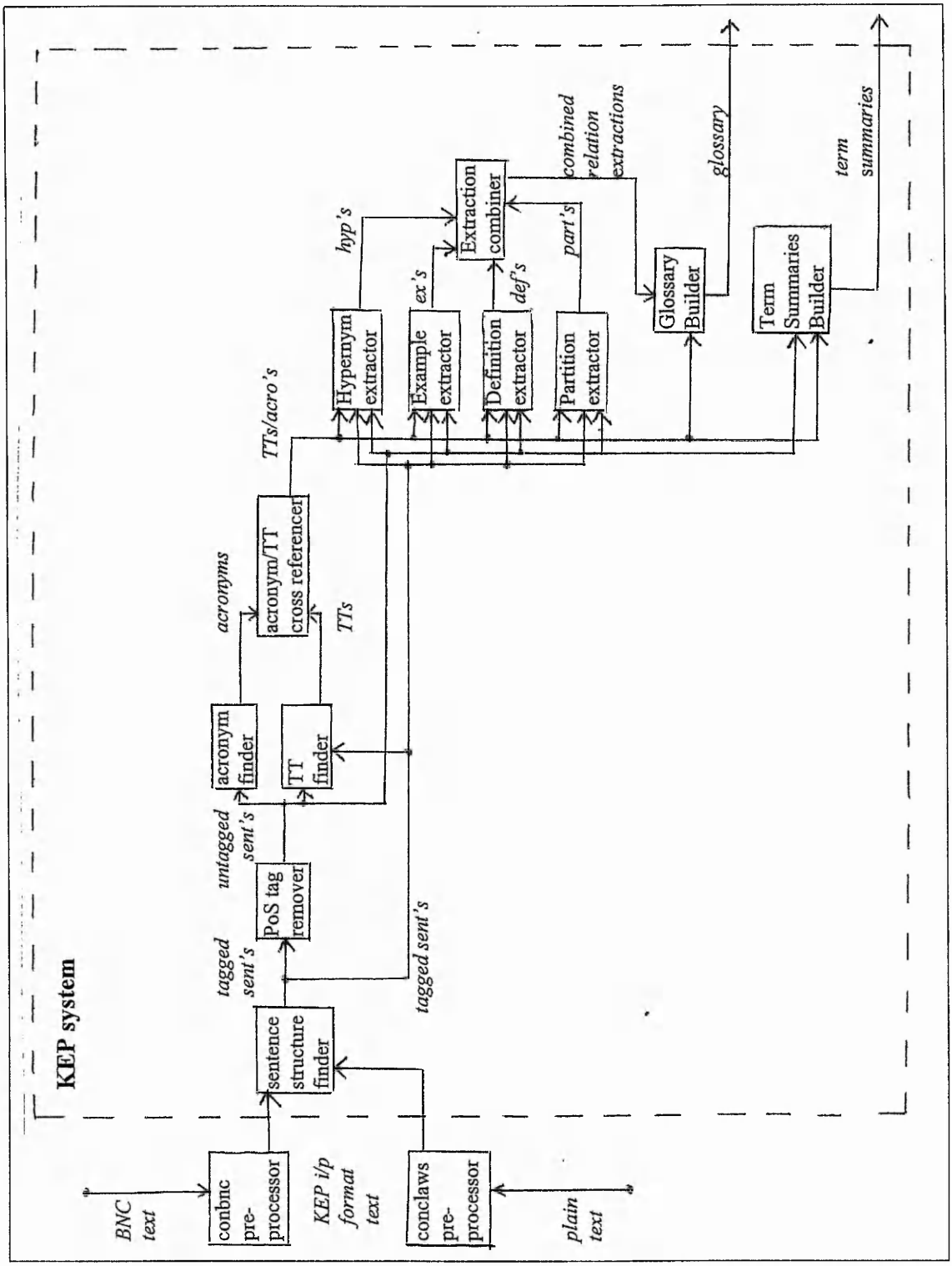

Figure 13. KEP System Architecture

4.6 KEP Processing

4.6.1 Pre-KEP Text Processing

4.6.1.1 Part of Speech Tagging

The KEP program relies upon its input being part-of-speech tagged prior to its being read in. Since this process is not carried out by any program designed by the author, it is worth discussing the nature of the process and why it is needed.

A part of speech tagger, usually referred to simply as a tagger, is a program which accepts a text and returns that text with each word 'tagged' with a part of speech code. Consider again Figure 1 on page 24. If the leaf nodes of the tree are attached to the terminal symbols using an underscore character, a tagged sentence results:

Fierce_ADJ dogs_NN attack_V ferociously_ADV

However, it is important to realise that a tagger is not a parser. Although the output from a tagger looks like the bottom line of a parse tree, the upper parts of the tree are absent, because they were never produced. Taggers work by looking up words in a lexicon and finding the possible parts of speech for each word. Each word may have more than one potential tag. For example, **dogs** in the lexicon may be a noun or a verb. The task of the tagger is therefore to select the correct tag out of the set of tags available for each word. Perhaps surprisingly, this process does not require a full parse of the sentence, because statistical methods may be employed to select the most likely tag for a particular word, given the tags of the words near to it. Using such methods most successful taggers are able to select correct tags for more than 90% of all the words in a text.

For example, the CLAWS tagger (Constituent Likelihood Automatic Word tagging System) takes the above approach. See Garside, Leech and Sampson (1987) for a detailed description of the original program, and Leech, Garside and Bryant (1996) for an overview of the latest incarnation (CLAWS4), which was used to tag the BNC. The heart of the tagger uses a Hidden Markov Model (see e.g. Charniak (1996) for an explanation of HMMs) for the assignment and disambiguation of tags, but CLAWS also contains a rule-driven contextual part-of-speech assignment section, designed to tag idioms (such as *he kicked the bucket*) i.e. parts of speech which extend over more than one orthographic word. (Thus if *kicked the bucket* is being used in the sense of *died*, then each of the three words *kicked*, *the*, and *bucket* should be tagged in such a way as to indicate that they are to be treated as a single entity.) This part of CLAWS4 is becoming more important, since there are general idioms such as *as much as*, which might be regarded as a single coordinator, and complex names such as *Dodge City* and *Mrs. Charlotte Green.*

Also handled are foreign expressions such as *annus horribilis*. Separate lexicons are used to handle these types, and they now hold over 3,000 entries *in toto*.

The CLAWS4 tagger has an error rate of about 1.5% (i.e. words incorrectly tagged) and leaves a further 3.3% of words ambiguously tagged (i.e. CLAWS4 is unable to decide upon a tag, and gives so-called *portmanteau* taggings i.e. lists of possible tags). However, these figures are averages and many factors should be examined when discussing error rates. For example, when tagging the word *horrifying* in *a horrifying adventure*, should it be labelled as an adjective or a verb particle? Clearly it is important to measure error rates against a published standard which details such cases, i.e. an annotation scheme. This is currently being produced for CLAWS4. Surprisingly, the size of the tagset does not seem to affect the error rate. Two tagsets were used for the BNC tagging: the C5 tagset of 58 tags for the whole corpus, and the C6 tagset of 138 tags used for a 'core corpus' part of the BNC.

Part of speech tags are used by KEP mainly for technical term acquisition (see section 4.6.4). Future enhancements may also utilise them for text fragment validation and pseudo-parsing.

The list of part of speech codes which a tagger uses (the *tagset*) normally contains several tens of different tags. Although the common grammatical categories exist within all taggers (singular noun, plural noun etc) there is not always a one-to-one correspondence between two different tagsets, because some categories may be subdivided in one tagset, absent in another etc. For this reason, automatic translation between tagsets is not easy. Thus KEP pattern files (see Table 6) must be created largely manually if a new tagger is used. The paper Qiao (1995) demonstrates the sorts of problems which need to be considered if automatic tagset translation is desired; here the mapping is actually between parsed corpora rather than merely tagged corpora (the Lancaster Parsed Corpus and the Susanne Corpus) but the requirement includes within it tagset mapping. Although the approach may be said to be applicable to any two tagsets, much of the detailed work is specific to the two annotation schemes involved. Thus whenever a new tagging scheme is encountered, a new translation program would be needed for every other existing tagging scheme. The upshot of this is that in order to re-tag a text one would not normally run a translator program, but simply re-tag using the new tagger. This is fine, but would not solve KEP's pattern file translation problem. In the work reported upon here this problem has been largely sidestepped by using just the one tagger and its corresponding tagset(s) - the CLAWS system as described above.

CLAWS may now be purchased for installation on a local computer. Thus any text file may be pre-processed to provide tagged input for KEP. In addition, existing BNC files come pre-tagged (by CLAWS, see above), although the format (layout) of these files requires pre-processing before KEP can use them.

Before leaving the subject of taggers, it is worth justifying their use in a non domain specific system. The following paragraphs discuss the issue of domain specificity and give examples of other NDS systems which use part-of-speech taggers. These arguments are used to justify the use of the CLAWS tagger by KEP as a pre-processor program.

All British English part-of-speech taggers have been designed to work on any (British English) text, whatever its subject matter or style. Thus taggers are inherently non domain specific NLP programs. Taggers do of course use internal lexicons, but these too are not domain specific. Since many technical terms are made from everyday words (e.g. *chain reaction*) they are properly tagged even within domain-specific texts. Furthermore, most taggers attempt to tag unknown words, with a high degree of success. In addition to being NDS, the tagging process does not involve human intervention and so a tagger is an automatic system. Thus a tagger is an *automatic*, *NDS* program. This justifies the use of a tagger as a first stage of a system which claims to be both fully automatic and NDS and which takes plain text as its input.

Other researchers have also used part of speech taggers in their NLP systems. For example, Zernik and Jacobs (1990) used pre-tagged text to discover thematic relations (i.e. the roles played by elements of text, such as *actor* or *recipient*). Word co-occurrence data was collected from a corpus of business articles, tagged by an in-house developed tagger (un-named). The word co-occurrence counts together with the word tags were used to discover facts such as that *shareholders* are the recipients of *pay*, whereas *dividends* are the objects. This is a good achievement for a system which does not use world knowledge, and shows the degree to which syntactical information may aid in KE. Manning (1993) used tagged text to discover verb subcategorisation information automatically from text. Subcategorisation information includes factors such as whether the verb is transitive, intransitive, stative etc, and whether certain prepositions may or must follow the verb. Tagged text is therefore required in order to identify the verbs and the other parts of speech following them, the verb and its associated text being processed by a finite state parser. Manning found the 5% error rate of the tagger used (a version of Kupiec's stochastic tagger, Kupiec (1992)) to be acceptable for his application. Thus taggers are beginning to be used as knowledge-discovery tools, both for purely syntactic knowledge (Manning) and higher-level knowledge including thematic WK (Zernik and Jacobs).

4.6.1.2 Pre-processing Programs

A tagging format is a description not of the tags used in a particular tagset, but of how those tags are attached to words in the text. In order to avoid the need to add modules to KEP to handle each of the tagging formats which might be encountered, KEP accepts only one tagging format - part of speech tags attached to word endings via a tag-attacher character or characters. KEP uses pre-processor programs to provide this. Corpora may use pre-word tags (e.g. BNC <w> tags) and usually contain other mark-up devices, which need to be stripped. In addition, some corpora (e.g. LOB) arrive in vertical format i.e.

each word (together with tagging and other information) is on a separate line. Currently, the pre-processor programs detailed in Table 4 have been written. By default, their output is placed in the file **kep.in**, which is the KEP default input file. Note that the third of these, conclaws.c, is designed to handle tagged texts which were originally input as plain text to the CLAWS4 tagger, rather than pre-existing corpus texts. It thus allows any text to be processable by KEP, e.g. texts created by word processor and saved as plain ASCII text, or other pre-existing ASCII documents. The conclaws program contains within it a mapping function from the CLAWS4 C7 tagset to the simpler C5 tagset used by much of the BNC (and by KEP).

Program Name	Source Converted
conlob.c	vertical-format LOB files
conbnc.c	BNC files (CLAWS C5 tagset, pre-word <>-tags as input)
conclaws.c	vertical-format CLAWS '.c7' files (C7 tagset or C5 tagset output)

Table 4. KEP preprocessor programs

It might be considered wasteful to strip out useful annotation information from a tagged text, such as a BNC text. Why remove the annotation which indicates sentence structure, say, but then try to recreate it within KEP? The answer is that KEP is designed to process texts from various sources. Plain text is first tagged and then passed to KEP. Since not all taggers provide higher-level annotation, it is not sensible for KEP to rely upon it. Higher-level tags (e.g. those to indicate headings, paragraph starts, sentence structure etc) are not usually created automatically (i.e. by a program) and so cannot legitimately be used in a non domain specific automatic KE program designed to handle any plain text. (There are, of course, exceptions; the CLAWS4 tagger does provide sentence fragmentation.)

4.6.2 Initial Processing

4.6.2.1 Starting the Program

KEP is started from the UNIX prompt by entering 'kep'. The user is then queried as to the various options required for this particular program run. The current list of queries generated by KEP is shown in Table 5. Single keystroke answers are accepted for the Yes/No questions. Note that, in keeping with good HCI practice, default values are provided wherever possible and KEP echoes responses back to the user. It is not possible to run KEP with a "bad" combination of answers, since the questions are structured and presented so as to prevent this.

The decision to use a simple prompt-based method of starting KEP was made because it was not the purpose of this research to develop a sophisticated front-end; cosmetic aspects may be handled at a later date e.g. when KEP is incorporated into some larger system. This approach is also detectable in the limited set of questions posed to the user at KEP start-up. Many variables are set to default values so that

KEP may be started with the minimum of keystrokes. (However, all default values and assumptions made are written to the long output file.) The set of questions presented to the user is alterable at compile time, so that where there are repeated runs of the program under similar circumstances of input it is not necessary to display all the questions. For example, a typical configuration (for processing BNC texts) is to include the questions on the second and third rows, the MENU rows, and the *Do you want to see/print?* rows.

During the program run, diagnostic and error messages (if any) are displayed on the screen. Messages are also displayed as to the current stage of processing attained. These confirm to the user that the program is still running and that all is well. (These messages are not given in the table.)

Query Text	Required Response	Purpose
Where's the input? (filename, or <cr> for default) :	Valid filename, or just <cr>	Names file holding text to be processed
Original BNC file name? :	3-char identifier, or just <cr>	Printed to output to identify the BNC text (if applicable) used as input
Output to where? (filename, or <cr> for default) :	Valid filename, or just <cr>	Names output file
Is the input text already in one sentence per line format? (y/n) n] :	y or n	Finds out whether sentence boundary detection is needed
Are there full stops etc terminating these sentences? (y/n) n] :	y or n	If y to last query, finds out if KEP needs to add punctuation
Is the input text part of speech tagged? (y/n) n] :	y or n	Stops parts of processing which need tags
Tag attachment character(s) ?	one or more characters	Tells KEP what to look for between word and its tag
What are the plural noun tag characters?	one or more characters	Used by KEP in some functions e.g. term acquisition
Do you want to ignore "is a" triggers? (y/n) n]	y or n	Allows general triggers based on verb "to be" to be ignored
Do you want to ignore apposition triggers? (y/n) n]	y or n	Allows specific apposition pattern triggers to be ignored
MENU choice 1 Just do acronyms	1	Only extract acronyms and their expansions
MENU choice 2 Just do TTs and acronyms	2	Get acronyms, their expansions, and technical terms only
MENU choice 3 Just do relation triggering	3	Do nothing but trigger sentences for relations
MENU choice 4 Just do TTs, acros and triggering	4	Get TTs and acronyms, then trigger for sentences having relations
MENU choice 5 Highlight triggered sentences bearing TTs	5	As above but select only triggered sentences having TTs in them
MENU choice 6 Full extractions, TT-bearing sentences only	6	Do all processing for triggered sentences bearing TTs
MENU choice 7 Full extractions, all triggered sentences	7	Do all processing for all triggered sentences
MENU choice 8 Exit	8	Abandon program run
If same text as last run, do you want to restore last run's TTs and acronyms? (y/n) n]	y or n	Allows user to shorten run time when text being processed is identical to that in previous run
Do you want to do all 4 relation types? (y/n) n]	y or n	Allows user to select all relations or pick a subset of them, by the following questions
Do definitions? (y/n) n]	y or n	Marks this relation to be done
Do exemplifications? (y/n) n]	y or n	Marks this relation to be done
Do partitions? (y/n) n]	y or n	Marks this relation to be done
Do hypernyms? (y/n) n]	y or n	Marks this relation to be done *(table continued on next page)*

Look ahead all the way to end of file? (y/n) n]	y or n	Concerns TT look-ahead distance
Look ahead for how many sentences?	an integer	Concerns TT look-ahead distance
Do you want to see the full output? (y/n) n] :	y or n	Allows user to see output on screen if desired (longer output, with diagnostic messages)
Do you want to see the short output? (y/n) n] :	y or n	Ditto, short output file (no diagnostics)
Do you want to see the glossary output? (y/n) n] :	y or n	Ditto, glossary output file
Do you want to see the term summaries output? (y/n) n] :	y or n	Ditto, term summaries output file
Do you want to print the full output? (y/n) n] :	y or n	Allows user to request printout
Do you want to print the short output? (y/n) n] :	y or n	Allows user to request printout
Do you want to print the glossary output? (y/n) n] :	y or n	Allows user to request printout
Do you want to print the term summaries output? (y/n) n] :	y or n	Allows user to request printout

Table 5. KEP user queries

4.6.2.2 External Storage

KEP uses permanent data held in various files on disk. In addition, both input and output reside in disk files. These files[7] are listed in Table 6 and their uses will be discussed in the descriptions which follow.

4.6.2.3 Internal Storage

Two major areas of internal storage are reserved by KEP: the first holds the input text on a line-by-line basis, and the second holds it on a sentence-by-sentence basis. In addition, various linked lists are created dynamically as required. The input text and sentence arrays are fixed in size and so do not present memory problems after compile time, except where sentences are longer than the pre-set maximum sentence length (800 characters). The linked lists use the standard 'C' library malloc() system call to obtain memory for new elements, and the success status of all such calls is monitored and reported upon. Failures are rare and only occur where memory for the process has been restricted by external factors beyond the user's control. Various internal limits have also been incorporated to prevent demands for excessive amounts of memory (hundreds of megabytes) and indeed to prevent excessive run times. These limits do have implications for KEP's performance but do not greatly affect recall and precision.

[7] NOTE: where files contain patterns of part-of-speech tags, these will be tagset dependent (and hence the correct version must be used for the tagged input).

Filename	Purpose
defcontag.txt	definition concept tag patterns
excontag.txt	exemplification concept tag patterns
ptcontag.txt	partition concept tag patterns
hypcontag.txt	hyponym concept tag patterns
defelutag.txt	definition elucidation tag patterns
exelutag.txt	exemplification elucidation tag patterns
ptelutag.txt	partition elucidation tag patterns
hypelutag.txt	hyponym elucidation tag patterns
deftrigs.txt	definition positive trigger phrases
defntrigs.txt	definition negative trigger phrases
extrigs.txt	exemplification positive trigger phrases
exntrigs.txt	exemplification negative trigger phrases
pttrigs.txt	partition positive trigger phrases
ptntrigs.txt	partition negative trigger phrases
hyptrigs.txt	hyponym positive trigger phrases
hypntrig.txt	hyponym negative trigger phrases
deftoks.txt	definition token characters
extoks.txt	exemplification token characters
pttoks.txt	partition token characters
hyptoks.txt	hyponym token characters
defpats.txt	definition pattern templates
expats.txt	exemplification pattern templates
ptpats.txt	partition pattern templates
hyppats.txt	hyponym pattern templates
defpres.txt	definition presentational indicators
expres.txt	exemplification presentational indicators
ptpres.txt	partition presentational indicators
hyppres.txt	hyponym presentational indicators
termtag.txt	technical term tag patterns
kep.in	default input text file
kep.out	default long output file
kep.tout	output of sentence structure only
kep.sout	default short output file
kep.gout	default glossary output file
kep.sumout	default term summaries output file
kep.ken	default knowledge extraction network output file
tttest.out	all text fragments used in making TTs
ttnola.out	all potential TTs that occurred just once
ttdisc.out	complete dump of all internal TT data
acdisc.out	complete dump of all internal acronym data

Table 6. Files associated with KEP

4.6.2.4 Obtaining Sentence Structure

The first action of KEP is to open the file containing the tagged input text (either a BNC file or a text tagged using CLAWS and then re-formatted using the conclaws program - see Table 4) and to read it into the line structure storage array (currently, a maximum of 14,000 lines can be processed). An example of tagged text read into the line structures is given in Figure 14. Note that not all words may be tagged - CLAWS tags only the first word of certain multi-word phrases, such as *apart from*, *in case*, *for sure* etc. This would cause processing problems due to the mixture of tagged and untagged text, so the second action taken is to find all untagged words in the input and attach a tag of the form **_XXX**.

```
Sorting_VVG is_VBZ the_AT0 action_NN1 of_PRF arranging_VVG data_NN0 items_NN2
into_PRP some_DT0 specific_AJ0 order_NN1 ._. We_PNP can_VM0 define_VVI a_AT0
sort_NN1 routine_NN1 (_( SR_UNC )_) to_TO0 be_VBI a_AT0 function_NN1 which_DTQ
orders_VVZ a_AT0 list_NN1 of_PRF items_NN2 according_II21 to_II22 some_DT0
criterion_NN1 ._. Examples_NN2 of_PRF SRs_NP0 include_VVB the_AT0 bubble_NN1
sort_NN1 and_CJC the_AT0 quick_AJ0 sort_NN1 ._. Sort_NN1 routines_NN2 are_VBB
composed_VVN of_PRF four_CRD elements_NN2 :_: input_NN1 list_NN1 ,_, output_NN1
list_NN1 ,_, sort_NN1 criterion_NN1 and_CJC sort_NN1 algorithm_NN1 ._. An_AT0
example_NN1 of_PRF a_AT0 sort_NN1 criterion_NN1 is_VBZ alphabetical_AJ0
order_NN1 ._. A_AT0 sort_NN1 routine_NN1 is_VBZ a_AT0 type_NN1 of_PRF data_NN0
rearrangement_NN1 algorithm_NN1 ,_, or_CJC DRA_NP0 ._. In_PRP these_DT0
,_, data_NN0 elements_NN2 are_VBB not_XX0 themselves_PNX altered_VVN ,_,
but_CJC their_DPS order_NN1 of_PRF presentation_NN1 is_VBZ changed_VVN to_TO0
assist_VVI the_AT0 calling_AJ0 application_NN1 ._.
```

Figure 14. Example of CLAWS-tagged input text, after 'conclaws' pre-processing with C5 tagset mapping

In the KEP system it was decided to attempt KE on a sentence by sentence basis, since the sentence is a natural unit of expression capable of holding facts, and has traditionally been described as "the complete expression of a single thought" (Crystal (1987)). Although some kinds of knowledge may be spread over several sentences, such as procedural descriptions (see e.g. Vander Linden and Martin (1995), Sutcliffe et al. (1995), Skuce et al. (1995)), the sentence is the basic unit from which texts are built. Texts are not arbitrary collections of unrelated sentences; they are coherent. The coherency-creating devices include anaphors and cataphors, but these do not essentially alter the position of the contained fact – it is still within the sentence, even if part of it is actually a pointer to some other part of the text.

However, it is not mandatory to use the sentence as the basic unit for pattern-matching KE, as used by KEP. It is conceivable that a KE system might ignore all sentences altogether, or it might consider phrases within sentences, and indeed partial parsers have been built which consider parts of sentences (e.g. the SPARSER system of (McDonald (1992)). Therefore some justification for using the sentence as a basic unit is required. The first justification has already been given in the above paragraph; a sentence is traditionally the natural unit for expressing a complete idea. Thus it is physically large enough to contain a fact. Sub-units of sentences are probably not big enough to contain complete definitions etc, although as sentences may be arbitrarily long this needs qualifying: *phrases* within sentences are

probably not large enough. The sentence therefore seems to be the minimum-length section of text worth considering for the chosen application.

What of the upper bound? Why not use a paragraph, section, chapter, or indeed an entire text? This is answered by practical considerations. The shallow extraction method to be used (justified in section 4.2.2) requires matching of lexical patterns against text. If the unit to which the pattern is to be matched is very large, this process becomes unwieldy and may in fact be infeasible. The specific technique used by KEP would not in fact be practicable for very large pieces of text, for it is based upon an exponential method (described shortly) which would not be able to make the match against a whole text. However, since patterns are matched against one sentence, and then the next, and so on, one might regard it as actually matching against the whole text in the sense that the pattern *scans through* it. The text is chunked into sentences for the purposes of this scanning, but it might equally well be chunked by the lines of input. The problem with the latter is that sentences, containing as they do an "entire thought", would not always be matched against the pattern, for sentences can go over the end of one line and onto the next etc. The patterns looked for are themselves "entire thoughts" containing definitions etc, i.e. they are themselves naturally expressed as sentences. Thus if one line were to end with the words **Let us define a widget as** then the pattern match for an entire definition such as **Let us define a X as Y** would never occur. Matching against a paragraph would work, but again this would be defeated by practical text-size considerations, as would any unit larger than a paragraph. Therefore the sentence emerges as the basic unit for the pattern matching approach to the extraction of definitions etc from text. Thus the first task is to cut the input text into sentences.

Code exists to handle the case where each sentence is on its own line, and if this is the case KEP copies the line structure array to the sentence structure array directly. However, the default case is that sentences are spread across any number of lines (including the case where more than one sentence can be present on a single line of input) and so KEP attempts to split the input into sentences to fill the sentence storage array. Splitting text into sentences is not a trivial task. It is very difficult to achieve 100% correct division if tags within the input are not used. (See discussion above concerning the deliberate non-use of <s> tags.)

It is generally conceded that a good sentence-end detector function should correctly reveal 95% of sentence boundaries or better (Palmer and Hearst (1994)). One of the main obstacles is the presence of potentially sentence-ending punctuation within abbreviations, such as in *He arrived at 5 p.m. in his car.* Another problem is the use of sentences within quoted speech, such as *"Let's go!", he said.* Headings also prove problematic if they were present in the original text in a different font without a terminating punctuation mark, or contain numbering involving full stops (e.g. see this section's heading). Sentence-end detection may also require the detection of the start of a following sentence, e.g. where a sentence ends in an ellipsis printed as three full stops.

Various sophisticated methods of finding sentence boundaries have been proposed, such as that described in Palmer and Hearst (1994), where a neural net is used to examine the likelihood of a sentence ending at a particular mark, based upon part of speech tag patterns in nearby text. The resulting claimed accuracy is 98.5%, most of the failures being due to title/name collocations where the name occurred in the lexicon (e.g. *Col. North*) and cases where the sentence ending was missed as a result of an abbreviation ending the sentence. However, unsophisticated methods relying upon large amounts of laboriously collected abbreviations etc can also give high accuracies, such as those of Wasson reported upon in Palmer and Hearst (1994). The pragmatic view taken for KEP was that as long as 95% or so of sentences were correctly delineated, useful KE could be performed. Thus the approach was to mark sentences as ending at certain punctuation marks (namely . ! ?) unless these occurred in certain lexical units, some of which are listed in Table 7. (For the purposes of brevity, upper case versions are not listed.) Tests on BNC files indicate that KEP correctly identifies over 90% of sentence boundaries (see section 5.3.1). This figure could undoubtedly be improved, but this task is a small part of the KE goal and was not a high priority area for KEP.

Phrase	**Problem Caused By**
e.g.	Either full stop
i.e.	Either full stop
fig. n	Full stop in reference (n is numeric)
no. n	Full stop in reference (n is numeric)
n.m	Full stop in reference (n,m numeric)
...	Full stops used as ellipsis punctuation character
Mr.	Full stop in title
Mrs.	Full stop in title
Dr.	Full stop in title
Ms.	Full stop in title

Table 7. Some sentence boundary exception phrases

Where sentences are more than the allotted 800 characters in length, they overrun into the following sentence array. To avoid this, too-long sentences and the sentences which immediately follow them are marked as unusable and do not take part in any further processing. Such sentences are rare (usually less than 1 in 200 sentences, although this ratio varies from text to text.) In most cases they arise through missing end-of-sentence markers in the tagged text input or in the plain text prior to tagging. Although no specific study has been made, it is not thought that such occurrences contribute noticeably to a drop in the extraction recall metric. This approach to too-long sentences, together with the _XXX tag adder code, ensures that KEP remains robust i.e. does not crash due to input irregularities.

The third action taken is to create a separate array of sentences stripped of tags. During the subsequent processing, depending on the function to be performed, the sentence array is chosen which makes the processing easier or faster. All sentences are numbered; corresponding tagged and untagged sentences have the same numbers. Sentence numbers are used widely in screen and file output.

4.6.3 Heading Identification

Titles, headlines, by-lines, chapter headings, column headings and section headings are all parts of printed documents such as newspapers, reports and books. These headings may be present in the tagged single-font single-pointsize ASCII text which is input by KEP, usually on their own lines of input. Unlike some systems designed to handle complex page layouts (such as Myers and Mulgaonkar (1995)), KEP is unconcerned with such matters. However, headings are rarely full grammatical sentences and often are not terminated by punctuation (since, in the original source, page layout conventions or a different font or point size were used). The lack of a terminating punctuation mark can result in a heading being prepended onto the first sentence following the heading by KEP's sentence-end detector, resulting in an incorrect sentence delineation. For this reason alone, KEP detects very simple commonly occurring headings in the input document so as to mark them as separate "sentences". Simple headings detected include "Introduction", "Conclusion" etc. This simple approach usually detects a handful of cases for each input BNC text, and contributes to the sentence-end detection rate.

Heading detection in the absence of font, pointsize, or vertical spacing clues is difficult. Even where there are numbered section headings the task is not simple. To do the job properly requires examination of the preceding line of text (e.g. to look for a terminating full stop) and the following line of text (e.g. to see if it appears to be the start of a new sentence). It may also include the need to scan several lines behind and forward, to search for headings in a numbered sequence. In addition, the ability to detect phrases which are not well-formed sentences is helpful. These tasks were not attempted for KEP since the effort did not justify the small increase in extraction performance which might result from perfect heading detection. In a sense this would have been an artificial task, for in those systems which do attempt to detect headings the layout clues are invariably used. The correct place for this task is at an earlier stage in document processing.

4.6.4 Technical Term Acquisition

Technical Terms (TT), or *specialist terms*, have been introduced earlier. KEP uses TTs in various ways. In the glossary output, TTs form most of the middle column. Technical terms are (some of) the concepts which are being defined etc in the input text, and so KEP needs to be able to identify them.

Technical terms are usually domain dependent. How then could a *non* domain dependent system ever hope to identify them? One way would be to provide the program with lists of technical terms from all domains. Clearly this is not a practical proposition; the lists would require extensive collection time and effort, would be huge (perhaps containing many thousands of entries), and would need to be constantly updated to add terms recently coined (e.g. *cold fusion*, *web spider* etc). Not only that, but as terms move through their lifecycle, from coinage through common usage to obsolescence, their accepted meanings often change. This process has been discussed by Ahmad (1996) and Ahmad and Collingham (1996).

Although specialist term bases do indeed exist, the NDS nature desired for KEP would mean that very many DS term bases would be required, and furthermore, KEP would then need a means of determining which of them to apply (this is essentially the text topic identification problem). This approach is clearly not practicable. Therefore, another method must be sought. This other method should detect most or all of the technical terms within a document, for any domain, but not return things which looked like terms but were not.

Fortunately, this goal turns out to be partially achievable, using a relatively straightforward method which relies upon the fact that technical terms, whatever their domain, take similar syntactical forms and usually occur more than once in any given document. The method unfortunately cannot detect single word terms, but gives a high success rate for n-word terms where n is 2 or more. This is the method of Justeson and Katz (1995). The basis of the algorithm is the observation that technical terms are almost always multi-word noun phrases, which consist of adjectives and nouns and sometimes prepositions, but very rarely verbs, adverbs or conjunctions. KEP uses a modified version of this method (the modification being a different way of determining parts of speech). In addition, KEP finds single-word TTs in a manner described shortly.

The point is made by Justeson and Katz that technical terms are *lexical* i.e. they can be treated as words and must appear in the lexicon. For example, *central processing unit* is to all intents and purposes a single word, whose meaning is quite specific and which is more than the sum of its parts. Technical terms also *occur repeatedly* throughout the text but also in such a form that *their modifiers vary less* than for other NPs. The two properties mentioned in italics are therefore used to aid in the technical term acquisition. Concerning the latter point, Justeson and Katz point out that there are factors within text which actually prevent the exact repetition of NPs which are *not* technical terms. One of these is variation so as to avoid monotony (for the reader). Another is the point that NP premodifiers are often there to emphasize some particular aspect of the entity in focus, and so later in the text where this aspect is no longer to the fore, different premodifiers are used. (Omission of modifiers is regarded as a form of change of modifier.) Thus for **non** technical terms, it is unusual to have the exact same NP repeated throughout a text. This means that when we do see a repeated NP (especially one of a particular part-of-speech pattern) it is likely to be a lexical item i.e. a technical term.

The claim about the NP nature of technical terms is backed up by technical dictionary studies, which suggest that between 92.5% and 99% of technical terms are NPs, of which 99% do not use verbs, adverbs or conjunctions. The reasons for this are discussed, including the notion that two or more words are needed for the required degree of precision (i.e. to distance the term from general usage senses), and the point that technical domains often use a hierarchical taxonomy which lends itself to multi-word terms where lower levels in the tree can be achieved by adding extra modifiers (these extra modifiers themselves being "standard" in the sense that they occur at the same level in the tree in different

branches). The average term length from the technical dictionaries was 1.91 words (medicine was 1.78, and fibre optics was 2.08 - these are the lowest and highest domain averages). So 2-word terms dominate. (Medicine is peculiar in that Greco-Latinate compounds are used, which are like multi-word terms in those languages - e.g. synarthrophysis = syn + arthro + physis = "together growing joints").

The algorithm involves looking for strings using the following rules. (1) Candidate strings must occur at least twice in the text (2) Look for candidate strings meeting the regular expression:

((A|N)+|((A|N)* NP)?)(A|N)*)N

where A is an adjective, N is a noun, P is a preposition

Putting this into words, "a candidate term is a multi-word noun phrase; and it either is a string of nouns and/or adjectives, ending in a noun, or it consists of two such strings, separated by a single preposition". The regular expression generates 2 patterns of length 2 words, and 5 of length 3 words. These are listed in Figure 15.

AN e.g. lexical ambiguity, conceptual relation
NN e.g. knowledge extraction, discourse structure, word sense, term acquisition, noun phrase
AAN e.g. Gaussian random variable,
ANN e.g. lexical ambiguity resolution, natural language processing
NAN e.g. domain independent extraction
NNN e.g. text analysis system
NPN e.g. analysis of text

Figure 15. Term patterns from Justeson and Katz (1995)

The implementation of the algorithm by Justeson and Katz themselves does not use a tagger. Instead, allowed parts of speech for each word are obtained from a dictionary and assigned as N, A, P by preference if this is possible. This is done so that e.g. *fixed disk drives* comes out as ANN and not VNN. The authors call this "filtering". They mention problems that it causes (verb/noun ambiguities), but results have been impressive (recall and precision, or "coverage" and "quality", as the authors call it, both consistently being over 90%). Terms not obtained sometimes occurred because they were only mentioned once despite being technical terms (e.g. *Heaviside function*), often because they were not topical in the text.

A version of this algorithm has been coded into a KEP function. KEP is able to utilise part of speech tags (and so does not need MRD lookup) and the patterns corresponding to those in Figure 15 are stored in an external file (termtag.txt - but see later for further discussion on this technique). The method does indeed appear to extract technical terms with high recall and precision, and the false positive rate seems to be low. (Full evaluation results are given in the next chapter.)

The operation of looking for two- and three-word TTs in sentences involves cutting each sentence into all possible fragments 2- and 3-words long, obtaining the tag set for each such fragment, and comparing each tagset against patterns held in termtag.txt. (Since punctuation marks are treated as individual words, fragments involving punctuation arise; these are rejected.) If a match occurs, then the sentence fragment is a potential TT. One practical problem encountered during the development of the TT stage was related to the tagset used by BNC texts (i.e. by the CLAWS tagger). The problem was that there are in fact over 2,000 ways of providing the part-of-speech tag patterns given in the above table, due to combinations caused by the fine divisions of tags (e.g. NN0, NN1, NN2, NN1-NN0, NN1-VVG, NN1-VVB etc are all tags for nouns). Thus each fragment of each sentence had to be compared with up to 2,000 or so patterns from termtag.txt. Since a sentence of **n** words has **2n-3** two- and three-word fragments (n >= 2), then a 20-word sentence would require up to about 75,000 comparison operations. This resulted in processing times of minutes per sentence. This problem was overcome by the realisation that fortunately the CLAWS tagset is such that only the first letter of each tag needs to be read in order to classify a word as a noun or an adjective, with the exception of the tags AT0 (used to tag articles), AJC (comparative adjectives), AV0 (adverbs), AVP (adverb particles) and AVQ (*wh*-adverbs). Thus the patterns given in Figure 15 were able to be hardcoded and only up to 7 comparisons performed for each fragment. This reduced potential TT identification processing times to a fraction of a second per sentence. However, this method is only possible because of the naming convention of the CLAWS tagset, and so the use of the termtag.txt file lookup might be required for another less well designed tagset.

When a potential TT has been identified in a sentence, then the rest of that sentence and all other sentences following must be checked for a second occurrence of the potential term. One way to do this is to store all potential TT fragments and increment counters as these are repeatedly found. However, this method turns out to be impractical, due to the memory requirements. In a 2,000-sentence text there might be 10,000 different potential TTs, of which only 200 have counts greater than one. Therefore KEP uses a look-ahead mechanism to detect second occurrences of potential TTs, and stores only those having a count of 2 or more. The look-ahead mechanism is a "fast" method which does not look for an exact match for a potential term, or a version of it in a different number (singular or plural - see next paragraph). In addition, the user may specify a look-ahead distance in sentences, in order to reduce processing times for long texts. Ad-hoc experiments have shown that the majority of potential TTs repeat within twenty sentences of their first occurrence; clearly this is related to the topic substructure of text. The look-ahead method will, however, miss TTs which are widely separated if the user does not choose a look-ahead right to the end of the text.

Since the algorithm requires the counting of occurrences of potential terms, some method of finding out whether a plural term is the same basic term as a singular term must be provided. For example, a text may contain one occurrence of *chain reaction* and one of *chain reactions*, or one of *director of studies* and one of *directors of studies*, and the function must count at least two occurrences of the potential term

in order to highlight it as a real term. Thus a function is required to either find the plural form of any singular noun, or the singular form of any plural noun. The latter approach has been adopted because it is probably easier (nouns such as *formula* have more than one recognised plural, i.e. *formulas* and *formulae*, so more checking would be required if the transformation were to go in this direction). Section 4.6.13 details the novel function which does this. Note also that the correct noun in the term must be singularised - the last in the case of terms made solely from adjectives and nouns, but the first in cases such as *men of war*.

Fast look-ahead does result in the storage of a few potential TTs which are subsequently not confirmed as such, due to the practice of examining only the first few characters of the fragment (i.e. coincidentally these characters form the start of a different phrase) or, in the case of terms of the format NPN, the last few characters of the fragment. Such "unconfirmed terms" are rare and therefore have no speed or memory repercussions. These cases do include real TTs which are mentioned only once in the text (and hence which are lost) but the majority are due to inconsistent tagging in the source. Conversely, some non-TTs are identified as real terms. Examples include *recent year*, *old friend*, *serious error* etc. An attempt has been made to identify such "duff terms", but the task is difficult and the present mechanism is not reliable. Unconfirmed terms and duff terms are listed in the long output. In addition, all tagged text fragments from which TTs were derived are placed in an external file (tttest.out), and all such fragments for which the corresponding TT occurs only once in the source are written to a separate output file (ttnola.out). These two output files are used in conjunction with the duff/unconfirmed lists to aid in TT performance evaluation.

The single word technical term extraction problem remains, despite the suspicion that technical terms are not often one word long. (However, some subjects are probably more prone to singleword TTs than others - medicine, for instance. For other domains, the percentage of such terms probably hovers around the 10% mark.[8]). Where these are acronyms, they can be recognised from their expansions. However, non-acronym single word terms may be detectable from hyponymic relations. This possibility has also been recognised by others, e.g. Reimer (1989) and Rousselot et al. (1996) and arises in situations where (a) one term is a substring of another, or (b) terms have common endings. The latter is exemplified by *mainframe computer* and *personal computer*, from which the term *computer* may be deduced. The former occurs with e.g. *room temperature superconductors* and *superconductors*, where the shorter term is the hypernym (parent class) of the longer term, the shorter term corresponding to the right-hand end of the longer term.

Method (a) has been coded into KEP for 2-word TTs only. Although useful, it can sometimes generate singleword terms which are too broad to be regarded as good TTs. For example, the terms *RMS error*

[8] In the domain of satellite communications, Nkwenti-Azeh (1994) found that single-element terms occurred as 9.15%, 7.30% and 30.73% of three separately-derived term lists respectively.

and *Poisson error* would give rise to the term *error*. Cases (a) and (b) are illustrated in Figure 16. A third method (c) is to consider terms where the first word is a noun acting adjectivally to a following noun. For example, *map error* and *map reference* would give the singleword term *map*. This method has not yet been coded for KEP, although interestingly it is often the case that singleword terms it would have created were already generated by method (a) from other sentences.

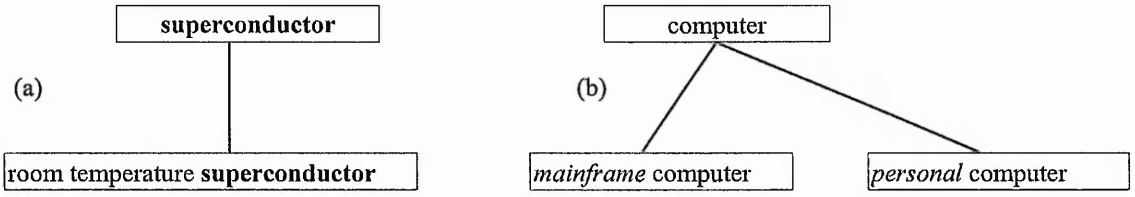

Figure 16. Hyponymic relations from technical terms

4.6.5 Acronym Acquisition

The next stage of processing is to find all acronyms in the input text, and if possible, what they stand for (called by the author the acronym's expansion). The novel acronym extraction method devised for KEP is described in detail here due to its importance in the extraction process; it has also been described in Bowden, Evett and Halstead (1998). Although the word 'acronym' has historically been used only to describe abbreviations made from word-initial letters which may be pronounced as a word (e.g. NATO), rather than as separate letters (e.g. BBC), it is used here for both senses. (See Daintith et al. (Eds) (1993) for a discussion on the nomenclature of abbreviations.)

Acronyms are used throughout explanatory text. The TLA (three-letter acronym) is a feature of modern technological life. Some organisations are more prone to their use than others; NASA, for example, uses many thousands of acronyms in an attempt to label the multitude of systems used in spacecraft. This often leads to problems for newcomers to such companies, for there is a large number of acronyms to be learnt. Within the aircraft manufacturer Boeing the situation became so critical that one of the computational linguists employed in technical manual production created an acronym look-up program available to online users all over the company. Although this was an "unofficial" project, it soon became one of the most heavily used software packages within the company. This system utilises tables of acronyms plus their expansions, so the system required many man-hours to create, and still requires continuous updating[9]. It is clear that there is a pressing need for an automatic acronym extractor capable of creating lists of acronyms and what they stand for, from text. The acronym extractor designed for

[9] No details of this system have been published. The author learnt of it during a conversation with its designer, Richard W. Wojcik, at the COLING'98 conference.

KEP does just this, although at present it does not archive acronyms from one run to the next. (This would be a minor enhancement to make.) Acronyms occur as the first column of the glossary output, and thus are linked to extracted middle-column terms. The knowledge of what an acronym stands for also allows better cross-referencing in the third column.

In the KEP glossary creator, acronyms are assumed to be all-caps, except where they are in the plural (e.g. CPUs). Acronyms can be *exact* (nomenclature of the author) meaning that each and every letter in the acronym matches a word in the expansion in the same order and that there are no words in the expansion not represented in the acronym. For example, for the acronym GIS the expansion words start G I S (Geographical Information Systems). Acronyms may also be *inexact*, meaning that there are extra letters in the expansion (e.g. as between IUCN and International Union for Conservation of Nature, which actually starts I U F C O N), or extra letters in the acronym (as between RIMNET and Radioactive Incident Monitoring Network, which starts R I M N).

The first step is to search for all-capital words. KEP is currently not capable of finding mixed-case shortenings (such as *DfE* for Department for the Environment). Acronyms containing full stops are not found in BNC texts - instead acronyms occur as capitalised words containing no punctuation, tagged in a variety of ways (such as NP0 for proper noun or NN2 for plural noun). Often the tag used is not correct, but this does not affect the acronym extractor since the tag is not read. Certain words such as FIGURE are held in a stop-list of probable non-acronyms, to reduce the occurrence of such forms being regarded as acronyms, and in addition all all-caps words longer than 7 letters are ignored. When a potential acronym is found, then if this is the first time it has been seen in the text, its expansion is searched for in the current sentence. If found, it is stored with the acronym. If not found, then the previous sentence is searched. If still not found, then when the acronym is next seen in a future sentence, the search will be repeated. Thus KEP never gives up looking for the acronym's expansion, which may not always be given near to the first occurrence of the acronym.

Acronym expansion detection involves fragmenting the sentence into n-word sections, where n is the number of letters in the acronym (e.g. n=3 for GIS), and *also* into other fragment lengths ranging from 1 to n+3 (but inside the range 1 to 10). This is necessary for inexact expansion finding. A list of candidate expansion fragments is then drawn up, using the test that a candidate must have word-initial letters forming a string which is better than 60% the same as the acronym. This is achieved using a version of the Ratcliff-Obershelp string-comparison algorithm (Computing (1992)). Candidate expansions must not start with the acronym itself, and must not start or end in a "glue" word such as *and, for, of, by*, etc.

For the expansion candidates, scoring heuristics are applied as follows:

(a) the percentage match as above adds points 10 for 100% match, 9 for 90 - 99%, 8 for 80 - 89% etc,

(b) the presence of the candidate expansion within brackets (or dashes, or other such markers) adds 10 points,

(c) the presence of the acronym within brackets (or dashes etc) *contiguous with* the candidate expansion adds a further 10 points, but only 5 points if it does not follow *immediately* after,

(d) if a candidate has glue words within it, then if when stripped of all its glue words this would give rise to an acronym identical to that being processed, another 10 points are added,

(e) if an acronym is constructed from *all* the capital letters in the candidate, and this matches the acronym, 10 points are added.

The highest scorer is chosen as the expansion, subject to a minimum threshold of 10 points.

In addition to the above, single-word candidates which contain hyphens, such as *red-green-blue* for RGB, are handled with extra code to remove the hyphens so that they may use the same scoring method. A separate function also attempts to identify Roman numerals, such as IV, which may or may not be acronyms (IV might stand for *In Vitro*). This is done by examining the preceding word. Thus *Mark IV* would indicate a Roman numeral. (A future development will augment the decision making process by searching for runs of numerals in preceding and following text.)

The above scoring heuristics were developed on an ad-hoc basis but subsequent evaluation (see section 5.3.3) has shown that the correct expansion is extracted approximately 85 times out of 100 if it was present in the text. Out of the five scoring rules given above, the dominant factor is usually rule (c), because it appears to be very common to introduce a new acronym by giving the phrase first and then placing the acronym in brackets immediately after it. Rule (e) is also quite useful since it can find abbreviations such as RIMNET if their expansions are like 'Radiation Incident Monitoring NETwork'. The acronym extractor produces a series of counts concerning the above scoring rules and thus is able to report upon the incidences of all types of acronym-expansion syntaxes used in a corpus.

Not all acronyms have their expansions found from the text (e.g. USA is rarely expanded in text, because probably all adult English-speaking readers know what it stands for). In keeping with KEP's NDS approach, no domain specific lists of acronyms are stored internally, but very common **NDS** acronyms such as UK, US etc are held internally, and the list of these is consulted only after all attempts to find

expansions from the text have failed. This allows local usage to take precedence over the general usage. This list contains less than ten entries.

Where an acronym has been found which has an expansion, then that expansion is stored in standard form as used for technical terms, i.e. lowercase word-initial letters and singular form. It is then checked against all the TTs previously found, to see if it is identical to one of them. If so, a cross-reference is made (2-ways) between the acronym and the existing TT. If an acronym is not linked to a TT, its expansion is added to the TT list as a *new* TT. This is done regardless of whether or not the expansion agrees with any of the TT allowed tag patterns. This catches the case where an author introduces a term and gives its acronym, but then subsequently refers to it only by its acronym. It also allows all acronyms for which an expansion has been found to appear in the glossary output.

Where acronyms occur *within* other TTs, they are restored to all-capitals form (because in standard TT form, **GIS** would be **gIS**). In the glossary output, cross-references are provided (so that for an entry for **GIS integrity** (middle column) there would be a comment in the third column **SEE ALSO geographical information system**). No processing yet takes place to handle acronyms within acronyms (e.g. XSQL = Extended SQL).

4.6.6 Term Summaries

KEP provides an output file which lists all TTs (or TT-acronym pairs if linked) together with the text in which they occur. Several pages from a substantial term summaries output are given in Appendix B (derived from the BNC text 'B1G'), although a smaller example has already been given (Figure 12). Blocks of text containing each TT are printed with vertical ellipses separating text blocks. If the gap between text blocks is less than or equal to two sentences then the gaps are filled-in with the non-TT bearing sentences. This approach allows the reader to pass smoothly over short sections of text which do not explicitly mention the term.

The term summaries allow a reader to focus on those parts of the source text relating to a given technical term (concept). They represent a type of term-specific text summarisation. There are several uses for the term summaries:

(1) Since they contain all the text from which any relation extractions have been made (see following section) they allow KEP extractions to be checked against the original text without the need to check a large document against a list of sentence numbers. Furthermore, if the KEP glossary maker were to be later incorporated into a word processor, it would be possible for the editor of the glossary to bring up the term summary entry for a highlighted concept in a separate window. This would allow the editor to refine KEP's attempted extractions by cut-and-paste methods from the original text.

(2) Term summaries are a form of high-level KE in their own right. Not only is KEP saying that "this document talks about concept X" but it is also saying "furthermore, this is what it says about X". A caution, however: ideas built up by a reading of the whole text may not be present in the term summary alone. Thus this method of concept extraction may miss text-level ideas. However, since "gap" sentences are often provided, and since ideas to be conveyed by the text are often developed within a contiguous section of text, it is likely that a useful summary of the concept is present. In the words of Rau, Jacobs and Zernik (1989), who were discussing the use of text summaries to answer specific user queries, "Summaries of whole texts do not replace source texts. In many cases, a document *or section of a document* is an appropriate response to a user query." (Italics the author's addition.) Thus term summaries have a role in query-based IR, or in a system which effectively auto-generates those queries (i.e. using technical terms as search terms).

(3) Term summaries provide information as to the structure of the input text, as to in which parts of the text a particular concept is discussed. Although this is not the aim of KEP, it may lead to methods of identifying topic substructures within text, currently an active area of research (see e.g. Hearst (1994), Rose and Evett (1993a)).

4.6.7 Relation Detection and Triggering

In the descriptions which follow, the processing described is that which attempts to fill the third column of the glossary-format output, and the equivalent parts of the other output formats. The desired extractions are the definitions, exemplifications, partitions and hypernyms of concepts. The extraction of these concept elucidations is a challenging task and has not been attempted before in a NDS way from explanatory text. Thus both the task itself and the method used are novel.

The purpose of the triggering stage is to find those sentences which possibly contain a conceptual relation of interest. To illustrate triggering, the definition conceptual relation alone will be discussed. However, the same basic method is used for all four relation types handled by KEP.

For systems which detect and process definition relations from dictionaries and thesauri, the act of relation *detection* is straightforward, for each book entry is certain to be a candidate. Such systems, for example Alshawi (1987), Zhu and Shadbolt (1995), Martin (1992) tend to concentrate on the extraction of the various semantic parts of the definition into some useful data structure. The aim of the KEP program is to extract entire word strings rather than to dissect extracted concepts into their semantic parts. However, the detection of definitions within running text is not as straightforward as the simple location of definitions in a dictionary. Consider sentences **a** through **d** below, which are intended to represent sentences taken at random from some body of text.

a A marsupial is defined as an animal with a pouch for its young.
b A byte is a contiguous group of eight bits.
c A television set is a modern marvel.
d There is a way to do this.

Clearly, **a** and **b** are definitions, whereas **c** is not (it is merely a statement about televisions in general). Sentence **d** is clearly not useful as a standalone. Note that **a** contains the trigger phrase *is defined as* but **b** through **d** contain only the very general phrase *is a*. It is difficult to pin down what makes **b** a definition but **c** not. However, notice that Skuce et al.'s definition-test method (see page 73) is successful in distinguishing **b** from **c**. Sentence **b** does indeed answer the question *What is the meaning of a byte?*, but sentence **c** does not answer the question *What is the meaning of (a) television set?*. This topic is returned to in section 6.2.2.

Relation detection in KEP is provided by a triggering mechanism. Positive and negative trigger phrases are used to locate possible instances of conceptual relations. The character string *define* is sufficient to catch sentence **a**. The negative trigger string *cannot be defined* can be used to rule out some sentences containing the characters *define* in the wrong sense (e.g. *The 3-body algorithm cannot be defined*). Thus the positive/negative triggering method first highlights all sentences containing positive triggers, and subsequently rejects some of these if the positive trigger phrase detected was part of a larger negative trigger phrase. Each relation type processed by KEP has two trigger data files associated with it, one each for positive and negative trigger lists (see Table 6). These lists have been found to be short, and the method of searching for them is described in the following chapter. It is found that very approximately one in 100 positively triggered sentences are subsequently de-triggered due to the presence of an overlapping negative trigger, although this figure can vary greatly from text to text.

Just one positive trigger (without a corresponding negative trigger) is enough to allow the sentence through the triggering filter. The triggering mechanism does not find *all* the positive triggers which exist in the sentence, because this is not necessary. But note that if a positive trigger is found in a sentence, and this positive trigger is subsequently found to be part of a negative trigger, then the sentence is de-triggered. However the sentence then searched for the *next* positive trigger in the positive trigger list. Thus the first positive trigger which is not cancelled out by a negative trigger is detected (if indeed there is such a thing in the sentence).

Triggering illustrates the filtering approach used throughout KEP. Starting with a set of sentences, too-long sentences are filtered out, then from what is left (good-length sentences) headings are filtered out, then from what remains (good-length sentences which are not headings) all sentences not containing positive triggers are filtered out, then from the remainder (good-length non-heading triggered sentences) those containing negative triggers are filtered out, and so on through subsequent stages of processing.

It is important to realise that the purpose of triggering is to highlight sentences that *might* contain an instance of a particular conceptual relation, not ones that *definitely do*. The latter is left to the pattern matching and subsequent stages. Thus KEP errs on the side of caution in this stage.

KEP does not need domain specific knowledge for triggering, unlike some similar systems. For example, the "wit" system of Reimer (1989) required a small amount of domain knowledge to "focus its attention" on relevant parts of the text, in addition to linguistic clues. However, KEP uses only surface linguistic knowledge to detect interesting sections of text.

4.6.8 Apposition Triggers

If no trigger phrases are found in a sentence, KEP looks for separated markers which may signal the presence of apposition. For example, in the sentence *The potto, a type of lemur, is rarely encountered* the two commas signal that *The potto* is in apposition to *a type of lemur*. This sentence contains an instance of the hypernym relation and therefore should be further processed. However, the two commas are separated by several variable words and so the positive triggering mechanism described above cannot be used since it searches for fixed phrases. Therefore a separate function is used to detect sentences that might contain appositive facts. Separated marks which might include apposition include , ... , and - ... - and (...) and - ... . (the ellipsis indicating one or more separating words which form the appositive phrase).

Although some have argued that apposition is a relation in its own right (Meyer (1991)) it is clear that the above apposition syntax may hold various conceptual relations. The appositive phrase may be a definition, a hypernym, a description of the components of the concept, a statement of the material it is made from, and so on. Thus it is not possible at this stage to determine the relation present (if any). KEP does not therefore use separate sets of apposition triggers for each relation type extracted. Although the phrase within the commas usually says something about the preceding concept, it is not possible to label that statement as a definition etc. This is a problem which is considered in the following chapter.

Not all types of apposition require separated punctuation marks (see Greenbaum and Quirk (1990)) and some of these other types may be triggered using the standard triggering mechanism. The trigger which is a comma followed by the word "or" is one example of this; the sentence *High-energy radiation with wavelengths shorter than visible light, or ultra-violet radiation, can be dangerous to the skin* defines *ultra-violet radiation* using apposition. Others may not be triggered at all by KEP. For example, in the sentence *The XL5 spaceship rocketed off its launcher* the name *XL5* is in apposition to *spaceship*. This sentence contains episodic knowledge as discussed earlier. It contains an instance of the instance relation; the specific object *XL5* is an instance of the class *spaceship*. KEP does not presently attempt to extract instances which use this syntax. However, since most taggers are capable of detecting proper

nouns, it would seem feasible that such instances could be detected by searching for proper noun /common noun collocations. Unfortunately, the pattern matcher used by KEP would not then be able to extract correctly from this sentence, since it does not use patterns of part-of-speech tags. Thus a separate function would be required; this work has been added to the "future enhancements" list.

Apposition triggering may be turned off by the user since it often results in the triggering of sentences which do not contain an instance of apposition. This facility allows the performance of KEP when triggering for apposition to be compared to the performance without such special trigger patterns.

4.6.9 Filtering of Presentational Sentences

Following the triggering stage an attempt is made to detect and hence filter out presentational sentences. Each of the four relation types has an associated presentational phrase file (see Table 6) containing phrases which indicate that the sentence is probably presentational. For example, the phrases *example given above* and *preceding examples* suggest that the sentence is talking about an exemplification given in previous text rather than in the current sentence. The author has dubbed these sentences relation references, since they are references to relations given elsewhere. Phrases used to detect relation references were collected by introspection and during developmental testing.

The use of filter phrases in this manner is somewhat simplistic. A sentence such as *The example given above is a poor one - a better example of a 3G language is 'C'* would be marked as presentational even though it is partly presentational and partly informational. The extraction of *'C'* as an example of *a 3G language* would be missed. Conversely, presentational sentences which are not relation references can pass through the filter. The sentence *It is difficult to find a good example of a sorting algorithm* would pass through the filter even though it is presentational. The only way to catch this sentence would be to add the phrase *difficult to find a good example* to the filter phrase list. This is not a sensible idea - many hundreds of similar phrases could be imagined and in practice a comprehensive list would probably not be achievable. (Note that the negative triggering stage does attempt part of this task e.g. by the use of a negative trigger phrase such as *cannot be defined*). The success rate of this filter is discussed in the relevant evaluation section in the following chapter.

4.6.10 Pattern Matching

The KEP program uses an essentially pattern-driven approach to relation extraction. After the above stages, a list of sentences is held for the relation being targeted. Pattern matching is then performed on each of these sentences. KEP does all extraction processing (for all sentences) for one relation type (e.g. definition) before moving on to the next relation type (e.g. exemplification). Where a concept has more than one extraction made for it, either of all the same relation type or of mixed relation types, then the extractions are merged at a later stage.

Pattern-matching techniques have proved successful in various robust parsing and extraction systems, such as that of those described in Chapter 2, and in the flexible parsing approach of Hayes and Mouradian (1981), in which the use of the FlexP parser for partial parsing is discussed. Hearst (1992) has also described a pattern-based system for extracting hyponym relations (having a very limited single specific syntax) from free text. What these systems have in common is the use of part-of-speech information to aid in the template matching. Where KEP differs from this approach is to perform an initial set of non-syntactic template matching operations to cut a sentence up into sections, but reserve syntactic information for the subsequent validation of each of the possible segmentations of the triggered sentence. The motivating idea behind this is that, should deeper processing eventually prove necessary, then it would be performed on small fragments of sentences rather than on whole sentences. Thus some of the problems of parsing etc (see Chapter 1) would be reduced in magnitude. Partial parsers have been created by several researchers (see e.g. McDonald (1992), Zhu and Shadbolt (1995), Hayes and Mouradian (1981), Burstein and Kaplan (1994)) so this is not an unreasonable approach.

The KEP program looks for single-sentence relations, although endophoric concepts are sometimes identified. The triggered sentence is segmented in a number of ways according to a tokenisation string generated combinatorially. This process is best explained using an example. Consider the sentence *An example of a 3G language is PASCAL.* This sentence contains an instance of the *exemplification* relation. Specifically, the concept is *3G language* and the example of it is *PASCAL.* It is the pattern matcher's task to perform the extraction which gives this answer. The explanation starts at the point where the sentence has been triggered as likely to contain an instance of exemplification, is not presentational, and with the knowledge that *3G language* is a technical term within the text. However, the latter piece of knowledge does not take part in the initial pattern matching stage, being reserved for fragment validation, as will be explained shortly.

4.6.10.1 Sentence Tokenisation

The first stage of the pattern matching process is tokenisation of the sentence. The sentence is reduced to a string of single characters by the replacement of words, groups of words, and punctuation by token characters.

Each relation type in KEP has an associated list of token/phrase pairs. The tokens are single-characters which are used to stand in for the phrase. For example, the token **e** could stand for the phrase *An example of,* and the token = for the phrase *is.* The list of token/phrase pairs is held in an external file. Phrases may be several words long, a single word, or a punctuation mark. Phrases including punctuation marks are also allowed. The pattern matcher regards punctuation marks as separate words in the sentence. Punctuation marks must however be tokenised by themselves, e.g. token=! and phrase=! . In addition, the numerals 0 to 9 may not be used as tokens, for a reason which will become clear shortly.

Some examples of token/phrase pairs are given in Table 8. Note that one token (such as **e**) may take part in several token/phrase pairs, with different phrases in each case. Thus the token **e** may stand for the phrases *example*, *an example of*, *An example of* etc.

Token	Phrase
e	example
I	instance
e	Examples of
I	Instances of
e	an example of
e	An example of
I	an instance of
I	An instance of
s	such as
l	like
f	for example
f	for instance
+	and
.	.
=	is
=	are

Table 8. Sample list of exemplification tokens

Sentence tokenisation also makes use of a special token, the X-token. This token's phrase is not fixed - it can be any group of words and/or punctuation marks. X matches anything.

The tokenisation algorithm accepts the input sentence as a string of words separated by spaces (all punctuation marks being regarded as words). Part-of-speech tags do not play any part in the tokenisation process, so the untagged sentence arrays are used. The algorithm reduces the sentence to a string of tokens, or rather, it reduces the sentence to *all possible* strings of tokens. This is done by using the token/phrase pairs, and the X token (whose associated phrase is any group of words). For example, one tokenisation of the sentence *An example of a 3G language is PASCAL* . , using the above token/phrase pairs, is **eX=X.** . To obtain this tokenisation, the phrase *An example of* was matched to the token **e**, the phrase *a 3G language* was matched to the token **X**, the phrase *is* was matched to the token =, the phrase *PASCAL* was matched to the token **X**, and the phrase . was matched to the token . . Note that the terminating full-stop takes part in the tokenisation process as a word in its own right.

The tokenisation process starts by listing all the tokens found in the sentence. In our example *An example of a 3G language is PASCAL.* there are four tokens: **e** for *An example of*, **e** for *example*, = for *is*, and **.** for a full stop. The tokenisation method firstly attempts the tokenisation using zero token/phrase pairs from the list. This gives the tokenisation **X** for any sentence (i.e. X matches the entire sentence). This is a

trivial tokenisation which is discarded. The tokeniser next attempts to tokenise the sentence using one token/pair, attempting this for every token/pair in the list. Given the token/pairs in Table 8 above, this would give the following tokenisations for the test sentence:

eX XeX X. X=X

The tokeniser next attempts to create tokenisations using two token/pairs at a time from the table. Certain pairs of token/pairs cannot be used to do this within the same tokenisation, i.e. where one phrase is part of the other phrase. This results in the following set of tokenisations of the test sentence:

eX. XeX. eX=X X=X. XeX=X

The tokeniser next uses three token/pairs at a time. This gives the following allowed tokenisations:

eX=X. XeX=X.

Note that strings such as XX, XXX etc do not occur in any tokenisation - these are always reduced to X. The process of tokenisation continues with increasing numbers of token/pairs used at a time, until no more are possible (i.e. until the number to be used at one time exceeds the number of recognised phrases in the sentence). The full list of tokenisations for the sentence is then stored. This list may well include duplicated tokenisation strings, because a single token such as **e** may be mapped to more than one phrase. However, stored alongside each tokenisation are the sentence fragments which gave rise to that tokenisation. Thus the tokeniser can distinguish between seemingly identical tokenisation strings. The sentence fragments are used in the relation extraction, as described in the following subsection.

The full list of tokenisations for the test sentence *An example of a 3G language is PASCAL* . (using the token/phrase pairs in Table 8) is given below:

eX XeX X. X=X eX. XeX. eX=X X=X. XeX=X eX=X. XeX=X.

The tokeniser has effectively cut the sentence up in all possible ways using a set of known phrases. Within the set of tokenisations there may be one or more in which X-tokens cover the concept to be elucidated, and the actual elucidation (exemplification). The punctuation and the relation-specific token/phrases act as sentence section boundaries and span markers. The task of the template pattern-matcher is to identify the cases of interest out of the tokenisations and to thereby extract text fragments which may act as concepts and examples (etc). This is described in the following section.

Because all possible combinations of token/phrase pairs are used in the tokenisation stage, there is an approximately exponential rise in the number of tokenisations to be attempted with an increase in the number of phrases from token/phrase pairs (p) actually present in the sentence. Table 9 shows how many tokenisations must be attempted for values of p up to the maximum of 16 used by the tokeniser (the trivial tokenisation X for any sentence is not counted). The tokenisation mechanism stores only those tokenisations which could feasibly give rise to extractions, all others being discarded. This minimizes memory requirements (since each tokenisation requires a fixed amount of memory capable of holding all the fragments of an 800-character sentence, plus other data such as the tokenisation itself and miscellaneous flags).

No. of distinct token phrases present in sentence (p)	Total no. of potential tokenisations arising because p phrases were found.
0	0
1	1
2	3
3	7
4	15
5	31
6	63
7	127
8	255
9	511
10	1,023
11	2,047
12	4,095
13	8,191
14	16,383
15	32,767
16	65,535

Table 9. Numbers of tokenisations needed for p tokens present in sentence

The maximum value of p is restricted not by memory but by running time. For the maximum p-value of 16 tokens, running time for one sentence on the development machine varied between 10 and 20 minutes depending upon the precise amount of processing required for each individual tokenisation. Fortunately, in most cases the p = 16 case did not arise, and the tokenisation times were measured in seconds rather than minutes. However, the exponential nature of the tokenisation method does have implications for token file design, and this is discussed later. (In brief, the number of token phrases should be kept as small as possible, so that the p = 16 limit is not reached or breached.)

In the case of our example, p = 4 and so potentially there are 15 tokenisations. However, 4 of these would have involved overlaps between the two phrases *An example of* and *example*, which both map to token **e**. (These were the cases where two tokens **ee**, three tokens **ee.** and **ee=** and four tokens **ee.=** would be tried at the same time, where the two **e**'s are the different e-tokens.) Such tokenisations are not

attempted by the tokeniser, since they are not capable of sectioning the sentence. This leaves the 11 tokenisations listed above. Discarded tokenisations also include those not ending in one of the three punctuation characters . ? ! since nearly all sentences end with one of these and since concepts or their elucidations do not. Thus, out of the 11 tokenisations given above, the following 6 are left:

X. **eX.** **XeX.** **X=X.** **eX=X.** **XeX=X.**

In addition, those tokenisations not having at least two X-tokens in them are discarded (i.e. those less than four tokens in length). This is because it is the X tokens that give rise to the concept and its elucidation, and so at least two X tokens are needed for any ultimately successful extraction. This leaves the following 4 tokenisations:

XeX. **X=X.** **eX=X.** **XeX=X.**

Tokenisations which pass these tests are labelled as good and stored.

4.6.10.2 Template Pattern Matching

Each relation type has a file of templates against which tokenisations are to be matched. Templates are similar in form to the tokenisations described above except that (1) they always end with a sentence-terminating punctuation mark, and (2) instead of containing X-tokens they contain the token C and the tokens 0, 1, 2 ...9. A sample list of templates for the exemplification relation is given in Table 10. (In reality this list is much longer, since it needs to capture all of the patterns for expressing exemplification. For example, the full list for the definition relation is given in Appendix C). The meaning of these templates will become clear shortly.

The pattern matcher performs a match between all tokenisations which pass through the "good tokenisation" filter (see above) and the templates in the template file. Each "good" tokenisation is matched against every template in the template file. It may match more than one template for reasons described shortly. Up to three matches are stored for any one tokenisation.

Template
eC=0.
eC=0+1.
X,eC=0.
0=eC,X.
0,1+2=eC.

Table 10. Some exemplification templates

A "match" occurs when the tokenisation matches the template character by character, except where the tokenisation has an X token. Here, the X is allowed to match either the C token in the template or any of

the 0 - 9 tokens. Returning to the example sentence, note that one template in the table takes the form **eC=0.** . This is deemed to match the tokenisation eX=X. , where the first X corresponds to C in the template, and the second to 0 in the template.

The C in the template stands for Concept, and the 0 in the template stands for the first example (up to ten, 0 through 9, may be present in the template). It was mentioned previously that the tokenisation process associates a word string with each token. For the tokenisation eX=X. , the associated word strings and corresponding template tokens are given in Table 11. When a tokenisation-template match occurs, the phrase associated with the C token is marked as the concept and the phrase associated with the 0 (etc) token is marked as the exemplification (in this case). Any initial indefinite articles (a, an, A, An) are then stripped from the concept. This gives the extraction **Concept:** *3G language* **Example_0:** *PASCAL.*

Tokenisation	Word string	Template token
e	An example of	e
X	a 3G language	C
=	is	=
X	PASCAL	0
.	.	.

Table 11. Example of a KEP tokenisation

This extraction is at this stage merely a candidate extraction. Because one single tokenisation may match up to three templates, and because there may be many "good" tokenisations all to be matched against the list of templates, there may be several candidate extractions. Some of these may differ only slightly but others may be completely different. Therefore validation and amalgamation methods are required to filter out the good extractions from the set of candidate extractions.

It was stated above that one tokenisation may match up to three templates. It is in fact theoretically possible that one tokenisation could match more than three templates (for relationships with multi-part elucidations using the 0 - 9 tokens). The 3-match limit was imposed mainly to limit processing, but also because in practice it is extremely rare for a tokenisation to match more than two templates (it usually matches just one). The 2-match case arises where the tokenisation matches not only the template *designed* to match it, but also one designed to match another tokenisation of the same length but with concept and elucidation in reversed position. For example, consider the tokenisation **eX=X.** which is designed to match a sentence such as *An example of a 3G language is PASCAL.* , which would have the template **eC=0.** . The problem here is that there might also be a template of the form **e0=C.** which corresponds to a sentence such as *As an example PASCAL is a 3G language.* (where in this case the token **e** stands for *As an example*). Both templates match the tokenisation **eX=X.** . Since the pattern matcher would and should detect both matches (because the pattern matcher has no way of knowing

which is the "correct" match) then both matches need to be stored for further processing. This situation is however rare; it is also avoidable by the judicious choice of token/phrase pairs i.e. by avoiding the re-use of tokens such as **e** for phrases which occur in quite different sentence structures.

For successful KEP operation the token and template files for each relation type must be populated in a consistent and comprehensive manner. *Consistency* is required in that there must be no tokens present in any template in a template file which are not defined in the corresponding token file. (KEP detects this situation if it arises and issues a warning message to the user, but this does not stop the run.) Furthermore, the set of tokens must have been created with the templates in mind, and vice versa, since the degree of "detail" to be matched is completely under the designer's control. *Comprehensiveness* is necessary because if a template is missing then extractions will be missed. The construction of the token and template files is described in the following chapter, since this process is best considered alongside evaluation.

4.6.11 Fragment Validation

The text fragments associated with the tokenisation-template matches arising above are tested for validity, and where these fail they are rejected. If all fail, no extraction will result. If only one passes, then this becomes the extraction reported. If more than one passes, they are amalgamated using a procedure described in a following section. The aim is to label just one extraction as the correct extraction, or construct such an extraction from the candidate extractions.

The text fragments are either supposed to be the concept or the concept's example (definition, partition, hypernym). Fragments from the C token are validated as concepts, and fragments from the 0 (1, 2, ...9) token are validated as examples (etc).

4.6.11.1 Validation as Technical Terms

For a sentence fragment which might be a valid concept, one way of validating it is to see if it is a technical term in the source document's domain. If it is, then it is likely that it is a valid concept. KEP validates concept fragments by checking them against the TT list previously constructed. This involves handling letter case differences (since a fragment starting a sentence may start with a capital letter, whereas one in the body of the sentence may not) and it also necessitates reducing any plural fragments to their singular forms before making a string comparison (since the fragment may be *industrial complexes* and the TT *industrial complex*, say).

TT validation is a most effective way of validating a concept fragment, but does occasionally lose good extractions because the TT is not stored. In addition to TTs, concepts which are recognised as acronyms having expansions are also marked as valid.

4.6.11.2 Tag Pattern Methods

It was originally envisaged that KEP would validate sentence fragments by the compilation of lists of tag patterns (for the concept parts of the extractions, and the elucidation parts). Technical term methods as above might mean this is not necessary for the concept parts, but the elucidation fragments need not be technical terms. Very often they are longer sections of sentences, especially for the definition relation. It would not be practicable to list all possible tag patterns for such segments. Some form of partial parsing is probably required, e.g. to detect fragments which are noun phrases (NP) (e.g. *small south american rodent*) or NP plus further elucidation such as by a *which* phrase (e.g. *small south american rodent which is mainly found in the rainforest*). Although KEP detects elucidation fragments which are TTs or acronyms, it does not reject fragments which are neither. All fragments are currently marked as valid. This is a ripe area for future research. (See also Chapter 6 for further discussions on parsing elucidation fragments.)

4.6.12 Candidate Extraction Amalgamation

After candidate fragment validation there may be zero, one, or more validated candidates i.e. extraction candidates where both the concept part and the elucidation part (definition, hypernym etc) are marked as valid. If there are no validated candidates, the extraction attempt has failed to make an extraction and processing has finished for this sentence for the relation type being searched for. If there is one validated candidate then this is presented as the successful extraction and ultimately printed to the output files.

For two or more validated extraction candidates a decision or an amalgamation process is necessary, since KEP is only able to perform one extraction for a given relation type for each sentence. Where all candidate extractions are identical in both parts, then any of them will do and so the first is presented as the extraction. This often occurs because for the concept part, initial indefinite articles are always stripped off. Where there are two or more candidate extractions and the concept parts are the same but the elucidation parts are different, then decision making/amalgamation is required on the elucidation parts. Where concepts differ between candidates and elucidations are the same, or where concepts and elucidations are a mixture, the decision making /amalgamation process can become complex, as is demonstrated below.

Where there are two extraction candidates having different concepts, the first candidate extraction is picked as the correct one. Where there are two extraction candidates having identical concepts but with some similarity in their elucidation text fragments (more than 30% similar), then the two elucidation parts are passed to a longest common substring (LCS) function. This returns the common core of the two fragments, and since the fragments usually differ only at the ends (where there are added or missing words) then this method usually returns the correct elucidation text. Where there are two extraction

candidates having identical concepts but differing greatly in their elucidation text fragments then the first extraction candidate is returned.

Note that the LCS function can only be used when it is known that the strings involved are similar, since the LCS of two widely different strings is usually very short or even non-existent. For example, there is no LCS for *large brown dog* and *small wild pig*. The LCS function developed for KEP works on a word-atomic basis (not on a character-atomic basis) so that the LCS of *large brown dog* and *large brownish dog* would be *large*, not 'large brown'. This LCS method is ideally suited for sentence-fragment pairs such as the following (where the LCS returned is given in bold on the third line):

```
  type of large brown dog found in the Andes , but
a type of large brown dog found in the Andes
  type of large brown dog found in the Andes
```

For three or more validated extraction candidates, the process becomes more difficult since it is possible to have many combinations of identical/different concept and elucidation parts. The approach taken actually applies to any number of validation candidates (including 1 or 2) and is to count each distinct concept and use the one with the highest count (or, in the case of a tie, the first group of concepts with the joint highest count). Then only elucidations from that group of candidates are considered. The largest group of identical or near-identical elucidations is chosen and the LCS from this returned as the elucidation extracted.

Despite the potential complexities of the above process, in practice it is mostly the case that only zero, one or two valid extraction candidates are put forward. Thus in the majority of cases the more difficult combinations within the amalgamation process are not explored. The amalgamation process is an important area for future improvement since the "correct" extraction is usually present within the amalgamation candidate set when the set is not actually empty. It is thus especially disappointing when the amalgamation procedure fails to find it, as occurs occasionally.

4.6.13 Noun Number Resolution

The techniques described in the above sections mention that it is often necessary to compare plural nouns with singular nouns, to check if the former is in fact the plural form of the latter. KEP contains a novel function developed specially to do this (the sing() function). This function accepts a word which is known to be a plural noun, and returns the singular form. Less than one word in every thousand is incorrectly singularised, a very high success rate of better than 99.9%. Furthermore, this is achieved without the use of a machine readable dictionary (MRD). The approach is fully detailed in Bowden, Halstead and Rose (1996c), but the major points will be outlined below. Note that the absence of the MRD means that KEP is more easily able to remain domain dependent. Not only is a lexicon of specialist domain terms not needed (an important factor for domains which have very large specialist

vocabularies, such as medicine), but also neologisms can be handled correctly (an important factor for domains which are fast moving, such as information technology).

Various factors complicate the at-first seemingly straightforward task of finding a singular form for a plural noun:

- homonymic plurals e.g. *bases* (*basis*, *base*)
- alternative plurals e.g. *pennies*, *pence* (*penny*)
- multiple singular forms e.g. *axes* (*ax*, *axe*)
- mixed homonymic/multiple e.g. *axes* (*ax*, *axe* and *axis*)
- multiple plurals e.g. *formulas*, *formulae* (*formula*)
- no change e.g. *series* (*series*)
- central vowel changes e.g. *feet* (*foot*)
- no meaningful singular e.g. *trousers* (*?trouser*)
- completely different word e.g. *people* (*person*)
- genuine oddities e.g. *dice* (*die*)
- hyphenated plurals e.g. *men-of-war* (*man-of-war*) [not yet handled]
- semantically-determined senses e.g. *mechanics* (people or discipline?)

In order to handle these, as well as the more usual "exceptions", the sing() function is rule-based, using also lists of exceptions to rules. It is essentially structured as a tree of *if-then-else* structures, where the *if* parts test the final characters of the input plural noun. For example, a rule such as **remove ies and add y** can handle words such as *cities*, but there are exceptions, such as with *pies*. In fact, up to seven levels of if-then-else structuring are used, arranged in such a way that the lists of exception words are kept as short as possible. Some of the exception lists needed are shown in Table 12.

aeries	koppies	
bogies	lassies	
calories	lies	
collies	mounties	
coolies	magpies	
cookies	movies	
corries	neckties	
cowries	verlies	
darkies	pies	*Some exceptions to the rule* **ies -> y**
dies	pixies	
dixies	quickies	
eyries	reveries	
falsies	sorties	
gillies	talkies	
genies	ties	
goalies	toughies	
indies	zombies	

Figure 17. sing() exception list 002

It is interesting to note that only about 350 exception words are needed in order to achieve the 99.9% success rate. The longest list is list 005, which contains words whose pronunciation gives clues as to how

to form the singular; clearly this information is arbitrary and therefore not available to the sing() function (e.g. the ***u*** vowel sounds in *buses*, *fuses* and *octopuses* are pronounced in three different ways by most native English speakers, although there are more than three ways of pronouncing them *in toto* if regional dialects are considered).

An example of one of the lists, list 002, is given in Figure 17. The simple structure of the function, together with the shortness of the exception lists, means that the sing() function is fast. This is an important factor in a program that may need to make thousands of such calls during a run.

EXCEPTION LIST NUMBER	DESCRIPTION OF WORDS	EXAMPLES
001	end ies, no change	series
002	end ies, lose s	pies, movies
003	end sses, no change	molasses
004	end sses, lose s	crevasses, posses
005	end ses (not sses), lose es	bonuses, gases
006	end ses (not sses), ses to sis	analyses, oases, theses
007	words in 006 with other sing.	bases
008	end xes, lose s	axes
009	words in 008, also xes to xis	axes
010	end ces, ces to x	appendices, matrices
011	end ices, ices to ex	indices, vertices
012	end ches, lose s	tranches
013	end ves, ves to f	calves, leaves
014	end oes, lose es	potatoes, cargoes
015	end ics, no change	electronics
016	end s (not ies etc), no change	trousers, tongs
017	the word corpora	corpora
018	end ice, ice to ouse	mice, lice
019	the word pence	pence
020	the word dice	dice
021	the word geese	geese
022	the word people	people
023	end ia, ia to ium	bacteria, media
024	end ves, ves to fe	knives, wives
025	end ies, ies to ey	monies
026	the word feet	feet
027	the word teeth	teeth
028	end zes, lose es	topazes, waltzes
029	the word brethren	brethren
030	end hes, no change	clothes
031	end ies, lose es	chillies
032	end la/ta/da, a to on	automata
033	end sses, lose ses	gasses

Table 12. Exception lists in the sing() function

4.6.14 Dealing with Anaphora

It is sometimes the case that fragments from the extraction stage are found to be anaphoric (pointing to previous text), cataphoric (pointing forwards in the text) and even semi-exophoric (pointing out of the text to some other entity on the page, such as a figure or a table). Simple anaphoric fragments include *this*, *these* etc. More complex constructs include phrases like *such devices*, *this type of <noun>*, *given in Figure. 5.7* etc. The simpler demonstratives are validated before syntax-checking, but the more complex phrases are handled in a function designed specifically to detect endophoric links.

Links which point to tables and figures within the text cannot be simply resolved, and where such a pointer is detected the output is therefore set to text such as *<given in an accompanying diagram>*. The set of phrases indicating such links is small and so is hardcoded into the detection function.

Links within the text proper do at least terminate on other phrases, and so some attempt could be made to follow them back (or forward) to the relevant concept (or example etc). For phrases like *such a device* in the sentence *An example of such a device is the laser printer* it is likely that the linked concept lies in the immediately preceding sentence, usually as the head. This is an area within KEP which is presently being worked on; currently the detection code is being implemented (using a file to hold trigger text patterns such as those given above). However, it is already evident that only simple target concepts will be extractable. Anaphoric links may point back to intangible concepts described by the whole of a preceding paragraph (or even larger textual unit). No simple syntax-based extraction method would ever succeed in resolving such links; systems incorporating semantic and pragmatic knowledge will be required. (For a discussion on how humans may form complex concepts whilst reading through a text see Kieras (1982). This paper concentrates on the way in which the reader finds the important topics/sentences in a text, so it is also relevant to the discipline of automatic text summarisation, discussed later.)

For further discussions on the anaphoric resolution function, together with ideas concerning more difficult types of anaphora (such as first-mention definite noun phrase anaphora) the reader is referred to Bowden, Halstead and Rose (1996d).

4.6.15 Merging of Extractions by Concept

The output shown in Figure 9 on page 84 groups four extractions (a definition, an exemplification, a hypernym and a partition) against one concept (*sort routine*). However, this was not how KEP originally found the extractions; they were found at separate times because KEP processes one relation type at a time, going through the whole text for each relation type. Thus a merging function is required to produce the output as shown. The merging function detects identical concepts and ensures that all extractions for them are grouped together.

Merging of concepts takes place only within a single run of KEP. Since the program does not at present maintain a memory of past runs (in the form of a semantic net KB) then merging is not required in such a structure. However, this problem will arise within curriculum graphs of the HypeLab/HyperTutor system if future work succeeds in providing a reliable interface. The updating of existing semantic nets is not a simple task (see e.g. comments by Hearst (1992)) but this work rightly resides within HypeLab/HyperTutor, and so unless it is later decided that KEP should maintain its own KB memory, it is not of concern here.

Merged extractions are stored in a linked list based structure configured as a spinal LL in which each element contains a unique extracted concept. Each element in this spinal LL may have up to four side LLs to hold lists of definitions, examples, parts and parent classes. Thus the entire set of extractions from the input text is held in a single data structure from which output may be obtained for the short, long and KEN output formats.

4.6.16 Construction of Output Files

Short output contains only a heading line and output taken directly from the spinal LL data structure, in the order present. KEN output is essentially a reformatted short output. Long output contains the same extraction data plus all processing comments and error messages (if any) as well as line and sentence structures. It also contains much statistical information regarding the extractions made or attempted. It is essentially a diagnostic and recording tool which shows in detail the processing performed. The file is usually very large (more than twice the size of the input text). The size of the long output given in Appendix D illustrates this point. Long output is rarely printed, since it is easier to read/search an online version.

Glossary output is also produced from the spinal LL but in a reformatted form that orders glossary entries alphabetically on the first column present (acronym or term). A separate LL structure is used to hold the glossary, each LL element holding one glossary entry. The glossary-maker function also builds cross-reference information between entries. These cross references have been kept deliberately few in order to avoid a plethora of mostly unhelpful links. For example, links between terms (middle column) and their hyponym/hypernym terms are avoided, since they can result in all the terms involved listing all the related terms. Instead, three types of cross reference are made: between terms used in the text of the third column and those terms' own glossary entries, between acronyms used in the text of the third column and the entries for their expansions, and between acronyms which are part of terms in the middle column and the entries for their expansions. Thus for example if the 3rd-column text mentions GIS, or if GIS forms part of the middle column, there will be a link of the form *SEE ALSO geographical information systems* if this has its own glossary entry elsewhere. This helps the reader to understand an unfamiliar acronym whilst reading a glossary entry.

Term summaries (introduced in section 4.6.6) are not held in the spinal LL data structure and are constructed after TTs and acronyms have been collected. This is essentially a search operation on the sentence arrays, although TTs must be searched for in both plural and singular forms, and with or without term-initial capitals. The process also reads sentence numbers to group the output into blocks and to fill inter-block gaps less than two sentences long.

4.6.17 Evaluation Considerations

It is a fact that a full extraction run, on a large text such as 'B1G', for all four relation types, and with apposition triggering and *is a* triggering switched on, may take many hours. This important issue will be discussed later. It can be a problem for practical reasons, such as non-availability of the computer for long periods (e.g. on some systems essential book-keeping programs are run overnight, requiring 100% of the computer's resources). However, since each of the four relation types is processed separately, it is possible to run KEP four separate times, once for each relation type. During evaluation (see Chapter 5), precision and recall can then be found for each in turn. (It is in fact possible to run KEP for any combination of the four relation types, since the user is asked whether to run for all four or to run for some lesser combination – see Table 5.)

However, running KEP for each of the four relations in turn would duplicate the effort of finding all the TTs and acronyms in each run, i.e. the process would have to be done four times, three of them unnecessarily. Since TT/acronym extraction itself may take several hours if full text look-ahead is used, this represents much wasted time. KEP has therefore been provided with the facility to store all internal term and acronym data structures to disk (files *ttdisc.out* and *acdisc.out* – see Table 6; these files were also useful during system development because they are human-readable). Upon running KEP, the user is queried as to whether the text being processed is the same text as for the last run. If so, the user may choose to restore the last run's TT and acronym data from disk. This leaves KEP in a position identical to the one it would have been in had it extracted TTs and acronyms from scratch; the only difference is that this point in the processing is reached several hours earlier. Clearly this is a useful feature during multiple extraction runs on the same file (e.g. 'B1G'). This method was used in the evaluations described in Chapter 5. (Note: In order to construct a full 4-relation glossary, KEP must be run at least once for all four relations together. Even here, however, TTs/acronyms may be restored from disk from the original run.)

4.7 Concluding Remarks

KEP is a large program (over 22,000 lines of 'C' code, split over eight source files). Its central feature is a novel pattern-matching facility which allows the segmentation of sentences around punctuation and special phrases. This pattern matcher is unlike those developed for domain specific NLP systems since it contains no domain specific phrases. Instead, *relation* specific phrases are utilised. Together with a

modified and enhanced form of an existing term acquisition method, and a completely novel acronym extractor, KEP attempts to extract instances of the definition, exemplification, partition and hypernym conceptual relations. KEP is designed to process explanatory texts, but these texts may be about any topic. Input texts must first be part-of-speech tagged using a commonly available tagger such as CLAWS. KEP has been designed to be robust; it is not stopped by irregularities in the input text and is capable of processing very large texts comprising thousands of lines or sentences. KEP requires no external resources such as machine readable dictionaries or thesauri, because a shallow approach is used which utilises lexical and syntactic information available within the input text, in-built knowledge of how plural nouns are formed in English, and lists of relation-specific phrases provided during development.

In the next chapter the performance of KEP is evaluated. In addition the methods used to acquire lists of relation triggers and phrases are described. In the final chapter, the limitations of the methods are considered and possible future enhancements discussed.

5. Evaluation

5.1 Introduction

This chapter reports upon evaluation and testing of the KEP system. The results of the evaluations are discussed so that the successes and limitations of the methods used by KEP may be considered.

The primary evaluations described in this chapter relate to KEP's ability to extract instances of the four conceptual relation types attempted (definition, exemplification, partition and hypernymy). Contributing to these evaluations are secondary evaluations of some of the elements of the above, such as how well KEP can extract acronyms or spot technical terms. At a third and lower level are evaluations of commonly-used functions such as KEP's plural noun singulariser.

In addition to evaluation, this chapter also contains a description of the methods used to populate the token and pattern files used in the central pattern-matching mechanism novel to KEP. Since these were created in a systematic way coupled closely to evaluation, this chapter is the natural place to do this.

5.2 Precision and Recall

IR (information retrieval), IE, MU and KE systems are properly evaluated against the metrics of precision and recall. These two measures may mean slightly different things in different circumstances, but the essential concepts remain the same. Both relate to the "answers" given by the system, and whether they are "correct". It is the nature of the answers which vary from application to application - an "answer" may be an extracted fact, or an identification of some linguistic structure, or the highlighting of a relevant part of a text etc. Correctness is likewise application dependent; it is necessary to state what one means by "correct" when looking at a certain type of answer. Although deciding whether a particular answer is correct can often be difficult, correctness has to be coerced to be a *boolean* variable - an answer must be either correct or incorrect for the calculation of precision.

The precision of a system is a measure of what fraction of the given answers are correct. It can be calculated as:

precision = (no. of correct answers / no. of answers presented) * 100

To take an example, consider KEP's acronym extractor. A precision figure can be calculated for those cases where an acronym has been found and an expansion for it presented. In this case, the "answer" is "an acronym from the text together with what it stands for". The "correct" value is either true or false; *true* means that both parts of the extraction are correct in all details i.e. the extraction is indeed of an

acronym, and that the expansion given is indeed the right one as mentioned in the text. *False* means that part of the extraction is wrong. Thus if the acronym detector found 84 acronym/expansion pairs and 80 of these were deemed to be correct, the precision would be (80/84)*100 = 95% (to the nearest percent). It is important not to use too many decimal places - for raw data counts of the order of 100 it would not make sense to give the precision to more than one percent. For raw data counts of around 1,000 then one place of decimals would be acceptable (i.e. to say that KEP has a 99.9% precision for finding the singular form of a plural noun implies that at least a thousand cases have been tried).

Note the importance of stating what is being measured. One could for example generate precision figures for just *identifying* acronyms in the text; in this case the correctness question might be a yes-no decision as to whether the identified word really is an acronym or not (irrespective of whether it had an expansion in the text).

In the above example, incorrect answers may arise for a number of reasons. The expansion may be present in the text but wrongly extracted. The expansion may not be present in the text, but one was extracted anyway (obviously incorrectly). The acronym identified might not have been an acronym at all, so the extraction would have been wrong whether or not an expansion was presented. The first of these three cases is a simple wrong answer, but the last two are examples of **false positives**. A false positive is an incorrect answer obtained because an answer was found where none actually existed. It is usual to state the false positive rate separately since it may be a separate variable that can be reduced without altering the rate of ordinary wrong answers. Precision figures which do not incorporate false positive rates are higher than those that do, and so one must always state whether a given precision figure includes it. For example, if three of the incorrect answers in the above example were due to false positives, then by removing these from the precision calculation the precision rate jumps to (80/81)*100 = 99%. Clearly, this can be misleading. All precision figures reported below include false positives i.e. take the most pessimistic figure.

Precision says nothing about how good a system is at finding all the "answers" in a text. A program might have 99% precision because it gets 99 out of every 100 answers right, but still only get 20% of the answers that are there to be found. This is where the recall metric is useful:

recall = (no. of correct answers presented / no. of answers available in the text) * 100

For example, if the acronym detector found 80 correct acronym/expansion pairs from a text which actually held 85, then the recall rate would be (80/85)*100 = 94%. In practice recall measures are usually lower (sometimes substantially lower) than precision measures. It is also often the case in real systems that recall and precision figures are linked inversely - tweaking a system to increase precision often causes a drop in recall, and vice versa. This is to be expected; those instances of items not found by

initial methods are likely to be the more difficult ones to extract. Thus, any improved method aimed at getting them is attempting to obtain the more difficult instances. Thus there is a higher likelihood of the new method making mistakes in these cases. The aim of course is to create a system having both high recall and high precision.

Thus the recall metric shows how *comprehensive* a KE/IE system is, and the precision metric how *accurate* it is with what it does find.[10] Recall is often the more difficult to calculate simply because it can be difficult to count the number of items available for extraction. For example, it is not easy to decide whether a definition is indeed present in a sentence, and so it is not easy to provide the denominator for the recall metric. This is why tests such as Skuce et al.'s definition-presence test (see page 73) are important.

The precision and recall figures which are given in the following evaluations adhere to the philosophy of being pessimistic. Wherever there is some doubt the decision is made so as to lower the precision and recall figure arising, not raise it. Where such decisions occur they are discussed in the accompanying text.

5.3 KEP Function Evaluations

The evaluations which follow have been arranged in the order of KEP processing as described in the previous chapter. Thus the prime evaluation, that of KEP's performance as an extractor of conceptual relation instances, appears towards the end of the chapter (section 5.3.6, page 153).

5.3.1 Sentence Delimitation

Sentence-end detection is not a high priority area for KEP, but it must be good enough to allow the program to function correctly. Thus evaluation need not be exact - it need only say whether an acceptable rate has been achieved. Manual checking of BNC texts (which may be tens of thousands of lines long) against KEP's sentence structure would be a laborious task and is not in fact necessary to achieve the above goal. Instead, a comparison between the number of sentences thought to exist in the text by KEP and by the CLAWS tagger is sufficient. Clearly this will not give an exact figure, since multiple errors may cancel each other out (i.e. sentences may be incorrectly concatenated, or incorrectly split). However it does indicate whether the two systems give comparable counts. The number of mutually-cancelling errors cannot in any case exceed the number of instances where two sentences are incorrectly concatenated, which in the vast majority of cases happens where a heading is prepended onto a following sentence. Since headings form only a small percentage of the sentences, it follows that the

[10]Some researchers present an average of recall and precision which they call *accuracy*; another measure is the F-measure, F = 2PR/(P+R)

mutual-cancellation rate must be low. Thus this method does give a good comparison of KEP's and CLAWS' sentence delineation decisions.

The number of sentences found by KEP is the number of the highest sentence number, plus one (because the first sentence is labelled as sentence 0, in the 'C'/UNIX tradition). This count includes headings where they have been identified. The corresponding count for a BNC text is obtained by examining the tagging declaration element of the BNC text's encoding description, part of the file header. An element of the form `<tagUsage gi=s occurs=2411>` indicates that the text contains 2,411 s-tags i.e. 2,411 sentences. The Users Reference Guide for the British National Corpus (Version 1.0) Burnard (1995) describes the <s> tag as being for a "sentence-like linguistic segment". This includes heading lines as with KEP. Thus the sentence count contained in the BNC text header is comparable with that described for KEP above.

Three large 'informative' BNC files were used to compare BNC and KEP sentence counts s(BNC) and s(KEP). The results are given in Table 13. In each case the accuracy (s(KEP) / s(BNC)) * 100 was calculated. (This figure is not a precision metric because the value of s(KEP) may be higher or lower than the s(BNC) count.) The figure gives an indication of the closeness of the two counts and shows that KEP achieves a rate within a 10% band.

BNC text name	**s(BNC)**	**s(KEP)**	**((s(KEP)/s(BNC))*100**
B1G	1650	1513	92
EAK	1346	1357	101
FTE	1514	1407	93

Table 13. BNC/KEP sentence count comparisons

The rates given above are pessimistic in that they do not make any allowance for errors created by the *conbnc* pre-processor program, which very occasionally splits or joins sentences due to errors during tag stripping and re-attachment. The counts are close in all three cases, but each file has its individual characteristics. For example, the text FTE has two domain-specific problems. The first of these is the naming of bacteria. Latin bacterial names are often written in short form like *E. Coli* or *B. Subtilis*. Like peoples' names, these trip up the detector due to the full stop after the initial capital letter. However, one would expect this problem to cause KEP to find more sentences than BNC, not less as actually occurred. The second DS peculiarity is that this text actually contains a large number of unterminated sub-headings, which are therefore wrongly prepended onto good sentences. This explains the shortfall. Since the text contains very many words having initial capitals (or being all capitals) it is difficult to see how any mechanism based upon sentence *start* words (beginning with a capital) could help this situation without causing many more false end detections. When these two types of error are subtracted from the output, nearly all sentences are correctly identified.

It is clear that the relatively simple approach to sentence-end detection used by KEP gives satisfactory accuracy rates, and since this specific area is not of prime interest to KEP, it will not be discussed further.

5.3.2 Technical Term Acquisition

5.3.2.1 TT Acquisition Performance

The calculation of precision and recall for KEP's technical term (TT) finder is prone to uncertainty due to the subjective nature of the task. Precision is the percentage of reported TTs which were indeed TTs, and recall is the percentage of TTs in the text which were reported as TTs. In both cases it is necessary to identify the TTs in the source. This is the problem. To reiterate a previous example: if *map error* and *map scale* are terms in a text from the cartography domain, then is *map* ? In a text on programming, *iteration* and *for loop* may be terms, but what about *looping* ? Can a phrase such as *serious error* ever be a TT? (What if the latter were part of a text on error calculation which classified errors as one of simple, intermediate, and serious?)

Issues such as the above must be resolved by a human decision maker, who must scan the entire text looking for all possible terms. This is a subjective process. It is also a difficult and time-consuming process, since every sentence in the text must be carefully checked, with every "possible" TT phrase identified and considered. The process of finding all the genuine TTs in the source text for the human evaluator is aided by the output files *tttest.out* and *ttnola.out*, which were described in the last chapter, and which contain lists of text fragments from which KEP derives all TTs. The process also requires the inspection of the lists of unconfirmed and duff terms given in the long output, and inspection of the source text or tagless sentences in the long output. This combination of resources gives a high degree of confidence that all 1-, 2- and 3-word terms are identified. However, tagging errors in the source may cause some real TTs to be missed completely by the sentence fragmenter. A term such as *lay by* may not be considered if *by* was tagged as a preposition. This is why manual inspection of the source is still required. It is not practicable to perform all the above for several BNC texts, especially where more than one look-ahead distance is chosen, and for this reason a single text was selected for evaluation and discussion purposes. This text, 'B1G', was chosen because it is one of the larger BNC texts, and hence allows of the possibility that many acronyms, terms and conceptual relation instances may be present, and also because it meets the criteria regarding "the right sort of text" as discussed in Section 3.2 (i.e. it is non-fiction (see Section 0), explanatory (Section 3.2.2), contains large sections of informational text (Section 3.2.3), contains much generic knowledge (Section 3.2.4), contains many facts (Section 3.2.5), is mostly declarative (Section 3.2.6), and is largely technical in nature (Section 3.2.8)). The TT results are given in Table 14.

BNC text	TT precision (a) 10-sentence (b) to end of file	TT recall (a) 10-sentence (b) to end of file	TT false positive (a) 10-sentence (b) to end of file
B1G	(a) 304/343 = 89% (b) 658/750 = 88%	(a) 304/3530 = 9% (b) 658/3530 = 19%	(a) 39/343 = 11% (b) 92/750 = 12%

Table 14. KEP TT extraction performance metrics for BNC text B1G

The precision and recall metrics have been described above; the false positive metric has been calculated as the number of terms reported by KEP which were judged not to be real TTs divided by the total number of TTs reported by KEP, multiplied by 100. Since in this instance all incorrect TTs are by definition false positives, the false positive rate is equal to 100 minus the precision. Figures are given for (a) a look-ahead distance of 10 sentences, and (b) a look-ahead distance of all the way to end of text. Errors due to bad tags in the input are included in the calculations.

The calculations do not include TTs more than three words long. The recall figure would undoubtedly fall if such terms were included, but it is thought that such terms are relatively rare if terms not involving prepositions (i.e. just nouns and adjectives) are considered. (Inspection of the BNC text B1G suggests that only a handful of such terms exist. An example of such a term from text B1G is *monte carlo simulation methodology*. Many of these terms end with *methodology*, *technique*, *approach*, *system* etc which indicate that they are really 3-word terms with an appended noun to which the 3-word term acts adjectivally.) Nkwenti-Azeh (1994) counted 4-element terms from the satellite communications domain, and for three separate corpora found that they represented 2.06%, 8.88% and 3.50% of the total term set respectively; >4-element terms were even rarer (0%, 1.97% and 0.8% respectively). If such figures can be extrapolated to other domains, this would suggest that *at most* only about a tenth of all terms are greater than three words in length.

Going on the number of terms reported by KEP, 750 - 343 = 407 extra terms were made by doing the full look-ahead. Thus roughly half of all terms re-occur locally (within a 10-sentence window). A difference operation on the two *ttnola.out* files shows that the terms added by doing a full look-ahead were general TTs relating to the overall topic of the text (such as *image processing*, *cartographic information*, *map scale* etc) or duff terms such as *cause of uncertainty*. This suggests a novel method for text topic identification: run the TT extractor with the two different window sizes, and use the added terms as above (less any duff terms) to find the overall topic of a text. This is equivalent to finding those TTs which occur throughout the document rather than concentrated in one part of it. KEP's output statistics do keep a record of where each term occurred in the text (by sentence number) and so it is possible to mark term occurrences on a graph representing the text. KEP's term summaries facility effectively draws the occurrence graph for each term (in a vertical format). Appendix B contains part of

the term summary output for BNC text B1G and demonstrates the localisation or otherwise of TTs in that text.

The precision figures given above are high. They confirm[11] Justeson and Katz's claims that (a) the method is effective, and (b) that it is rare to find TTs having the form NPN. (Approximately one term in ten reported by KEP contains a preposition (this includes false positives), and in most cases it is the word "of", e.g. as in *census of population*.) Furthermore, almost all of the terms extracted appear to be related to the subject matter of the text, which bodes well for an automatic glossary maker.

Recall figures are lower, and at first sight appear disappointing. The vast majority of the missed terms occurred only once in the text, and so defeated the shallow mechanism used to find them. Many of these missed once-only TTs are terms from other domains (i.e. they are not related to the topic of the input text) which explains why they were less likely to be repeated. Thus although the recall figures are lower than one might wish, many of the TTs not detected were in fact terms *not relevant to the input text*. This is an important point. The evaluation has asked the question "is this a TT in any domain?". Perhaps the question should have been "is this a TT in the domain of this text ?". The latter is the more relevant question for a system aiming to build a glossary for a given input text - in a text on GIS one does not need to be told that *police force* is a TT, even though it would be a useful term in a glossary derived from an article discussing crime rates, say.

Determining whether a given term is or is not relevant to the domain of a given text is a highly subjective process, and therefore any count of domain-relevant TTs must be subject to fluctuation between different human evaluators. It is also an extremely difficult process, because there are many terms which appear borderline to the text's topic[12], or which force one to reconsider what the topic of the text actually is. Having said this, an attempt was made to estimate the number of terms relevant to GIS from text B1G, and hence a re-calculation of the recall metric made. This becomes 658/1220 = 54%. Clearly this is a much higher recall figure. But perhaps even this figure is not high enough; there is a case for arguing that by definition a TT isn't relevant to the domain of a text *unless it occurs at least twice*. Of course one might argue that it is quite possible for a text to be about a topic *X* without once mentioning *X* explicitly, but this has not happened with any of the BNC texts encountered during KEP evaluation, and so appears to be very rare. (At the very least, the topic of a text may be mentioned in its main heading.) The above seems as good a definition of "relevant" as any, and at least it has the advantage of testability. Of course, using this definition, which is the approach used in the KEP TT

[11] Unlike in other branches of science, it appears rare in this field for one researcher to attempt to confirm or refute the results of another. It is difficult to say why this is, but the author believes it would do us no harm to have more confirmatory studies.

[12] Meyer and Mackintosh (1996) discuss this problem and refer to it as the problem of where to place the "side boundaries".

extractor, recall will be 100% because KEP does not fail to detect any potential term which occurs twice or more (for full look-ahead), even if the occurrences have different number (singular/plural) or capitalisation (e.g. due to one occurrence starting a sentence and another residing mid-sentence). As discussed earlier, KEP also detects the case where a term occurs once but is from that point onwards represented by an acronym. (The usual caveats about terms longer than 3 words apply.)

Some of the terms missed by the extractor were single-word terms not uncovered by the added hypernym mechanisms because they did not take part in longer terms, or because where they did take part it was always as the first word in a 2-word term (the term never occurring in two separate terms as the second word), a situation not currently handled (e.g. *traffic movement* + *traffic monitoring* giving *traffic*, where there were no terms such as *heavy traffic* and *light traffic*). Future enhancements may address the issue of better terms-within-terms resolution.

5.3.2.2 TT False Positives

Although KEP's overall performance on relevant TT detection is good, perhaps the most interesting aspect is the false positive rate. False positive TTs appear to fall into four categories. The first of these contains phrases which are right-hand-side sub-phrases of a genuine TT, but not TTs themselves. These include phrases such as *carlo method* (genuine TT is *monte carlo method*) and *processing unit* (genuine TT is *central processing unit*). The problem here is that there is a strong bond between the first pair of words in these 3-word phrases, and this cohesion is not reflected by any morphological clue such as a hyphen linking the two first words. One way of dealing with this situation would be to reject any 2-word term which always occurs as the last two words of the same 3-word term (and never by itself). However, it is conceivable that the 2-word term is indeed a TT in the text's domain, but that it just so happens that it is not used in the text under consideration. A future enhancement will test this method.

The second category of false positives includes terms made from commonly occurring adjectives and nouns, particularly those which may be used in a general manner in all domains. The above example *serious error* is one such term, as is *recent year* and *massive change*. These were introduced as "duff terms" on page 103 where it was stated that the method of finding them was poorly defined. Indeed, as has been shown above, it is not just a matter of finding such terms, since context may make them genuine TTs. It is interesting to note that such terms arise in languages other than English – Daille (1995) also discovered them, in TT extractions from French. Daille's method also utilised repeated part-of-speech patterns, so this appears to be a fundamental drawback of such methods.

KEP includes a simple function which attempts to identify 2-word duff terms using lists of first-word and second-word items. Three types of terms are rejected. The first comprises mainly temporal adjectives followed by any word, such as *aforementioned program*, *last year* and *previous quarter*. This

method is designed to catch terms which are anaphoric to a previously mentioned good term (or indeed cataphoric to a future item), i.e. to detect presentational terms. Although this type uses mostly temporal qualifiers, spatial adjectives referring to some other point in the text also occur (e.g. *accompanying diagram*). The second type comprises combinations of first word / second word, generated from two lists. Examples include *considerable importance* and *potential application*. This category is designed to catch commonly used phrases from all subject areas. To detect duff terms of the *carlo simulation* type, the third type of duff term, a list of words is maintained that cannot start a 2-word term because they must occur prepended by another given word. Thus *carlo* may not start a term, or *der* (as in *Van der Waals*) etc. (Names of people are valid terms.)

The duff-term mechanism will reject terms such as *serious error* even where context shows them to be good TTs. It also suffers from the serious problem of having to maintain lists of the common/general adjectives/nouns, an open-ended task in that these lists invariably need adding-to after each new text has thrown up more examples. Clearly this mechanism is not ultimately satisfactory and presents an interesting opportunity for future research. However, it is thought that the method does reduce the false positive rate without greatly reducing recall. Although the duff-term function started off with word lists generated by introspection, the precision and recall metrics given above were calculated before new words were added to the duff-term lists i.e. in the above tests KEP reported some duff terms as being good. Only after the run were new words added, ready for the next run. Thus the above results do not incorporate any "training" for a particular source text. However, for the purposes of illustration, after new word addition a re-run on file B1G gave rise to the list of duff terms given in Figure 18. Note that none of these terms appears to be a genuine TT wrongly labelled as duff, with the possible exception of *direct interpretation*, which might have a specific meaning in the remote sensing/GIS arena.

previous part	considerable importance		potential application
organizational issue	possible link	carlo approach	carlo simulation
different map	different source	different level	different scale
other issue	other data	recent study	recent research
different type	other disaster	recent work	recent year
wide range	direct interpretation	general approach	
potential application field			

Figure 18. Some 'duff' terms from text B1G

There is a handful of reported terms which are incorrect due to a related problem to the *monte carlo simulation* problem, and these represent the third category of false positives. These are left-hand-side sub-phrase cases. In these cases the first word of three qualifies the last. The term *toxic air pollution* is good, and so is *air pollution*, and so is the derived *toxic pollution*. However, *toxic air* is not good. The 3-word term can be read as *toxic (air pollution)* but not *(toxic air) pollution*. (See also the *park border*

plants discussion on page 26). Air pollution is pollution of the *air*, toxic pollution is pollution which has the property *toxic* - these are different ways of classifying pollution. The TT extractor will inevitably find the term *toxic air*, since it has the pattern AN. In most cases there is not a problem and all three terms from the ANN parent pattern (i.e. AN, NN and ANN) can be regarded as good terms, but occasionally the above situation arises.

Resolution of problems such as that of *toxic air pollution* may be resolvable using mutual information methods for detecting word collocations. The paper Kita et al. (1994) describes two approaches, one based on mutual information (MI), and the other on a new technique which they call Cost Criteria (CC). The latter is a method which attempts to find collocations based upon the reduction of effort which a learner of the language would experience if the words involved were learnt as if they were a single word. It tends to find "frozen" phrases such as "thankyou very much". The former seems more useful for our problem, however, since it is able to rank the possible phrases (toxic air pollution, toxic air, air pollution, toxic pollution) numerically. Thus if *toxic air* was given a low rank compared with the other phrases, it could be ruled out as a good TT. It is not desirable to go into the detail of these methods at this point, but the incorporation of such approaches will certainly be investigated for future versions of KEP.

The fourth category of bad terms comes from the hypernym-derived single terms. Single words may be derived which are so general as to be meaningless. These include *method*, *time*, *process* etc. For the all-way look-ahead test on B1G reported above, 109 singleword terms were found, of which 59 were judged to be bad. These 59 bad terms represent a high percentage of all the bad terms (59/92 = 64%) and reduce overall precision disproportionately. Turning off the single-word mechanism would cause the precision figure to rise to (658-(92-59))/(750-92) = 95% and change the false positive rate therefore to 5%. Since output from other BNC texts often throws up the same bad singletons, a simple method of easing this problem would be to include a stop list of bad single-word terms; this has been left for the future. However, even terms such as *process* may be correct, e.g. in computer science where *process* is the term used for a running program. Thus the stop list approach may mark some good terms as bad, and by so doing fail to extract a relation instance (e.g. from "We can define a process as a running program").

5.3.2.3 Other Term Acquisition Approaches

The above discussions may have given the impression that Justeson and Katz are the only people to have put forward a TT acquisition method, but this is not the case. As was stated in Chapter 1, terminology extraction is an established field. Many researchers have investigated term acquisition methods, and some of these may be helpful within an improved KEP. Some methods require part-of-speech information, but others do not. For example, Enguehard (1994) in the ANA system claims to have dispensed with the need for part-of-speech tags, although a DS 'bootstrapping' vocabulary is required. This system aims to find new partial-terms such as *shade of* in *shade of paint* by detecting the similarity of this phrase with known phrases such as *colour of paint*. Goldin and Berry's AbstFinder system

(described below) also dispenses with PoS information, and indeed word boundaries. Some of the more recent papers on term extraction are reviewed below; however, this review is in no way comprehensive. The literature on term extraction is extensive, and it would not be appropriate to provide a comprehensive review within this thesis. Interested readers are directed to Salton (1988), who discusses syntactic approaches to automatic index term extraction, and to the references sections of the papers discussed below, which often refer to non-overlapping sets of previous papers.

Rousselot et al. (1996) present an interactive method, also using untagged text, which takes a similar approach in an attempt to discover verb-based relations in text. For example, patterns of the form **A VERB B** are searched for, e.g. *lions eat zebra*, *cats eat mice* etc. Using knowledge of the categories of the fillers of the **A** and **B** slots in the pattern, the system discovers the **eat** relation. This method strictly finds new relations rather than terms, but where a system needs to find *abstractions* this ability is important. Abstractions are discussed shortly. The above example concerns the *eating* abstraction; several terms may map to it (*food consumption*, *eating food*, etc) since, as was discussed in Chapter 1, terms are alternative lexical ways of representing an underlying concept.

Hahn, Klenner and Schnattinger (1996) also dispense with tags, utilising instead examinations of the syntax surrounding new (unknown) concepts. These examinations are used to create sets of *hypotheses* as to the nature of the new concept. These hypotheses are then tested by looking for other instances of the new concept, so that the best hypothesis is found. For example, if *OS/2* is an unknown concept seen in the phrase *a computer with OS/2* then if it is already known that *a computer has part motherboard*, *a computer has part operating system*, and *a computer has part keyboard* then the three hypotheses about OS/2 are that it is either a motherboard, operating system or keyboard. The correct hypothesis is discovered when later in the text phrases such as *the operating system OS/2* or *operating systems like OS/2* are found. It is not clear, however, whether this method has actually been coded into a computer program. It does however provide one method of extracting concepts from appositions e.g. to extract the concept *operating system* and its example *OS/2* from the appositive phrase *the operating system OS/2*.

TT extraction methods may not only dispense with word tags, but sometimes with the word boundaries too. The *AbstFinder* method of Goldin and Berry (1994) utilises a shift-then-compare TT identifier, where the shifts take place at the character level. The method identifies repeated phrases made from characters, including the space character. It is a domain-specific method because certain DS phrases are ignored in order to see better other TTs. Also ignored are common words and phrases (e.g. "the", "and") placed in stop lists. The intended application (analysis of requirements specifications) requires the recognition that a phrase such as *buy widgets* is essentially the same concept ("abstraction") as *purchase widgets*, so a synonym dictionary (actually a DS file compiled by the user) is utilised. The method does seem to be moderately successful. However, it is to be noted that the abstractions looked for by AbstFinder may not correspond to TTs as detected by KEP, since the abstractions clearly do not have to

have exactly the same orthographic forms. They are higher-level concepts and processes such as *buying valves*. These abstractions are clearly relevant to a program which ultimately would want to estimate the number of tasks and corresponding workloads in an engineering project. (Incidentally, the paper Goldin and Berry (1994) also states the desire for an acronym finder. Berry (personal email correspondence) has expressed an interest in KEP's acronym detector as described in this thesis and in the paper Bowden, Evett and Halstead (1998).)

TT acquisition is also possible in languages other than English. Nakagawa (1997) has made a study of "index words" for Japanese. Here, TTs usually take the form of compound nouns, with few or no adjectives (personal e-mail correspondence with H. Nakagawa). Nakagawa does not take account of the fact that TTs are repeated in a text because of the way his method counts *distinct* words. The scoring proceeds as follows. For a distinct noun N in the text, the following two counts are defined:

Pre(N) = number of distinct nouns that immediately precede N in the text

Post(N) = number of distinct nouns that immediately follow N in the text

(Note: the wording is that of the author of this thesis; Nakagawa says "come just before" (rather than "immediately precede"), and he also adds the phrase "and make compound nouns with N" to each of the above.) The above counts are combined to give an *Importance* score for any single noun or compound noun in the text (*Imp*). Several different *Imp* formulae are suggested. One is a simple addition of all the Pre and Post scores for each N in the compound (or single) noun. Others involve products rather than sums (with Pre and Post values having 1 added to them to avoid times-by-zero problems). The candidate TTs are then sorted into decreasing Imp order. The list is then examined. For high Imp values the terms are very likely to be "index words" (as judged by a panel of humans), and furthermore these terms are mostly compound nouns (as opposed to single words).

The results fit well with the Justeson and Katz approach. The main difference seems to be that Nakagawa is taking a *distributed* approach. Instead of looking for repeated NNs etc, he looks at the individual Ns in the terms, and checks whether they occur in *other* terms. If they do, they get a higher score. So what this method does is to find out how *likely* a particular N is to be involved in TTs, and weights it accordingly. Thus a term made from three "very likely" Ns will have a higher score than one made from not-so-likely Ns.

Furthermore, the singleword TT identifier inherent in Nakagawa's approach is equivalent to a hyponym method with higher scores for single nouns that can be found in larger numbers of longer terms. This is an interesting improvement; once one knows that *disk file* and *output file* are TTs, then one can be fairly

sure that *file* is also a TT. The discovery of a third term *input file* adds a little more certainty to this hypothesis. Nakagawa's method quantifies this extra bit of confirmatory knowledge.

French researchers are also interested in automatic TT acquisition. Oueslati, Frath and Rousselot (1996) describe a system for French which finds repeated terms yet rejects 'duff' terms using stop lists. This system builds tree hierarchies so that hyponyms are identified. Bad terms are also discussed in Daille (1995) which reviews various TT acquisition methods for French, concluding that a simple frequency count is almost as good a determiner of TT status than any other (more complex) method.

One of the problems that besets automatic TT acquisition is that of the singleword term. This issue has been raised before in this thesis, and some solutions have been suggested. Nakagawa's method above tackles it in a manner which is integrated with the overall approach taken. The "hypernym term" method finds single word TTs in a manner already described (see Figure 16), and has been suggested independently by several researchers.

However, a novel method has also been suggested by Yang Huizhong (1986). This approach uses data from several separate texts known to be about different topics. As such it is not useful for KEP, but it is nevertheless worth considering, since the method might form the basis of some future KEP enhancement. The method for single word term discovery is to examine *several* texts from the corpus, each from a different subject area, doing word frequency counts for all non-function words. Then those words having high *peakratio* and high *rangeratio* are likely to be single word terms. These metrics are defined as follows. *Peakratio* is the maximum frequency of occurrence of a word (i.e. the number of times the word occurs, in the text in which it occurs *most*) divided by the average frequency (i.e. the total number of occurrences of the word in all the texts, divided by the number of texts). Thus peakratio is a measure of how specific a chosen word is to one particular text (the one in which it might be a technical term). *Rangeratio* is maximum frequency divided by minimum frequency. It therefore measures for the chosen word the relative range of occurrence over the texts.

The method amounts to looking for single words (nouns) that occur a lot in a certain subject, but not much elsewhere. These are then likely to be single word technical terms in that subject. Although the author does not make the point, clearly the method's success depends upon it being used on a large sample of texts, where each subject (medicine, nuclear physics etc) has *several* texts to represent it. In this way the effects of idiosyncratic authoring (e.g. the tendency of a particular author to use certain words) might be ruled out. Although this approach is of little use to a system which processes one text at a time, it might prove useful in a learning system which compiles lists of DS TTs for use by other programs. This type of approach to finding singleword terms has also independently been suggested by Ahmad (1995), who refers to the document-specificity of certain terms as their "weirdness". The method is appealing because it mimics one of the ways in which a human surely detects TTs. However, it cannot

be the whole story, for a human reader is able to detect common words used as technical terms within a single document, even where the exact meanings of those TTs are not apparent. Thus there must still be scope for devising a novel function to detect singleword TTs within a single text.

It is difficult to compare the performance of the term extractor embodied in KEP with other working term extractors, particularly where these are designed to process natural languages other than English. There is as yet no MUC- or TREC-like competition designed to compare automatic term extraction systems. Where evaluations are reported there is often lack of consensus as to the metrics to be used, despite the availability of precision, recall and false positive rate as good measures. Evaluations often focus on some particular aspect such as on multi-word terms only, or on all terms excluding those involving prepositions. For example, Lauriston (1994) has described the TERMINO system (which is a term extractor for French) and claims that 51% of all complex (i.e. non–singleword) terms from an 8500-word text containing 592 manually-identified terms were extracted, with "noise" (i.e. false positive rate) being 52% of all terms reported. However, no mention is made as to the method of manually identifying terms (e.g. as to scope – within the domain or without), and in any case only one text has been evaluated. (It is also not clear as to whether these figures relate to *distinct* terms, or to all occurrences of all terms.) Perhaps therefore there is a need for a competition within the automatic term extraction research community to help standardise the metrics and, ironically, terminology to be used.

Clearly, TT acquisition is an important field of research, not least because of its obvious practical uses. Although KEP's TT extraction performance is good, its central importance within the KEP program means that future enhancements must regard TT acquisition performance improvement as of the highest priority.

5.3.3 Acronym Extraction

The acronym extractor has been evaluated against 20 BNC informative texts. KEP was run in acronym-only mode for each text and comparison of the output made against a manual inspection of the source. The results are summarised in Table 15 on page 143. The following studies have also been reported in Bowden, Evett and Halstead (1998).

In the table, the columns headed C0 through C9 are manual and KEP-made counts/percentages made from the source. These columns are about how acronyms occurred in the texts rather than about KEP's performance in retrieving them. They are constituted as follows:

C0	Percentage of distinct acronyms present in the text which had an expansion in the text
C1	Percentage of acronyms where the *expansion* was bracketed in some way
C2	Percentage of acronyms where the *acronym* was bracketed in some way
C3	Percentage of acronyms where a bracketed acronym immediately followed the expansion
C4	Percentage of acronyms where the expansion was a single hyphenated word
C5	Percentage of acronyms where the expansion was a single non-hyphenated word
C6	Percentage of acronyms where the acronym was made exactly from capitals in the expansion
C7	Percentage of acronyms where acronym was made from expansion with "glue" words deleted
C8	Percentage of acronyms which were *exact*
C9	Percentage of acronyms where expansion did not occur near the first occurrence of the acronym

In counts C1 to C9, for *acronyms* read *acronyms having expansions explicitly stated in the text somewhere*. The term *acronym* is defined here as *any word comprising 2 to 7 letters, all of which are capitals*. Thus acronyms longer than 7 letters have not been investigated and do not contribute to the metrics given in the table. These are thought to be extremely rare, although no formal count has been made of them. Also not included are initialisms such as D*f*E (Department for the Environment) i.e. mixed-case abbreviations.

Note that the counts are non-exclusive; for example, all those occurrences counted for C3 (cases where a bracketed acronym immediately followed its expansion) were also counted as part of C2 (cases where the acronym was bracketed, regardless of its position with respect to the expansion). Similarly, acronyms counted for C6 might also take part in C5, C2 etc, and so on.

The first column (No. pres.) gives the actual number of expanded acronyms/initialisms present in the named text. Note that this count includes cases where the expansion is some way from the acronym - in other words where the expansion is *somewhere* in the text, even though there are no syntactical clues to link it to the distant acronym. This is in keeping with the philosophy of asking whether a typical human reader would have discovered what the acronym stood for. This approach ensures that recall figures are conservative. However, the figure does not include those cases where world knowledge and a degree of mental processing would be required to find the acronym expansion (such as the knowledge that "RU" often stands for "Research Unit", so that for an acronym like *SSRU* a human would look for the *SS* expansion separately). Thus the figure in the first column is a count of distinct acronyms having *explicitly stated* expansions somewhere in the text.

It is important to know this figure, because a 100% recall and precision sometimes occurs for BNC texts having only a handful of expanded acronyms (see e.g. ALG and CP9). Conversely, oddly low figures can occur (e.g. 33% recall for HT6). Clearly, where there are only a few expanded acronyms in a text, the precision and recall figures are prone to wide fluctuations due to the large percentage impact of each single acronym occurrence. Thus those texts with larger numbers of expanded acronyms are to be

preferred when attempting to determine KEP's performance. For example, texts B1G, B77, CTR, B3C and K5C probably give a more realistic idea of KEP's acronym-extraction performance.

KEP's major acronym extraction performance metrics, recall and precision, are given in the rightmost two columns of the table. Recall is calculated as the number of distinct full extractions correctly found divided by the number present in the text (as established manually), expressed as a percentage. Precision is calculated as the number of distinct full extractions correctly found divided by the number of distinct full extractions reported, expressed as a percentage. Precision includes false positives i.e. cases where KEP found an expansion even though none was present in the source. The false positive rate has not been individually calculated but is estimated to be around 2 to 3% on average. In the above definitions, "correctly" means that the given acronym/expansion pair was exactly right in all details.

The reported figures show that the acronym extractor has the ability to detect and extract capitalised acronyms and initialisms from a wide range of informative texts with high recall and precision, where those acronyms etc are explained in the text. For the five larger texts suggested in the previous paragraph but one, the mean precision is 84% and the mean recall 72%. For all 20 texts, the mean precision is 90% and the mean recall 73%.

Although the acronym extractor detects all capitalised words of 2 to 7 letters in length, the above figures do not say what percentage of "acronyms" triggered by KEP were really names, Roman numerals, or words capitalised for stylistic reasons. Resolution of these cases is not always a simple task even for a human reader, since in the absence of anything that looks like an expansion it is not always possible to make a firm decision as to whether a capitalised word really is an acronym or initialism (rather than a name of a project or item of equipment, say). Although the algorithm always detects capitalised words that might be Roman numerals, it is not always able to confirm this suspicion since it does not currently study the text for contiguous runs of numbered points.

Roman numerals and other capitalised words without expansions slow down processing due to fruitless searches for expansions, but due to the low false positive rate this does not greatly affect recall and precision. In order to speed up processing a future enhancement will involve giving up trying to find the expansion for a suspected acronym after, say, five tries. It is not thought that this would greatly affect the two performance metrics.

BNC text	No. pres.	C0 %	C1 %	C2 %	C3 %	C4 %	C5 %	C6 %	C7 %	C8 %	C9 %	KEP prec.	KEP recll.
B1G	60	50	12	88	83	3	2	52	12	57	7	93	92
HU6	7	30	0	12	12	0	0	4	4	8	4	83	71
ALG	4	100	0	25	25	0	0	100	25	75	25	100	100
B77	31	21	10	35	29	0	0	58	19	68	3	86	61
CND	7	11	14	0	0	0	0	86	29	57	0	75	86
CP9	1	2	0	0	0	0	0	100	100	0	0	100	100
CTR	25	26	0	12	12	12	0	72	12	80	24	77	80
G14	7	47	14	29	29	14	14	14	0	29	0	80	57
HT6	3	33	0	100	100	0	67	0	0	33	0	100	33
HX2	11	26	9	73	73	0	0	73	36	45	9	83	91
ARC	2	12	0	0	0	0	0	50	0	50	50	100	50
B3C	17	27	6	59	59	0	0	65	24	35	18	75	71
EAR	0	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
F9D	14	25	7	57	57	0	7	57	7	64	14	79	79
FCH	0	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
FTY	9	24	0	33	33	0	0	33	11	33	22	100	56
HD1	2	100	0	0	0	0	50	0	0	50	0	100	50
HXF	6	46	67	0	0	0	0	50	33	33	17	80	57
H0H	3	37	0	0	0	0	0	100	66	33	66	100	67
K5C	24	16	12	33	33	0	4	79	21	58	29	87	54

Table 15. Acronym extraction results

It is also worth considering the comments made earlier relating to texts such as surgeons' reports, which are essentially historical narratives between experts. Here, even very specialised acronyms are not expanded - there is no need, since the reader is bound to understand them. Several of the above texts showed evidence of this effect. For example, the BNC text CND did so (it was on computing, and did not explain terms such as IBM, PC, MIPS, CPU etc). At a more general level, there are some acronyms for which we are *all* experts, such as UK, USA, UN etc. Again, these are rarely expanded.

The above corpus study gives a clear indication of how acronyms are introduced in informative text. The most popular way is by stating the phrase to be abbreviated and following this with the acronym in brackets of some description (most commonly round brackets, but sometimes paired commas or dashes etc) - see column C2. In most cases the bracketed acronym immediately follows the expansion (column C3). It appears to be much less popular to state the acronym but place the expansion in brackets (column C1). Column C9 shows that it is rare to expand an acronym other than at its first occurrence - but the small rate shown may be due in part to the use of an acronym in a heading before its use and explanation in the following paragraph. Counts C6 - C8 show that where capitals, glue-words or exactitude are concerned, acronyms are variable and highly text-dependent. In particular it is clear that most acronyms

are not exact (C8). These results are indicative but clearly a larger number of texts would be desirable for a definitive study of acronym/initialism occurrence in British English explanatory text.

It would be a relatively simple matter to add code to allow KEP to update a permanent acronym file on disc. Any acronyms found by KEP would be added to this (including multiple expansions for a single acronym, such as *polychlorinated biphenyl* and *printed circuit board* for PCB). This could be useful to compilers of dictionaries of abbreviations. However, in keeping with the domain-independent philosophy it would not be advisable to use this list as a look-up table for future extraction attempts.

The above experiments also showed areas in which KEP's acronym performance could be improved still further - such as in sentence delineation. Sometimes an acronym was not extracted solely because KEP failed to cut the text into sentences correctly. This means that the recall and precision figures given above could have been better, had this other aspect of KEP worked better. Another area of improvement would be to prevent suggested expansions such as "teapots , violins" for *TV*. There are probably very few acronyms (of any length) having a three-word expansion with the middle "word" being a comma (although commas do validly occur in longer expansions).

5.3.4 Triggering

5.3.4.1 Triggering Evaluation

The triggering process aims to place a tick against those sentences which might have within them an instance of a relation (definition, exemplification etc). This process does not aim to mark *only* those sentences that definitely do contain a relation instance. Triggering is a filtering operation that rejects (blocks) sentences which do not have possible relation instances in them. It is undesirable yet acceptable to allow sentences not having relation instances to pass through the filter; undesirable because they will waste processing effort, but acceptable because they will probably *not* give rise to an extraction (i.e. a false positive extraction). In these circumstances the most appropriate way to judge the performance of the triggering mechanism is to calculate how many sentences were blocked by the filter when they should have passed through it, the *wrongly blocked rate*, which is a count of the incorrectly blocked sentences, s(wrongly-blocked), expressed as a percentage of those sentences which did or should have passed through (s(passed) + s(wrongly-blocked)). It is the s(wrongly-blocked) sentences which will give rise to lost extraction opportunities and hence contribute to a fall in extraction recall rates.

Despite the above argument, it has to be said that a filter which passed all sentences presented to it would be useless (in fact, it would be worse than useless, since it would have performed substantial unnecessary processing). Therefore some measure of the fraction of sentences passing through the filter would be of interest. This fraction is the *ideal pass rate* calculated as (s(passed) + s(wrongly-blocked)) / s(all), where s(all) is the total number of sentences in the text, not including "too long" sentences, as determined by KEP. (Thus s(all) = s(blocked) + s(passed).) The ideal pass rate adjusts for the wrongly-blocked sentences, whereas the *actual pass rate* shows the actual fraction achieved, s(passed) / s(all).

It is important to realise that the ideal pass rate is not the fraction of sentences that actually did contain an instance of a relation, for it includes sentences which passed through the filter even though they did not ultimately contain a relation instance. The number of extractions made is usually substantially lower than the number of sentences passing through the triggering filter because (a) only triggered sentences *containing a TT* will be further processed, (b) some *presentational sentences* will be filtered out, (c) not all sentences left will have extraction templates stored in the external files, this being mainly due to the fact that many of them do not contain a relation instance at all (merely a trigger phrase used in some other context). It is cases of this last type which dominate the "wrongly passed" rate.

The above task involves manual examination of all sentences not passed by the triggering stages so that it can be decided whether they ought to have passed through the filter. The wrongly blocked rate, actual pass rate and ideal pass rate were found for the BNC file 'B1G' as given in Table 16. The results are given for each of the four relation types searched for. In this table the counts do not include apposition trigger patterns; this is discussed shortly. The figures also omit counts for *is a* type triggers.

Relation	s(all)	s(passed)	s(blocked)	s(wrongly-blocked)	wrongly-blocked rate	actual pass rate	ideal pass rate
D	1489	96	1393	6	6/(96+6) = 6%	96/1489 = 6%	(96+6)/1489 = 7%
E	1489	212	1277	5	5/(212+5) = 2%	212/1489 = 14%	(212+5)/1489 = 15%
P	1489	73	1416	1	1/(73+1) = 1%	73/1489 = 4%	(73+1)/1489 = 4%
H	1489	226	1263	8	8/(226+8) = 3%	226/1489 = 15%	(226+8)/1489 = 16%

Table 16. Triggering evaluation results for BNC text 'B1G'

Note that the wrongly blocked rates are low and so there is little difference between the ideal and actual pass rates (less than or equal to one percent in all cases). This is good, for it means that not too many extractions will ultimately be missed simply because the sentences containing them were ignored. The pass rates are likewise acceptably low (a high pass rate indicating that the filter is not doing much useful work). Thus KEP's dual triggering mechanism (positive and negative triggering) appears to perform very well for non-apposition triggers.

This performance does however become degraded if apposition triggers such as **,*,** are included (i.e. any sentence containing two commas with text between them, and so forth). Very many sentences, perhaps as many as three-quarters of all the sentences in a text, follow such patterns, and few of them will actually contain an appositive relation instance. It is difficult to know how to deal with this situation. At the moment the appositive trigger patterns are searched for by the positive triggering function (no negative triggers being allowed for these). If appositive triggers are switched off, no appositions will be detected or extracted. If they are switched on, much unnecessary processing occurs. One solution might be to re-write apposition detection as an entirely separate function outside of the general approach taken so far. This seems a sensible idea given that appositives suffer from another apparently intractable problem, which is that even if apposition is present, it is not possible to tell from the pattern alone *which* relation is present. There may be no key words such as *define* or *an example of* to help this determination. In the sentence *The byte, a contiguous group of eight bits, is the basic unit of memory* there is a definition (of *byte*). In the sentence *The Scimitar, a sports car, remains ever popular* there is a hypernym/hyponym relation (a Scimitar is a type of sports car). This problem is similar to the "is a"

problem discussed in various places in this thesis. Resolving it may require world knowledge, for example to determine that *sports car* is itself a type of *car* and so it is likely that this sentence is all about classification, and therefore likely to contain a hyponym. This separate apposition function will be left for future research.

It is to be noted that KEP is searching for *explicitly stated* relation instances. Sometimes a relation instance is *implicitly* present in a sentence. This often occurs with the hypernym relation. For the BNC text FTE, a separate test showed that 14 sentences blocked by the triggering filter contained such implicit hypernyms. These could not have been extracted by KEP fully, and so it would not be correct to count these as wrongly-blocked sentences. An example of such an occurrence would be a sentence containing the phrase *oligonucleotides I and III.* Clearly, *oligonucleotide I* and *oligonucleotide III* are types of *oligonucleotide.* This situation is similar to that occurring when single-word hypernymic terms are found during the TT extraction phase; in this situation KEP does not state in a glossary entry that the longer terms are types of the singleword term. For example, if two TTs *right-wing politician* and *left-wing politician* give rise to the single-word TT *politician*, KEP does not add text in the glossary entry for *right-wing politician* stating that this is a type of *politician.* This hypernymic relationship is implicit in the glossary as a whole (and in many cases a SEE ALSO link is present). Although it would have been technically feasible to make explicit such "is a type of" links in the glossary entries, it was thought that this was not necessary, and might in fact be regarded as a nuisance (e.g. it is obvious that *map error* is a type of *error*).

In many cases implicitly-stated hypernyms are discovered by the single-word TT mechanism, even though KEP does not extract them from a single sentence. In some cases the sentence structure is so convoluted that even a human reader would have to think carefully (possibly bringing in other world knowledge) to detect an *is a type of* relationship. For example, in the below sentence from text FTE it is probably the case that *polybrene* is a type of *polycation*:

Because of the relatively high transfection rate observed with polybrene, we examined the relative efficiencies of other polycations for their abilities to transfect CHO cells.

KEP cannot in principle extract such difficult cases of the hypernym relation, and so it is not correct to mark such sentences as "wrongly blocked" by the triggering stage. Note however that KEP did extract the hypernym-hyponym terms *cell* and *CHO cell* from this and other sentences. (From a different sentence it also found that CHO stood for Chinese Hamster Ovary, and made the correct cross-references in the glossary.)

5.3.4.2 Trigger and Template Collection

In this section the method by which the positive and negative trigger files were populated prior to the above evaluation will be described. The complete list of trigger phrases for the definition relation, divided into positive and negative sections, is given in Appendix C.

Ahmad and Fulford (1992) have described the method they used to create lists of phrases ("knowledge probes") which indicate the presence of a named conceptual relation ("semantic relation"). The five relations for which lists ("archives") were created were: synonymy, hyponymy, partitive, causal, and material. Note that KEP shares an interest in two of these (hyponymy and partition). Ahmad and Fulford compiled their lists in a multi-stage process which started with a manual identification of likely phrases from the Surrey English Automotive Corpus (reference not provided) together with synonym dictionaries (so that in a phrase such as *X is a type of Y* the synonyms of *type* are found, e.g. *kind*). Wildcard characters were used, so that probe **g%v* rise to** would match *gives rise to*, *gave rise to*, *giving rise to* etc. (This is not an approach which was taken within KEP, where each phrase must be separately listed.) Following this first stage an iterative process was then followed to improve the original lists (particularly the wildcarding patterns) using more automotive texts. The third stage was to use a thesaurus (the Macquarie Encyclopaedic Thesaurus) to refine the lists by searching for more synonyms and related phrases.

The approach taken by KEP has been similarly pragmatic. Initially the positive trigger lists were compiled through introspection. These were then supplemented with synonyms and related phrases using Roget's Thesaurus (Browning (1978)). The lists were also supplemented using phrases gleaned from other published sources, including studies such as that of Ahmad and Fulford (1992) and textbooks e.g. Cruse (1986). Negative triggers were likewise collected, although many of these arose during initial ad-hoc testing of the mechanism during development. A more formal trigger-collection process was also performed using 25 randomly-selected BNC informatives. These texts were passed through KEP's triggering stages (menu choice 3) so that sentences not triggered could be examined to see if they should have been triggered, with the consequent addition of a new trigger phrase. In addition, the triggered sentences were checked to see if any of them could benefit from a negative trigger i.e. because they did not contain a relation instance and because a negative trigger (corresponding to the positive trigger found) was indeed possible.

Note that although the above method was used for the trigger files, the lists of phrases obtained and the patterns (templates) they occurred in (what Ahmad and Fulford term "formulae") were used to start the token and template files used by KEP for performing extractions (see previous chapter). Thus although this section is concerned with triggering, this is a good place to continue the description of phrase collection for the extraction process.

Trigger phrase collection and token/template collection are largely the same operation. Thus *is a type of* would be used as a positive trigger, whereas the wider clause it occurred in, *X is a type of Y* , would be used to create a template such as *Ct0.* , and any token/phrase pairs such as "*t* stands for *is a type of*" would also be created. The initial file population was a manual process which involved examining BNC texts (or the tag-stripped versions in KEP's long output file) and then editing the various files to add the new elements. However, in addition to those phrases collected in the trigger list creation phase, extraction patterns were found using an interactive training mode, selected by choosing option 5 on the main menu and subsequently indicating that interactive mode is required. This mode presents to a user sentences which have passed the triggering filters and which contain TTs and/or acronyms. These are the sentences from which successful extractions usually arise. Such sentences are presented one at a time and the user is able to add a new template which would extract the relation instance, or he may reject the sentence. If any of the tokens in the newly-added template are unknown, KEP prompts for them and adds the new token/phrase pairs to the correct token file. A subsequent extraction run of KEP on the test text then confirms that an extraction has been attempted for the newly-added patterns.

The above interactive process was performed on 5 BNC texts: EAK, ALG, CND, GW6 and CLT. Note that these texts do not include the large text 'B1G' which was the subject of the detailed evaluation presented shortly (section 5.3.6). Thus the text 'B1G' remains "unseen" for extraction purposes. (Clearly, if 'B1G' had been included in the training stage then all possible tokens/templates would have been provided, giving an unrealistic picture of KEP's extraction performance on a truly unseen text. However, this is not to say that the recall and precision figures would then have been 100%, for there are instances where templates could not in theory be provided. These issues are discussed later.)

Having an interactive training mode provides advantages and disadvantages. The main advantage is that a great deal of the scanning effort is removed; the human user need only respond to selected sentences placed before him. Because the program handles file editing, it is not necessary to know how to use a text editor to add a new extraction template. Furthermore, the system itself is able to notice "missing" tokens, and prompt for them when this occurs (either deliberately or by accident). The system is also able to recognise patterns which are already in the template files, and so avoid duplicating them. Thus the interactive mode maintains file integrity, which in turn prevents system error message creation due to missing tokens or badly-terminated entries. (In all cases the external files have lines conforming to a fixed format: for example, the token files require lines of the form *defined===d* i.e. with three equals signs separating the phrase from its corresponding token. Other files may use a terminator: the trigger files have entries of the form *define~*, where the tilde character ends the trigger phrase.) One minor difficulty with this method is that the user needs to be aware of the existing contents of the files, so as to avoid needless duplication of tokens etc. (The system prevents duplicate lines being entered but allows two different tokens to be attached to the same phrase, for example.) For this reason, KEP displays the meanings of all tokens used in a newly-added template. This keeps the user informed. The main

disadvantage of the method, which applies to any interactive method, is that it involves working at the terminal rather than working away from the terminal (from paper listings). Of course this may be done too - the existence of the interactive mode does not preclude off-line methods. But an interactive session may last hours on a large text and should really be performed in one sitting.

One other disadvantage of the interactive mode as described is that it does provide opportunities for good relation instances to slip through the net, because only triggered sentences bearing TTs/acronyms are presented. It is possible that a sentence was not triggered when it should have been, so the comprehensivity of the training mode does depend on the quality of the trigger lists (i.e. whether they approach being complete). It is also possible that a pattern, for a relation instance containing a genuine TT (which occurred just once), is missed (i.e. because KEP did not recognise that TT). Also, relation instances may occur for concepts which are not TTs at all. In a sentence such as *An example of the use of loops in a C++ program is the use of the do-while loop for input collection* the concept is *the use of loops in a C++ program*, which clearly is not a TT as found by KEP (although it may contain a TT - more on this later during the relation instance extraction performance discussions). Long TTs not found by KEP are also a problem (e.g. "relation instance extraction performance", which is a TT in this text comprising four nouns). However, it is thought that the advantages of having an interactive training mode outweigh the disadvantages, especially as this mode is an additional method of collecting patterns and not the exclusive method.

The interactive mode does in fact represent a semi-automatic learning mechanism for pattern-matcher patterns. Automatic learning of pattern-matcher templates is currently a vigorous area of research. Most systems, including KEP, require a human to "close the loop" i.e. to indicate to the system that a suggested pattern is good or bad. A few systems do without the human input (e.g. the domain-specific AUTOSLOG information extraction system, which learns *transitive verb/ recipient of action* pairs such as *shot general Perez* (Riloff (1996), Lehnert et al. (1992))), but this is not possible for KEP because there is usually one good extraction pattern for a relation-bearing sentence, hidden amongst many bad patterns. Therefore a person is required to indicate to KEP which is the good pattern to be stored as an extraction template. However, whether a system is totally automatic in its learning or human-aided, it will require a corpus of text as training data. KEP's training corpus is a subset of the BNC, although a training run may be performed at any time on a CLAWS-tagged text from any source. Cardie (1997) provides a review of corpus-based learning systems in Information Extraction (IE), the field most closely related to Knowledge Extraction (KE). This review suggests that most automatic-loop systems require both syntactic and semantic/pragmatic knowledge. Thus such systems often tag or parse the text, identify the grammatical subject and object, and then use domain specific pragmatic knowledge and verb subcategorisation knowledge to determine role-fillers (e.g. that *Lima* is a city, that *Perez* is a general, that *The Shining Path* is a terrorist group, that generals are often the targets of terrorist groups, and that the direct object of *shot* might be a general and the subject might be a terrorist group). Since it is a matter of

design philosophy that KEP does not maintain DS knowledge bases, and since deep processes such as parsing are not used, KEP cannot learn new extraction templates completely automatically.

5.3.5 Detection of Presentational Sentences

The mechanism used to identify presentational sentences has been described in section 4.6.9. The point at which KEP tests for a presentational sentence is after a triggered sentence has been identified as containing at least one TT or acronym. This stage is once again a filtering process, the aim being to filter out *relation references*. A sample of filter phrases is given in Figure 19 below. These phrases were collected in an ad-hoc fashion during test runs, and from introspection.

Definition filters	definition above previous definition was defined in in the definition no definition is possible
Hypernym filters	previous type earlier kind type above above categorisation
Exemplification filters	example above aforementioned example a poor example consider for example see e.g.
Partition filters	listed above previous list components given above parts given in

Figure 19. Sample of presentational filter phrases for each relation type

KEP was run in highlight-only mode with a 10-sentence TT look-ahead window (apposition triggering and *is a* triggering being switched *on*). Highlighted sentences were counted (h), and out of these, those marked as presentational by KEP counted (p(KEP)). Of the sentences highlighted by KEP, the number deemed to be relation references was counted (p(KEP_HUM)). For example, for the BNC text B1G, h = 2,056 sentences were highlighted (figures for all four relation types have been added) of which only p(KEP) = 13 were marked by KEP as presentational, and of which p(KEP_HUM) = 10 appeared to be genuine references to relation instances given elsewhere. Thus the filter only removed a very small percentage (13/2056 = 0.6%) of the highlighted sentences. In this case, the filter was right to remove these sentences from consideration since they either did not contain a relation instance at all or did contain a relation reference. No attempt was made to read each of the 2,056 highlighted sentence-cases to see if there were any relation references which inadvertently passed through the filter, since this was

an impracticably large task not justified by the likely gain. The figures for two BNC texts are given in Table 17.

BNC text	Total no. of highlighted sentences, h	No. of blocked sentences, p(KEP)	No. of genuine relation refs blocked, p(KEP_HUM)	Fraction of h filtered out	Fraction of p(KEP) which were relation refs.
B1G	2056	13	10	0.6%	77%
FTE	2973	22	1	0.75%	approx. 5%

Table 17. Detection of presentational sentences

The number of triggered TT-bearing presentational sentences in the selected BNC texts appears to be genuinely low. Since there are so few relation references to be discarded, it follows that the improvement of this filter facility may be assigned a low priority. The presentational filter in practice makes little difference to KEP's extraction performance metrics, as described in the following section. Thus although it seemed a useful thing to do, in practice it turns out to be of little use. It does indeed highlight sentences containing pointers to figures, tables, and previous parts of the text, but such sentences do not appear to be plentiful enough to justify this function, even if it could be made "perfect".

Although not appearing of prime importance to KEP, the presentational vs. informational divide has been considered by other computational linguists. Moore and Pollack (1989) addressed the issue during a criticism of *Rhetorical Structure Theory* (RST) (Mann and Thompson 1988). Their thesis is that a text is simultaneously both presentational and informational, so that a single RST analysis will not do. For example, in the text *George Bush supports big business. He's sure to veto House Bill 1711* two equally plausible RST analyses exist, one using the evidence relation, and the other using the volitional cause relation. The evidence relation is presentational, but the volitional cause relation is informational. What is the solution? It is not given in the paper, but it could be that one needs to perform two separate RST analyses on the text, one utilising solely presentational relations, and the other using only informational relations. The problem with this is that one may affect the other. The same piece of text may supply different information if its intention is perceived differently. So possibly one needs to go through the text, assigning *all possible* relations (of either type) wherever they can be assigned. A following process could then track through the marked-up text attempting to produce a coherent pathway in which the two views coexisted. This might require world knowledge, so may not be straightforward. In any case, RST is not an automatic procedure and so the usefulness of such an approach to automatic KE programs is doubtful.

Tucker, Nirenburg and Raskin (1986) also consider the presentational / informational dichotomy, although these exact terms are not used. The main topic of the paper is *focus shift* and how it may be

evidenced e.g. by definitions and exemplifications. It is clear that the presentational structure of a document is reflected in the placing of examples etc (see also Mittal and Paris (1993)) so this is a topic which should not be ignored in the long term.

5.3.6 Conceptual Relation Extraction

5.3.6.1 Introduction – How to Detect a Relation Instance

Precision and recall for relation extraction are the prime metrics for KEP evaluation. They test the novel pattern-matcher/extractor employed by KEP and they demonstrate the degree of success of the domain independent, shallow approach attempted by this program. They subsume the metrics for the lower-level functions described elsewhere in this chapter and act as the prime evidence for KEP's claimed success rate.

As with technical terms and other aspects above, there is inevitably a degree of subjectivity involved in determining precision and recall for relation instance extractions. As usual, the difficulty lies in deciding whether or not the items to be extracted are really present in the text. Ultimately, deciding upon whether a given sentence contains a definition etc or not must reside with the human evaluator, which in this case is the author of this thesis. To do this, use was made of the definitions (of definitions, exemplifications etc) described in Chapter 3, together with Skuce et al.'s test (page 73), and the author's own judgement. Despite this, the decision process was still difficult. Take for example definitions: in the table of results following shortly it is stated that there were 12 definitions found manually in the text. There were in fact only 8 *clear-cut* instances of single-sentence definitions – but there were also 4 other cases where one *might* regard the sentence as holding a definition, bearing in mind Skuce et al.'s test. Here are two definitions, the first "clear-cut", and the second "possible", both taken from text B1G:

> `The areal interpolation problem can be defined as the transfer of data from one set (source units or zones) to a second set (target units) of overlapping, non-hierarchical areal units.`

> `The point interpolation methods essentially use a point, usually the centroid, as a surrogate for the areal units and then apply conventional point interpolation methods.`

The former is a straightforward definition of *areal interpolation problem*. The latter is probably a statement about *point interpolation methods*, but the word *essentially* indicates that it is talking about the *essence* of these, i.e. what their meaning is (see Skuce et al.'s test, and the discussion about definitions in section 3.3). (KEP correctly extracted the first of these.) This example demonstrates the difficulties of identifying relation instances in text, even when a formal definition of the relation is available to aid in the decision-making process. However, in order to provide conservative results, all twelve

probable/possible definitions were included in the recall calculation for definitions given later. These are listed in Table 19 which follows shortly.

Skuce et al.'s test applies only to *definitions*. It would be helpful to have similar tests for the other three relation types used by KEP. Therefore, to aid in the manual detection of *partitions*, the partition sub-categories put forward by Winston, Chaffin and Herrman (1987) have been considered; these are investigated in detail in an paper concerning the taxonomy of the partition relation Bowden, Evett and Halstead (currently being written). In this paper the scheme of Winston et al. has been modified slightly, in that the *Stuff/Object* partition type of Winston et al. is not regarded as a type of partition (it is instead regarded as a type of *material* relation - what some thing is partially made of). The types of partition used are: (1) *Component/Integral Object,* (2) *Member/Collection,* (3) *Portion/Mass,* (4) *Feature/Activity,* (5) *Place/Area.* These types are used to determine whether a partition relation is indeed present; if it is not possible to place the putative partition instance into one of the sub-categories, then it is not in fact a partition. This test procedure says something about the partition relation: that partition may not be a single relation type. Although partition types are all about dividing some thing into smaller things, in the long term for programs such as KEP it might be better to regard all the partition relation sub-types as distinct relation types, each having its own set of trigger and template files.

For *exemplifications*, where the choice of the relation is often difficult, a test question is proposed: *Is the purpose of this relation instance mainly that of providing an example of the concept, or is it really to define, categorise, explain etc the elucidation of the concept?* This test again requires that a subjective decision be made (i.e. as to what, in the evaluator's opinion, the writer's purpose was when he constructed the sentence) but assuming that the purpose is usually detectable the test question allows the evaluator to find examples given purely for the sake of exemplification *of the concept*. Note that the test forces the evaluator to ask whether the relation instance is really about the concept or about the concept's "example". True exemplifications occur where the text is about some concept, which needs exemplifying so that the reader gains more knowledge of the *concept*, not of the example of that concept. Thus an article on high-level programming languages may use an example of a high-level language (e.g. PASCAL) to aid the reader in understanding what a high-level language is, rather than to say anything about PASCAL (PASCAL is not the subject of the article).

Where *hypernyms* are concerned, the test question is simpler. It is merely a matter of determining whether a concept / parent-class pair is being given. Again, it is the *concept* which is being explained by way of extra information for the reader (i.e. what sort of class the concept falls into). This relation type was the easiest of the four to identify, perhaps because of its purity. However, the hypernym relation occurs frequently in implicit forms. KEP has not been designed to extract implicit hypernyms and therefore such cases are not counted in these evaluations. (Many of the hyponym/hypernym pairs implicitly present in text are in any case "dubious" for a glossary maker – e.g. in the phrase *birds with*

long legs one would probably not want to make the extraction **Concept:** bird with long legs **Hypernym:** bird.)

Having discussed how relation instances were identified in text, it is necessary to consider whether all such instances are to take place in the evaluation. (The discussion that follows concerns only single-sentence relations; multi-sentence relations were not counted, except where they were multi-sentence solely because of an anaphoric link to another sentence. Such cases are effectively single-sentence instances.) It is apparent that two evaluation approaches are possible: (1) calculate metrics only for those cases where KEP has identified the concept being defined etc as a technical term, or (2) calculate metrics for all definitions etc regardless of whether they are for a KEP-identified TT or not. Since these approaches essentially differ only in whether a TT was recognised (assuming the definition etc syntaxes used are from the same set in both cases), it was decided to take approach (2). This also has the advantages of being conservative in its estimate of KEP's performance, and of being more practically useful, i.e. it answers the questions *What percentage of all the single-sentence definitions etc present in the text does KEP extract?* and *What percentage of the proposed definitions etc are correct?*. In other words, given an explanatory text, the approach (2) answers the question *How good is KEP at extracting the chosen types of fact (definitions, examples, hypernyms, partitions) from this text?*

Since the process of determining recall and precision involves a great deal of detailed manual study of the test texts, it was not feasible to carry out the process on large numbers of texts. Manually processing a large (1500-sentence) text takes many times as long as merely reading it. The detailed study of one text to the level demonstrated below may take several weeks to complete, and there are no short cuts available. Thus it was decided to take one large BNC text and apply a detailed analysis for recall and precision. The chosen text was BNC text 'B1G' as encountered in previous evaluation.

5.3.6.2 Evaluation of Precision and Recall

The evaluation was done using the long-output file, which contains a readable tag-stripped version of the text, segmented into numbered sentences, these sentence numbers being cross-referenced to following extraction attempt reports. The first stage was to manually inspect every sentence in the input text for each of the four relation types, and record details of all sentences where a relation instance was deemed to be present. This was done using the sentence-structure part of the output only, i.e. without reference to KEP's own extraction attempts. Details recorded included the concept being elucidated, the relation present, the actual elucidation (definition, hypernym etc), whether the concept was too complex or too long to be a TT as found by KEP, and whether there were anaphoric elements to the concept.

The second stage was to compare the manually identified cases with extractions reported by KEP. Recall was then calculated as the percentage of manual extractions also reported correctly by KEP, and precision as the percentage of correct extractions given by KEP from the total number reported by KEP.

A correct extraction is defined as a reported extraction attempt where (a) this was one of those detected manually, and (b) both parts (concept and elucidation) were the same as those given in the manual extraction. (This precision figure has been labelled as 'strict precision' in the results table given below; the other precision rate, 'useful precision', is explained shortly.)

The recall and precision metrics are given in the table for each of the four relation types. Correct extractions include those with anaphoric concepts such as "this", i.e. where de-referencing of the anaphor would have given the correct concept. Note that the denominator in the Strict Precision column may be larger or smaller than that in the Recall column, since KEP may report more or fewer extractions (denominator in Strict Precision) than *manually* identified (denominator in Recall). Where KEP identifies a relation instance not identified manually, a false positive extraction has arisen. As we shall see shortly, such false positives are not always plain wrong; they may often be of some worth.

	Recall	**Strict Precision**	**Useful Precision**
Definition	5/12 = 42%	5/64 = 8%	24/64 = 37%
Hypernymy	1/12 = 8%	1/5 = 20%	1/5 = 20%
Exemplification	5/50 = 10%	5/7 = 71%	5/7 = 71%
Partition	3/14 = 21%	3/9 = 33%	5/9 = 56%

Table 18 Recall and Precision for each of 4 relation types for BNC text 'B1G'

It is immediately clear that the numbers of relation instances present in the test text vary quite widely with the relation type. Most common were examples – 50 of these were detected manually. Partitions were the next most common with 14 of these being manually identified. Definitions and hypernyms were equally scarce on 12 each. It is also apparent that the strict precision values were low in most cases (exemplification being an exception, discussed shortly), although "useful precision" figures (explained shortly) could be higher. Recall too was low in all cases apart from the definition relation, where KEP has performed reasonably well.

However, terms such as "low" are of course subjective – one might regard the figures given as remarkably high for a shallow NDS KE system. None of the figures is zero i.e. KEP did manage to find relation instances correctly for all four relation types. What is more, the figures do not even represent KEP's maximum theoretical performance, since text 'B1G' was processed "unseen" for patterns in the template files. Adding the missing tokens and patterns improves the figures in all cases, sometimes greatly (see e.g. the forthcoming discussion on partitions). The author believes that the fact that KEP did actually extract some of the available target facts from the test text is a respectable achievement.

Hypernyms

The hypernym relation was relatively rare in the test text. The denominator in the recall for hypernymy reflects this. The figure of 12 is greater than the number of sentences in which they occurred (8), since some sentences contained more than one.

KEP was designed to extract hypernyms i.e. cases such as *X is a type of Y*, where X is the concept and Y its hypernym (parent class). It was not designed to extract hyponyms i.e. cases such as *X includes Y* where X is the concept and Y is its hyponym. Whilst it is true that the hypernym and hyponym relations are symmetrical between a parent class and an object in it, this is not the case for the textual forms used to hold such cases, as the above example phrases show. (It would of course be perfectly feasible to provide KEP with the ability to extract hyponyms as a separate relation type. For a clarification of the difference between the hypernym and the hyponym relation, the reader is referred to the discussion on relation naming on page 80.)

The hypernyms found in 'B1G' were a diverse set; they included both straightforward concept-parent cases and complex cases involving several hypernyms for more than one concept in the same sentence. However, even in the simple cases some degree of WK was required in order to spot the fact that a classification was being made. Here is a relatively straightforward case:

> Dutch elm disease is a floral hazard but is exacerbated by the transport of infected logs.

Clearly, Dutch elm disease *is a type of* floral hazard (note use of *is a* in sentence). There follows a sentence which actually contains two concepts (underlined), each of which has two hypernyms (in italics). The wording of the sentence shows that it is deliberately making the point that each concept can be placed in more than one parent class:

> It is more useful (Johnson (1983) to adopt a non-partitional scheme (fig. 10.1) that recognizes, for instance, food poisoning as both a *personal* and *consumer hazard*, or lead pollution (e.g. from car exhausts) as both a *meteorological event* and a *technological hazard* (both public and private).

Furthermore, ellipsis affects one of the hypernyms - *personal hazard* – and another hypernym has attached to it two of its other hyponyms, each ellipted – *technological hazard (public technological hazard, private technological hazard)* .

There was even one semi-exophoric hypernym-bearing sentence, in which the hypernyms were listed in a nearby table. It is not surprising, therefore, that KEP faired relatively poorly with this relation. In fact, the only one of the 12 manually-identified hypernym relations correctly extracted by KEP was from the following sentence: *Data integration is one of the fundamental GIS operations (Burrough (1986)*. Here,

the identification of the concept *data integration* and its parent class *fundamental GIS operations (Burrough (1986)* occurred. Note, however, that this sentence was also a target for the definition relation (discussed shortly).

Partitions

The hypernym and other relations are easily confused where the member/collection partition subtype or the member/set relation occurs. For a phrase such as *England and France are EU countries* it is important to realise that this is **not** saying that England *is a type of* EU country. It is saying England *is a member of the set of* EU countries. However, as discussed in section 3.3, the member/set relation is not regarded as a subtype of the partition relation, but as a relation in its own right. (The member/collection subtype is however one of the partition types counted.) However, the example given shows how difficult it can be to decide upon the exact relation present, since a statement such as "England is part of the EU" seems reasonable. (KEP found 2 Member/Set relation instances, mistaking them for partitions, in 'B1G'; both had anaphoric concepts. These two instances took part in the "useful precision" metric, as explained in the paragraph on Definitions shortly.)

Although the partition relation was subdivided for the purpose of identifying it, counts were made for the relation as a whole and not for each sub-relation. The reason for this is statistical – one would require much more data before meaningful conclusions could be drawn concerning the individual partition sub-types. However, a partition-only investigation is planned using many tens of BNC texts in order to discover significant trends in British English (see Bowden, Evett and Halstead (currently being written)). The topic of relation sub-types is considered in the next chapter.

KEP correctly extracted the partition from the following sentence:

```
The basic components of a DSS comprise: data storage files; data
analysis modules; display and interactive use technology.
```

Here, using the previously-extracted acronym *DSS*, KEP gave the extraction:

```
Concept: decision support system
Part: data storage files
Part: data analysis modules
Part: display and interactive use technology
```

The disappointment of the low percentage of partitions extracted by KEP is ameliorated somewhat by the fact that another 9 of the non-extracted partitions could have been extracted by KEP if their triggers, tokens and patterns had been present in the relevant files. This would raise recall to 12/14 = 86%, and represents the best that KEP could have done on the manually-identified partition set. This is a very high level of recall. For example, there is no reason why KEP should not be able to make a successful extraction from the following:

```
The second main component of the GUI is the imaging model which
controls the screen representations such as fonts and icons; an
example of this is Display Postscript.
```

Here, the concept is *GUI* (actually generic despite the use of the definite article preceding it – this from context) and this has the part "imaging model which...". This is probably a Component/Integral Object partition subtype, although due to the nature of code it is difficult to place a piece of computer software and its components into the chosen subtype set (in some cases the Feature/Activity subtype is more applicable, or even Member/Collection if the software is viewed as a collection of modules). The sentence above is actually part of a multi-sentence partition instance, where the other sentences do not contain pattern-matchable instances. Nevertheless, the opportunity to extract should not be overlooked in such cases.

Note also that there are three exemplifications in this sentence – two for the concept *screen representation*, and one for anaphoric "this" = *imaging model* (= second main component of the GUI). The author has noticed that this sometimes occurs; sentences often contain instances of more than one relation type. It is possible that such "fact density" observations have uses in automatic text summarisation systems, discussed in Chapter 6.

Exemplifications

Exemplifications were rarely extracted correctly by KEP, despite the large number of them manually identified. The poor recall figure arose almost exclusively because the concepts being exemplified were long TTs (more than 3 words) or complex entities not present in the text in the required lexical form. This is an interesting result, and is further considered shortly, during a discussion on concept non-recognition. However, the precision figures were actually high for exemplification: 5 of the 7 reported examples were good. In fact 4 of these 5 were cases in which the concept was an anaphoric "this", which as stated above is counted as correct if de-referencing would have given the correct concept. Since it is easy for such a pointer to reference a long previously introduced concept, it is actually easier for KEP to find these cases. This explains the unexpectedly high precision figures.

Definitions

The strict precision figures given in Table 18 were calculated as the fraction of extractions given which were correct i.e. previously identified manually in exactly the same form. These strict precision figures are on the low side for definitions. This may give the impression that KEP is reporting mainly nonsense with one or two nuggets of knowledge embedded within it. In fact, this is not the case, because KEP will not report any extraction unless the concept is a valid technical term. Since most TTs found are valid concepts for the text, this means that many of the incorrect extractions are near-miss definitions etc, or

useful pieces of information about a valid concept. However, they have not been labelled as correct since they were not one of the manually-identified definitions etc.

This is a very strict approach. Relaxing it to allow all extractions which one might find "useful" in a glossary gives higher precision rates in some cases. For example, for definitions there were 24 extractions which provided useful information *about* the (valid) concept. Thus 24/64 = 37% of definition extractions were "good" in the sense that they would be useful in the third column of the glossary. Such useful precision figures have been given for all four relation types in Table 18. (Note that useful precision can equal strict precision where there were no extra useful extractions reported.)

For example, here is one such "useful" case, actually two separate definition extractions for the same concept and later merged, not labelled as strictly correct since they were not manually-identified:

```
Concept: data integration
Definition: one of the fundamental GIS operations (Burrough
(1986)
Definition: especially a problem for geographers because
information synthesis is at the very heart of the discipline
```

Placing these "definitions" in the third column of the glossary entry for *data integration* would provide useful knowledge to the glossary reader, even though they are not strictly definitions of the concept (they are statements about it). The following triple-extraction also bears useful information to the reader of an article on recent GIS developments:

```
Concept: user interface
Definition: vital element of any GIS
Definition: now beginning to attract its due attention
Definition: not just pretty screen representations: as their use
is extended they will come to express the whole nature of the
system data model, and will probably become highly specialized
as the interfaces move from function-oriented to task-oriented
forms
```

Some of the extractions are unintentionally humorous and yet reveal views held by the writer, as the following pair of extractions demonstrates:

```
Concept: mrs. thatcher
Definition: a recent convert to environmental conservation

Concept: political concern
Definition: not altruistic
```

Thus the extractions provided by KEP are capable of reporting information/facts which may not strictly fall into the rigid conceptual relation boundaries set for them, but which nevertheless provide a useful insight into the contents of the text being processed.

The remaining definition extractions are not useable, mostly because they report episodic knowledge, usually for a broad term being used in a specific case in the text. Usually the TT reported is dubious for the text, due to its broad meaning. The following are typical cases:

```
Concept: range
Definition: now from -3506 to 204

Concept: system
Definition: used to determine restriction zones for the movement
of sheep after the explosion at Chernobyl in 1986
```

It is interesting to note that for all four relation types many of the non-useful episodic cases were for single-word concepts, such as those immediately above. A plausible explanation for this is that single-word terms are often general terms understood by any reader (range, system, method, approach etc), and hence which do not need to be defined etc. Thus where an apparent definition is detected it is more likely to be for a specific instance of the term, than for the term in its general sense.

5.3.6.3 Failures to Extract Definitions

The precision and recall rates achieved by KEP are encouraging, but it is worth detailing further the reasons for the failure to attempt to extract a given relation instance (recall) and for errors in extractions (precision). This will be done in this and in following sections. However, in this section the definition relation alone will be examined. The focus of this section is the recall metric.

Since there were only a few definitions found manually in the text, it is possible to detail all of them together with their failure reasons where appropriate. These are given in Table 19.

n = KEP sentence number Sentence	Ideal extraction C: concept D: definition	Definition extracted correctly?	Comments
18 The areal interpolation problem can be defined as the transfer of data from one set (source units or zones) to a second set (target units) of overlapping, non-hierarchical areal units.	C: areal interpolation problem D: the transfer of data from one set (source units or zones) to a second set (target units) of overlapping, non-hierarchical areal units	y	A perfect example of a definition given in an explanatory text.
30 The point interpolation methods essentially use a point, usually the centroid, as a surrogate for the areal units and then apply conventional point interpolation methods.	C: point interpolation method D: essentially use a point, usually the centroid, as a surrogate for the areal units and then apply conventional point interpolation methods	n	No pattern stored – context shows that this is probably a definition of the concept (point interpolation method).
465 This is a device which supports PostScript, and which has a specified resolution of 300 pixels/inch, both the vertical and the horizontal direction.	C: This D: device which supports PostScript, and which has a specified resolution of 300 pixels/inch, both the vertical and the horizontal direction	y	Anaphoric "This" not linked to concept (IBM 4216 PagePrinter) in previous sentence, but identification of "This" as concept is counted as good extraction.
509 Characteristics of a user interface A user interface, at its most basic, consists simply of a system for communication with the computer.	C: user interface D: system for communication with the computer	n	Sentence delimiter failed to detect end of heading, which has therefore been prepended to sentence. Also, pattern looks like partition.
514 The GUI is an audio-visual display on the computer screen which presents a screen metaphor for the actions which the computer or program can carry out.	C: graphical user interface D: audio-visual display on the computer screen which presents a screen metaphor for the actions which the computer or program can carry out	y	Note use of "is a" for definition of concept (GUI). Acronym is expanded for concept part.
608 Thus, Bertin (1983) defined the concept of a "map-to-see" as "a clear graphic representation which can be comprehended in a short moment".	C: map-to-see D: a clear graphic representation which can be comprehended in a short moment	n	Concept (map-to-see) not a TT; it only occurred once in the text.
905 A 4-year experimental programme to "collect, co-ordinate and ensure the consistency of information on the state of the environment and natural resources in the European Communities" was set up and labelled CORINE.	C: CORINE D: 4-year experimental programme to "collect, co-ordinate and ensure the consistency of information on the state of the environment and natural resources in the European Communities"	n	Sentence not triggered for definition. Probably a definition of the concept (CORINE) although it appears as appellation (naming) of the entity described by the first part of the sentence. *(table continued on next page)*

1052 Formally, a hazard can be defined as: "a physical situation with a potential for human injury, damage to property, damage to the environment, or some combination of these" (Health and Safety Executive 1989: 30).	C: hazard D: "a physical situation with a potential for human injury, damage to property, damage to the environment, or some combination of these" (Health and Safety Executive 1989: 30)	y	A perfect example of a definition given in an explanatory text.
1053 A hazard is a threat which, given a set of circumstances, may become translated into a realized event.	C: hazard D: threat which, given a set of circumstances, may become translated into a realized event	y	Note use of "is a" for definition of concept (hazard).
1079 By "risk" we understand "the likelihood of a specified undesired event occurring within a specified period or in specified circumstances" (Health and safety Executive 1989: 30).	C: risk D: the likelihood of a specified undesired event occurring within a specified period or in specified circumstances	n	Pattern **b"C"u"0"X.** not in file, where **b** is token for 'By' and **u** for 'we understand'. This extraction could in theory have been made.
1452 Decision support systems and GIS A decision support system (DSS) can be considered as an integration of computer hardware and software specifically designed to complement the human thought process in problem-solving, decision-making and information processing (Benbasat 1977).	C: decision support system D: an integration of computer hardware and software specifically designed to complement the human thought process in problem-solving, decision-making and information processing (Benbasat 1977)	n	Sentence delimiter failed to detect end of heading, which has therefore been prepended to sentence. Also pattern **C(X)c0.** not in file, where token **c** means 'can be considered as'.
1453 According to Berke and Stubbs (1989) a DSS can often be conceptualized as a tool to be used as part of an interactive learning process allowing the user to undertake "what if" analyses and view the consequences of such alternatives.	C: decision support system D: tool to be used as part of an interactive learning process allowing the user to undertake "what if" analyses and view the consequences of such alternatives	n	Pattern **X)C=0.** and token = for 'can often be conceptualized as' not in files. This extraction could in theory have been made.

Table 19. Manually-found definitions from 'B1G' with KEP extraction results and explanations

The main problem with definitions appears to be in recognising them. The table gives instances where context (usually the preceding sentence) demonstrates that the sentence is effectively a definition, although it appears in the guise of a different relation type (e.g. appellation/nomination or partition). This again supports the suggestion that a definition is not a 'pure' relation type – it is a *functional* relation type, where the function of definition may be achieved using other relation types. Thus although KEP has had some success in (a) finding and (b) extracting definitions, in order to do (a) properly it is clear that syntactic solutions alone will not suffice. In the cases examined above, the sentences containing the functional definition occurred near the introduction of the concept. Thus it is conceivable that some extra weighting could be given to such possible definitions, based upon proximity to the first mention of the concept in the text (see also the previous discussion in section 3.3 concerning *grounders*, as introduced

in Flowerdew (1992b)). However, to do the job properly would undoubtedly require a deeper treatment involving semantics.

It is worth asking how KEP's definition extractor compares with approaches by other researchers. The author has found only one other researcher who is attempting to extract definitions from text using lexical patterns on a "(semi)-automatic" basis. Pearson (1996) describes an approach based upon two common lexical patterns involving hyponyms plus further defining characteristics. Two major patterns are looked for:

X = Y + distinguishing characteristic

Y + distinguishing characteristic = X

In the above, X is the concept being defined, and Y is its hypernym. In order to match to these patterns, X, Y and = have validation conditions placed upon them. X must be a term (as with KEP) – but Pearson does not discuss how terms are identified. In addition, conditions relating to the presence of definite and indefinite articles are placed on X and Y in each pattern above. Some of these dismiss patterns which the author of this thesis would regard as containing valid definitions. For example, Pearson would dismiss *The anvil is a tool used by blacksmiths* because in the first pattern she states that X must not be preceded by the definite article. Similarly, Pearson would not consider *Anvils were tools used by blacksmiths* because she does not allow past tense forms in the defining phrase, which she calls the "hinge" (the = part). Pearson also ignores hinges incorporating modal auxiliaries, such as **can be defined as**, on the grounds that they allow for other ways of defining a concept, and hence are not *the* definition of the concept. This seems far too constricting to the author of this thesis; KEP attempts to extract anything that a human would say was a definition (using Skuce et al.'s test), and KEP's recall and precision figures are based upon this assumption. The first definition given in Table 19 actually uses the phrase **can be defined as**, and the author regards this as a perfectly legitimate definition that ought to be extracted. In fact, at least nine of the manually-identified definitions given in Table 19 would not have been found by Pearson (due to their not matching one of the two basic patterns, or due to infringements of definite/indefinite article precedence rules and/or allowed hinge-syntax rules), and it is not clear that the other three would have been extracted either.

Unfortunately, the Pearson paper does not make it clear whether the extractions were performed using a computer program, or whether it reports a purely manual exercise; the latter seems likely, since there is no mention of such a program in the paper. The lack of any discussion regarding automatic term acquisition, part of speech identification, pattern matching algorithms used, running time etc, and the lack of any reference to any previous or concurrent work involving a computer system also support this conclusion. Furthermore, although the paper states that "all simple formal definitions" are retrieved

(which implies a 100% recall), Pearson's "simple formal definitions" are not as broad in scope as those searched for by KEP, as has been demonstrated in the previous paragraph. No false positive rates are given, even though Pearson recognises that "there are many statements in the corpora which match the above patterns for formal definitions but which are not themselves formal definitions". The implication is that the false positive rate is zero; this is difficult to credit, and again suggests that an automatic system was not in fact used.

However, Pearson does seem to recognise that her approach does not in fact retrieve all the definitions within an explanatory text, because she states that the investigation "has shown that it is possible to retrieve *at least some of* the definitional information which would normally be collected through consultation with subject experts" (italics by author of this thesis). Pearson's work also supports the suggestion that definitions will be rare in reports between domain experts (see discussion in section 1.1.3.3), since her study of articles from the journal *Nature* showed far fewer definitions than from the other two corpora used (ITU, GCSE), which were less specialised in domain.

Table 19 details all the manually identified definitions together with reasons for non-extraction where appropriate. We shall not examine the other three relation types separately in the above detailed fashion; it is more useful to consider reasons for non-extractions by category for all four relation types.

5.3.6.4 Concept Non-Recognition

One of the major causes of failure to extract a real relation instance is that of non-recognition of the concept as a TT. For definitions, one of the available 12 instances was not extracted because the concept being defined was not recognised as a TT in the sentence ("map-to-see"). Resolving this would raise recall for definition to 6/12 = 50%. For "map-to-see" there is an obvious solution: examination of the sentence reveals the phrase *the concept of a "map-to-see"*. This phrase is telling us directly that *map-to-see* is a concept. Thus a future KEP function could look for introductory phrases such as *the concept of X* or *the X concept*, where X is a concept in the text. This method has the advantage of working for concepts of all lengths, including one word.

Although TT non-recognition can happen because the concept, (although of the correct part-of-speech pattern) occurred only once in the text ("map-to-see"), or for reasons connected with code not yet written (sentence numbers for terms derived from hyponyms are not yet recorded against all single-word terms), it often occurs because the concept was a more complex entity not conforming to the Justeson and Katz tag patterns or derived hypernym patterns (which are of course shorter and simpler). It is wasteful to lose a good relation extraction and so future studies should examine these cases with a view to detecting patterns of usage that might be simply resolvable. For example, concepts such as *the use of TT*, where TT is a term already identified, might easily be spotted and hence added to the glossary output.

However, it is probable that the majority of instances will not be simple cases of TTs found within longer strings; in most cases they will be TTs of more than 3 words in length (not yet catered for), or more problematically, concepts not actually present word-for-word in the text.

Concept non-recognition was more of a problem for exemplifications than for the other relation types. Many of the examples found manually in 'B1G' concerned complex concepts not present explicitly in nearby or preceding text. Here is a typical example:

> Where the source zones nest hierarchically into the target zones, for example UK administrative EDs nest exactly in wards, transfer of data from the source units to the target units is one of simple aggregation.

Here, although the example is indeed contained within a single sentence, the concept being exemplified is "the hierarchical nesting of source zones in target zones", a complex concept not present in this exact lexical form in the sentence. Clearly the TT concept validation method is inadequate in such cases, for not only are the concepts not TTs in the text, but in addition they are not even present as a lexical string.

The vast majority of manually-identified single-sentence exemplifications in 'B1G' concerned long or complex concepts (at least 40 of the 50). Even where the concept was present in the text in the exactly correct lexical form it was often too long to be a KEP-type TT. Why do exemplifications suffer most from this situation? The reason must lie in the purpose of the relation. Let us recall earlier discussions on example types (in section 3.3) – in particular the *positive examples* of Mittal and Paris (1993), which play an 'elaborative' role. It is clear that examples are often used to aid in the explanation of a concept. The need to do this is greater for the more difficult (complex) concepts, because these are the ones that a reader will have more difficulty in understanding. A good writer intuitively knows this, and provides examples to aid in reader understanding. Thus the exemplification relation thrives on complex concepts.

Studying the instances of exemplification in 'B1G' in such detail has convinced the author of this thesis that examples are a fundamental coherency-creating device used in explanatory texts. By elaborating complex concepts introduced as a result of a reading of a section of text, they bind that previous text together over long distances (i.e. several sentences). Examples may themselves take several sentences to describe, and in fact there were many such multi-sentence examples discovered manually in the test text. Examples are fundamentally concerned with concept *understanding* – this is their purpose. On hindsight it is hardly surprising, therefore, that a simple, shallow, pattern-matching technique has difficulty in extracting them.

5.3.6.5 Amalgamator Failure Rate

There were no amalgamation failures in the 'B1G' test, i.e. cases where there were several candidates found for a correct extraction, and where one of these candidates was the correct one, but where the

amalgamation code (see section 4.6.12) returned the incorrect one. At first sight this appears a remarkable result, and could be taken to imply that the amalgamation algorithm is extremely good. However, although the amalgamator code did in fact work well in all cases when called upon, much of the credit for this situation must come from the fact that the vast majority of correct extractions arose from cases where *only one* candidate was put forward (and hence where the amalgamation stage was not in fact called). For example, for definitions the mean number of amalgamation candidates for correct extractions was 1.25. This figure shows that the amalgamation code was not frequently invoked, since it is close to an average of one (i.e. one single candidate extraction per correct extraction reported).

This situation itself is good, however. It shows that the basic idea of cutting sentences into sections according to lexical patterns specific to various conceptual relations is sound. It might well have turned out that many candidate extractions arose for each attempted extraction, which would have required heavy use of the amalgamator and which would have implied that the idea of cutting-up sentences using a simple pattern-matcher was in some sense inadequate. Heavy use of the amalgamator code would also have created many chances for the wrong candidate to be chosen, and might thereby have reduced precision to a miniscule rate. This has not been the case – in the large majority of cases KEP decided that there was only one possible way of cutting up the sentence so as to give a valid concept plus its elucidation. Thus the novel pattern-matching approach specific to individual conceptual relations does appear to be a valid one.

5.3.6.6 *This*-anaphora counts

Manual inspection of the source text revealed eight cases (Definition: 3, Partition: 1, Exemplification: 4, Hypernymy: 0) where a *this*-concept took part in a correctly-extracted relation instance (counts include *this*, *This*, *these*, *These*). In two of these cases, the antecedent was a known TT (or TT-like in form) in the same sentence, but in no cases was it a more complex construct in the same sentence. In three cases the antecedent was a known TT (or TT-like in form) in the preceding sentence, and in three cases it was a more complex construct in the preceding sentence. Surprisingly, in no cases was the concept further back in the text, either as a simple TT-like form or as a more complex construct.

These results are probably too few in number to generalise from, but indicate that although the antecedent is often in the previous sentence, it is not always a simple TT that can be looked for and snipped out. Where it is not a simple TT form, it would be acceptable for some applications to return the entire previous sentence as the concept being defined, exemplified etc. However, this is not likely to be a useful strategy for the glossary output, where a short concept is placed in the middle column. Furthermore, the act of finding out whether the antecedent is simple or not is surely part of the resolution process i.e. inseparable from it.

This-concepts make up 8/35 = 23% of the 'useful' extractions reported. This is not an insignificant fraction of the reported extractions. Because KEP does not resolve *this*-concepts at present, they cannot appear in the glossary. Thus in the case of text 'B1G' one fifth of the useful extractions made by KEP did not appear in the glossary. This is a waste; resolution of *this*-anaphors must remain a high-priority task for future research.

5.3.6.7 Effect of Ellipted Material

Ellipsis alone was responsible for no failures to make an extraction from 'B1G'. Clearly, it can and does occur, as the "food poisoning" example given previously shows (see discussion of hypernym results in section 5.3.6.2), but in such cases other difficulties prevent the extraction long before ellipsis becomes critical. Ellipsis is likely to be more critical when it occurs in the concept part of the extraction, since concepts must be matched against the TT list (elucidation parts are usually longer sections of text which are understandable by the reader even if they include ellipsis). KEP does not presently contain functions aimed at restoring ellipted text; this would represent a large area of research in its own right, akin to the resolution of anaphora in its complexity. For these reasons ellipsis has not been assigned a high priority for future research, although of undoubted linguistic interest.

5.3.6.8 Effect of Fronting, Cleft Sentences and Embedded Phrases

Fronting, cleft constructions and embedded clauses did not prevent any extractions from taking place. Patterns can be devised for most fronted and cleft constructions, but where a concept is dispersed due to an intervening phrase there would be problems. However, since KEP recognises concepts only if they are technical terms, and such terms are by their nature 'lexical' i.e. not usually split, this situation does not currently arise. Should the anaphora- resolver be developed, however, this might not be the case. For exemplifications, where an anaphoric 'this' is often used as the concept, the antecedent is often very complex (as has already been discussed). In such situations the complex antecedent might well be interrupted by subordinate phrases. This work is outside the scope of the current research.

5.3.6.9 Missing Tokens and Templates

Some extraction failures arose simply because the correct extraction templates and tokens were not present in the external pattern-matcher files. For definitions, as shown in Table 19, there were 2 such occurrences. For partitions, there were 9. These are in theory easily corrected by the addition of extra tokens and patterns. However, KEP currently has a 16-token limit for any one sentence and so the addition of many new tokens might cause the non-processing of other good sentences. For this reason the change in precision and recall are not easily calculated without re-running KEP after the new additions. Having said this, assuming that adding the new tokens and templates would not cause any new token-limit breaches, then the recall for definitions would rise to 7/12 = 58%, and for partitions to 12/14 = 86%.

In fact, the 16-token limit turned out to be a very minor factor in failures to extract relation instances. Although there were several occasions where the 16-token limit was breached, in most cases a higher token limit (as high as necessary) would not have changed the outcome. Only 10 extractions in total (for all four relation types) were missed *solely* due to KEP abandoning sentences having more than the maximum 16 tokens in them. This low figure indicates that the correct balance has been achieved between the token limit (and hence processing time) and the token sets chosen. Token numbers sometimes rose as high as 25, but these were for unusually long and rich sentences, often without any relation instance in them to be extracted (despite their being triggered). Thus there does not appear to be a need for a higher token-number limit (and hence the requirement for much longer processing times per sentence – see discussion surrounding Table 9). This is an important point – the exponential nature of the tokeniser means that processing times potentially double for each additional increment of the token limit.

5.3.6.10 Apposition False Triggerings

Appositive definitions etc can occur signalled only by punctuation such as paired commas or brackets. (Appositive triggers are not looked for until after other triggering methods have failed.) Since very many sentences contain such structures, a high false triggering rate and hence a high false positive rate may arise. A sentence will not however be processed after triggering unless it also contains a TT, because a sentence which does not contain a known TT can never give rise to an extraction with a good concept. However, sentences may be triggered for an appositive structure and yet contain a TT even though no relation instance is actually present.

In 'B1G' there were 14 false positive extractions arising as a result of appositive false triggerings. These instances represent only those cases where an extraction resulted from an appositive triggering. Many more appositive triggers were in fact detected which did not give rise to an extraction in the end. This glut of triggerings leads to a significant increase in total run time (approximately doubling the run time in this case). For this reason, KEP has been provided with a user-choice to allow apposition triggers to be ignored. Since in the evaluation run there were in fact only 2 cases (for all four relation types) where an apposition structure contained a good definition, example etc (this being determined by comparing outputs from 'apposition-on' and 'apposition-off' runs), turning off apposition triggering in this way would hardly have affected recall, whilst raising precision noticeably due to the reduction in bad extractions. Thus the 'B1G' experience indicates that it is best not to search for apposition structures at all, because not searching for them (1) has little effect on recall, (2) gives a significant rise in precision, and (3) allows a much shorter run time. It is of course conceivable that in a different text, where for example an author was particularly fond of appositive definitions (such as the implicit definitions described by Selinker, Trimble and Trimble (1976) and Darian (1981)), this omission might give rise to poor performance on recall. However, ad-hoc tests performed on other BNC texts do not seem to indicate that this will be a major problem.

Ultimately, the apposition problem must be tackled if 100% recall is to be achieved. It is similar in scope to the *is a* problem, which is discussed in the following chapter, and therefore solutions may also be similar to those later suggested. Apposition itself will not however be considered further in this thesis.

5.3.6.11 The Sparse Nature of the Glossary

Note that there were not many extractions to be found in the text B1G (12 Definitions, 12 Hypernyms, 50 Exemplifications and 14 Partitions as determined by manual inspection). Thus even with a perfect KE program the glossary which would be produced would be sparse in the sense that few of the middle-column entries would have 3rd-column explanations to them. This is to be expected; it is rare, even in explanatory text, for an author to define, exemplify etc all concepts used. (This is, of course, why glossaries are needed. One might therefore argue that it is not enough to base a glossary on the contents of the text alone. However, this argument is not valid since definitions etc may occur in sections of text not yet read by the reader, who wishes to check the encountered unknown term without having to hunt through the rest of the text for its possible definition. This mode of glossary usage is particularly appropriate where a reader is "dipping into" a large text.)

With the less-than-perfect KEP extractor, the glossary becomes even sparser, because (1) some good relation instances are not extracted, and (2) *this*-concepts are not yet resolved. However, this is partially offset by the addition of "useful" extractions placed in the third column. (Note that such useful extractions could be made to look less wrong by removing the relation specifier printed before them – the 3rd column entry then merely looks like information "about" the term i.e. it loses the granularity of the separate relation types.) Part of the glossary output for file B1G is reproduced in Figure 20. Notice that only one entry from the chosen page of the glossary has 3rd-column text. In fact, many pages have only middle-column entries (terms), perhaps with the odd acronym. This demonstrates the sparse nature of the glossary.

It would be possible to create a "condensed" glossary, in which only those terms having 3rd-column entries were printed. This might prove to be a way of identifying the most important terms in a text (i.e. they are the most important ones because they have been defined etc). However, it is thought preferable to allow the user (editor) of the glossary to make the choices as to which TTs are to be deleted (if any) and to add explanations to TTs as seen fit; in most cases the user will want to add explanations rather than delete the entry altogether. It is better that possibly-good material is provided, than possibly-good material is omitted, since deleting an entry is a very simple operation (decision plus action) whereas devising and entering a new term and its explanation is an 'intelligent' time-consuming operation, requiring a reading/understanding of the whole text. Making life easier for the WP user is, after all, the purpose of having an automatic glossary maker.

	data source	
	data storage	
	data structure	
	data uncertainty	
	data volume	
	data	
	database for hazard	
	database management system	
	database management	
	database view	
	database	
DSS	decision support system	Parts: data storage files, data analysis models and display and interactive use technology. SEE ALSO data analysis, data storage, interactive use, data, analysis, use, model
	decision support	
	deep repository	
	defence system	
	derived polygon	
	design	

Figure 20. Part of the Glossary Output for BNC text 'B1G' after full evaluation run

5.3.6.12 Concluding Remarks on 'B1G' Evaluation

The important issue is that KEP should be able to extract as many of the present relation instances as possible. The precision and recall figures given in Table 18 fall short of the ideal 100% but show that KEP succeeds reasonably well at this task, and indeed especially well if "useful" extractions of any sort are counted. KEP seems rather poor at examples but particularly good at definitions.

These evaluations have demonstrated that KEP's novel pattern-matching approach to conceptual relation extraction is a valid one which can do useful fact extraction. However, they have also demonstrated that the approach will never achieve 100% precision and recall due to: (1) the need to examine context in order to confirm the presence of a specific relation type, (2) the inability of the pattern matcher to cut up sentences around TTs in some cases (this is discussed in the following chapter), (3) self-imposed time constraints arising from the 16-token limit, and (4) non-recognition of concept-TTs for long or "constructed" concepts. Point (1) arises for a sentence within the context of a surrounding group of sentences, and also for an appositive phrase in the context of its surrounding sentence. Points (1) and (4) may well require semantic and pragmatic processing in order to resolve them, although in some cases a simpler method based on WK (e.g. a MR thesaurus) may suffice. Points (2) and (3) are a matter of software and technology and there is no reason why progress should not be made for these, with the caveat that in point (3) the exponential nature of the process will inevitably impose some limit, albeit higher than at present.

Despite the fundamental limitations of the approach demonstrated in points (1) and (4) above, it is clear from these evaluations that KEP can and does provide a useful glossary output. The output is a good starting point for the construction of a glossary for an existing document that does not have one. Most of the essential concepts within the text are found, and many of the clear-cut cases of definition, exemplification etc are presented. Where facts do not fall exactly into the chosen relation category they often still present useful information about the concept which an editor might want to use. Most expanded acronyms are correctly reported and linked to 2nd-column concepts. Cross references where provided are useful and yet not overly numerous, and the whole glossary is correctly sorted alphabetically. Furthermore, although the proto-glossary so produced is somewhat sparse in the sense that there are not many 3rd-column entries, only a small fraction of it comprises bad lines which need to be deleted rather than expanded upon.

5.3.7 Plural Noun Singulariser

The sing() function has been described in Chapter 4. A detailed description of it has been given in Bowden, Halstead and Rose (1996c) where its performance has been evaluated. This function has a precision rate (i.e. (number of plural nouns correctly singularised / number of plural noun singularisations attempted) * 100) of better than 99.9%. (Recall is not appropriate, since every word passed to sing() is assumed to be a plural noun - the function does not test this.) Full details of the development and evaluation method are given in the paper and so will not be repeated here.

This function is a vital part of the KEP system, for without it the technical term extractions would not be possible. It is used in several places, wherever there is the need to compare nominal items of unknown number (*s.* or *pl.*). However, very occasionally it is found that it has apparently failed. In most cases of this detected so far it has transpired that the sing() function did not in fact fail; the cause of the error was

a bad tag in the input text. Where TTs are reported in the plural form, this bad tagging (of a plural noun as a singular noun) must have occurred at least twice in the text. Manual correction of the bad tags by editing the source text causes the plural term to go away, confirming that sing() has indeed performed correctly.

Very rarely, however, KEP fails where there are two possible singular nouns for the input plural – such as for the word *bases*. Although an interactive version of KEP returns both possible singular forms to the user (*basis* and *base*), the version of the function built into KEP has to make a decision on which is correct. Currently this is done without any intelligence – the first version is always returned. There is one example of this failure in the glossary given in Appendix E, where KEP has invented the term *knowledge basis*. The source text shows that this arose from the words *knowledge bases*, i.e. plural of *knowledge base*. This situation is so rare that it has not yet been addressed in the code. However, the author has considered ways of resolving the problem, and the use of context for doing so has been discussed in Bowden, Halstead and Rose (1996c).

Although it might seem unnecessary to develop a self-contained function to perform plural noun singularisation, there are good reasons for doing so. Firstly, no external machine-readable dictionary (MRD) is then needed. Since KEP does not use one for any other purpose, it would be unwieldy to introduce one simply for this purpose. MRD look-up requires open and closing of files, searching, and possibly morphological processing to obtain the singular form. The latter may not be simple - most HRDs (human-readable dictionaries), from which MRDs usually derive, contain the singular forms of nouns with an indication of how to form the plurals. For example, consider the (simplified) entry *phenomenon (n), pl. -na*. Here a human reader has no difficulty in realising that the plural form is made by matching the *-na* element to the end of *phenomenon* so as to make *phenomena* and not *phenomenona*. This is a difficult task in itself, but worse still, most dictionaries use different conventions with different nouns; with *dog (n), pl. -s* there is no matching needed and the *s* is simply appended to the singular, but with *series (n), pl. series* the whole plural word is stated. Thus a complex set of rules may be needed to generate the plural. (The direction of transformation, from singular to plural or vice versa, is not a problem, however, since KEP could just as easliy have been written to work in this direction i.e. instead of a sing() function there could have been a plur() function using the MRD.)

Not only does the above describe a *complex* process, which may in practice be difficult to get right in all cases, but also it describes a *slow* process. Opening and closing of files via system calls, searching, and the necessary morphological processing all take time. Since KEP may need to call the sing() function tens of thousands of times during the processing of a single text, a fast sing() function is highly desirable. The standalone function developed for KEP is capable of returning the singular form in a time which is for all practical purposes instantaneous.

A further reason for developing a function such as KEP relates to the linguistic interest. Evaluation in Bowden, Halstead and Rose (1996c) has shown that the most commonly used nouns with non-*s* plurals are almost without exception words long established in English, which is of interest to historical linguists. (Words such as *feet*, *lives*, *wives*, *teeth*, *leaves*, *geese*, *knives*, *calves* etc fall into this category). Also, actually writing a function such as sing() forces the designer to state all the possible rules and exceptions for creating plural forms in British English, a categorisation which may not have been stated explicitly in as much detail before.

Noun plural formation is a small but interesting area of NLP research. Wothke (1986) discusses the automatic learning of English noun plurals from singular/plural pair examples in a learning corpus. Wothke describes a program (the PRISM system) which takes pairs such as *city/cities*, *house/houses* etc and induces the rules required to pluralise (rather than those to singularise). Wothke has essentially proposed automating what was done manually when developing the sing() function. However, the scope of PRISM is broader since it is not confined to nouns; *any* derivations, such as finding the negative versions of adjectives, are targeted (e.g. *perfect* becomes ***im****perfect*). Since this approach depends upon the quality of the training corpus, good performance for less frequent forms (e.g. *mouse/mice*) is dependent upon the system having seen the exceptional case during training rather than through human introspection and subsequent rule-coding. The difference between sing() and PRISM is ultimately that in the case of PRISM the system itself infers the rules from pairs of words, whereas with sing() this inference function was performed by the author of this thesis. Although the end result is the same, Wothke's interest lies mainly with the inferencing mechanism, which is not of immediate concern here.

In Bear (1986) a theoretical approach is taken to lexical morphology. Bear distinguishes between "lexical characters" and "surface characters", where the word "lexical" is being used to indicate the form of a word based upon morphemes and their functions, and where "surface" means the actual spelling used. Thus for the string of lexical characters [b o x + s] there is the string of surface characters [b o x e s]. Here it is seen that a morpheme boundary at the lexical level (+) corresponds to an **e** at the surface level. Bear provides a formalism for writing *rules* which show how the lexical level provides the surface level forms. So if + lies between an **x** and an **s**, the rule might be:

R1) + -> e {x | z | y/i | c h} _s

This rule is read as: Rule number 1: a morpheme boundary (the + symbol) at the lexical level corresponds to (the arrow ->) an **e** at the surface level whenever (the braces { ... }) it is between an **x** and an **s**, or (the vertical bar |) between a **z** and an **s**, or between a lexical **y** corresponding to (the forward slash /) a surface **i** and an **s**, or between **ch** and an **s**. Several rules may be required to form the plural form for a singular noun, applied in a manner so that the result is correct. This is discussed in

some detail. Bear's contribution appears to be mainly a formalism for writing rules, such as the rules for forming plural nouns from singular nouns, together with methods of applying the rules.

The formalism is designed to be code, function, and language independent - it is a framework for a general lexical morphological analyser which would work together with or as part of a parser. However, it is not clear that Bear has actually built a system using his formalism. Thus it is not clear that it would be able to pluralise nouns with a high success rate. The rules built into sing(), however, do work. Thus although the works of Wothke and of Bear are interesting, the problem of plural noun singularisation has in practice been solved for KEP by the sing() function. There are no plans to develop sing() further, although an interesting future study might be to write the corresponding plur() function. This would reveal whether it is indeed simpler to go from plural to singular rather than from singular to plural.

5.4 Processing This Thesis using KEP

5.4.1 Introduction to the Thesis Test

Although the above evaluations demonstrate KEP's performance, both in terms of its individual functions and overall, they may not convey to the reader of this thesis the full flavour of KEP's text processing abilities, since the reader of this thesis has probably not read the individual BNC test texts used. What is required is a large text which the reader has indeed read. The ideal candidate is of course this thesis itself.

Therefore, as a concluding demonstration of KEP's abilities, a large part of the body of the submitted version of this thesis was processed, i.e. chapters 1 to 4 inclusive[13]. The WORD file containing this thesis was saved as plain ASCII text (with line breaks), copied from PC to the UNIX machine running KEP, pre-edited (using the 'vi' editor) to remove extra ctrl-M characters (placed there by WORD) and to contain the single **<text>** and **</text>** SGML pair of tags required by CLAWS, tagged using CLAWS, and pre-processed by **conclaws.c** (see Table 4) into the **kep.in** default KEP input file. KEP was then run for glossary production. Term acquisition was run with full look-ahead, and elucidation extraction was run with apposition triggering turned *off* (see earlier results regarding the lack of usefulness of apposition triggering). The full glossary output is given in Appendix E.

5.4.2 Thesis Test Results

The duration of the thesis test run was approximately 60 hours on a lightly-loaded machine (SPARCStation 10, running SunOS 4.1.3). The length of the untagged ASCII input file was 60,145 words on 4,695 lines (372,305 characters). KEP converted this input into 2,737 sentences.

[13] It is stressed that this exercise is not intended to represent an unbiassed evaluation of KEP, since the creator of KEP also wrote this thesis. The purpose is merely demonstrative.

It was not the purpose of this demonstration to repeat the detailed evaluations of the above sections. Rather, the aim was to provide a positive demonstration of what KEP can do, and to show that the output does indeed form a good basis for a glossary for this thesis. The output given is "warts and all", and so reveals both the successes and the false-positives, together with a handful of minor bugs. We shall consider the performance of individual glossary functions.

5.4.2.1 Acronyms (First Column)

Almost all of the reported acronyms/expansions are correct - 57 out of 58, giving a precision of 98%. (Where the minor error of wrongly reporting the expansion as a plural occurred, these were counted as correct. There were 5 such cases, arising because the expansion was given in the text in the plural whereas the acronym itself was singular i.e. without a final lowercase 's'. Although not a problem for a human reader, these cases can cause problems at the TT-linking stage i.e. the plural and singular expansions are listed as separate TTs.) The one bad acronym was a false positive i.e. was not an acronym in the text. There are 69 acronyms with expansions given in the input text, so recall was 57 out of 69 or 83%. These figures are very good but probably reflect the author's systematic use of brackets – 36 of the reported acronyms were bracketed, and 7 expansions were bracketed. However, since in no case were *both* expansion and acronym bracketed, this means that the acronym extractor found 57 – (36 + 7) = 14 correct acronyms not signalled by bracketing in some way. Of the reported acronyms, 16 were non-exact, and 1 was a single compound word made from hyphenated words (OO from object-oriented), although hyphens did occur linking some of the words in 3 other cases (all correctly reported). Sixteen of the reported acronyms were not expanded near their first use, which at first surprised the author, but which on investigation turned out to be cases where the acronym appeared in a heading before its introduction in the following text.

KEP correctly rejected IV as a roman numeral in the context of its use. However, it failed to find expansions for NL, MUC, PS, CD, MR, FS, CPU, XSQL and LLC, despite the fact that all of these were expanded somewhere in the text in a form which a diligent human reader could have spotted. Two of the omissions were blatant – PS and CD, both of which were placed in brackets after their expansions. Clearly, two-letter acronyms require enhanced processing. However, this will not simply be a dropping of the reporting threshold, since this would also increase the false positive rate. (It is conceivable that some acronyms were not extracted because the expansion occurred in a "too-long" sentence, which was not examined by KEP after being labelled as such; there were 44 of these sentences. This information is available in implicit form in the long output plus input text.)

Overall, KEP has made a very good attempt at finding and expanding the acronyms given in the first four chapters of this thesis. Precision was 98% and recall was 83%. About one entry in every 19 in the glossary has an acronym in column 1.

5.4.2.2 Technical Terms (Middle Column)

KEP reported 1,078 technical terms. Of these, 115 were deemed to be 'bad' by the author, i.e. about one in 9 reported terms was probably not a valid TT for the document. In the following discussion, each bad term has been placed into a single category which defines what was perceived to be wrong with the term. This was a subjective process and at times it was difficult to decide which category to place a bad term into. However, the resulting lists do give an reasonable picture of how common each type of error was.

Some of the bad terms were 'duff' as defined earlier (14 of them). (Other duff terms such as **recent year** and **other method** had correctly been removed by the duff-term detector.) There were also 29 2-word terms arising wrongly due to 3-word terms (and 3- from 4- etc), such as **interbank money** from **interbank money transfer** (see discussion in section 5.3.2.2). Clearly, this is a serious problem for the chosen term identification method and steps need to be taken to reduce the incidence of such terms. Of the 188 terms involving a preposition (i.e. found using a pattern of the form NPN), 10 appear to be bad (e.g. **case of terms**), although some bad NPN terms were counted in the previous category rather than in this one, so this rate might actually be higher than 1 in 19. For the hypernymic 1-word terms, 20 out of the 125 reported are probably not TTs for the text because they are just too general in scope (**way**, **type**, **amount**, **item** etc) although most are definitely good in the context of KE (e.g. **sentence**, **language**, **meaning**, **sense**, **phrase**, **information**, **knowledge**, **syntax**).

The remaining 42 bad terms were difficult to categorise under one of the above headings. They include terms such as **s definition**, which arose because apostrophe-*s* is separated out as a word by the CLAWS tagger, and terms such as **just name** and **phrases example** which meet the Justeson and Katz A/N patterns but which are obviously not good. A few bad terms were genuinely mysterious, such as **purpose wheres**, which may have arisen through bad pre-processing by the *conclaws* program. In a few cases, terms appeared as separate entries in both singular and plural forms, due to incorrect acronym expansion reporting or other reasons (see above), e.g. for **technical term** itself, which appeared next to **technical terms** (which had the acronym *TT* attached). In one single case the sing() function failed and created a bad term, namely **knowledge basis** from **knowledge bases** (it should have been **knowledge base**), although the correct term was also present.

Overall, KEP has reported 1,078 TTs, of which 963 appear to be good for the text used. This represents a precision of 89%. In other words, only about one tenth of the TTs reported in the glossary are bad, and hence would need to be deleted by a human post-editor. This is regarded as an acceptable level of false-positive reporting.

5.4.2.3 Explanations (Third Column)

As expected, the third column was sparsely filled. The text provided is mostly useful, and the correct definitions etc are particularly pleasing. There are however incorrect extractions, and these tend to stand out in an obvious fashion. This is, however, no bad thing – it is much better to have easily-identified errors than subtle ones which are therefore not removed by the human post-editor. It is very difficult to provide reliable counts for the precision in the 3rd column, since there are many factors to consider (e.g. do we only look at entries for terms which we deemed "good" in the 2nd column, or do we look at them all? Do we count both "useful" and strictly-definition etc cases as good, or count these separately? Do we mark the whole entry as correct/incorrect or count its individual definitions etc separately? etc).

Instead, some general observations can be made. Firstly, definitions are much more common than the other three relation types in the glossary output given as Appendix E. To some extent this is a result of the order in which the four relation types are combined to form the explanation (D,H,E,P) since code exists to prevent repeated explanations for one concept (e.g. where the exact same elucidation occurs as both a definition and a hypernym for some concept, the hypernym elucidation is *not* added to the explanation by the Extraction Combiner). Thus one might expect there to be some suppressed hypernyms, examples and partitions, but no suppressed definitions; this is another reason why it is not a good idea to make precision counts from the glossary output, but rather from the long output file. KEP does not currently count the numbers of suppressed elucidations (either in total, or separately by type H, E and P), although this would clearly be a useful enhancement. However, having written the input text, the author of this thesis believes that the preponderance of definitions in the output probably does reflect to a close degree their relative numbers in the source, even if many of those given are not in fact strict definitions, but "useful" facts relating to the associated 2nd-column term.

The explanation column demonstrates the difficult problem of reporting specific instances as though they were generic. Several of the explanations clearly refer to specific programs, systems, items etc and not to generic examples of such things. For example, the definition entry for **pattern matcher** actually refers to the manual approach of Ahmad and Fulford (1992), rather than to pattern matchers in general. As has been discussed elsewhere in this thesis, this is a difficult problem to overcome in a shallow "text cutting" system. In many cases, however, the specificity does not appear to matter, since it is directly relevant to the content of the input text. For example, the explanation for **bare template** is highly specific, and yet is actually a perfect description of the bare templates used by the Preference Semantics approach as described in this thesis.

Many of the explanations given, although not strictly definitions, examples etc are nevertheless "useful". This phenomenon has been discussed earlier. In such cases a human post-editor could quickly make them good; one way of doing this would be to introduce categories such as *Usage:* or *Characteristics:*

alongside the existing *Definition:*, *Examples:* etc. For example, the entry for **information** is given as "distinguished from knowledge in that it is intended to be used within a short time after its reception" – this might be better described after *Characteristics:* rather than as a *Definition:* of the concept. The entries for **human reader**, **generic fact** and **british national corpus** (amongst others) also give characteristics. The entry for **sentence number** might be better described as a *Usage:*, this also having the advantage of negating the specific nature of the explanation given ("used widely in screen and file output"). An alternative approach would be to produce a glossary in which the individual relation types were not mentioned at all, these being replaced by a simple dash. This then leaves it up to the reader to decide what kind of knowledge is being presented. However, this approach would seem to negate the effort of finding the explanations separately by conceptual relation, and in some cases might not be appropriate (e.g. where a list of examples was presented, although even here a human reader would probably realise that an explanation such as "- PASCAL, FORTRAN and C" was a list of examples for a hypothetical concept like **third generation language**).

The output in Appendix E does however include several fully successful extractions, i.e. where the explanation given really is a definition , list of examples, hypernym or list of parts. These may be of two or three columns filled (2nd and 3rd columns, all three columns). There are excellent entries for the concepts **artificial intelligence**, **deep technique**, **fact** (with removal of two words), **glossary**, **historical text** (one definition, one set of characteristics), **knowledge extraction** (two definitions, one set of characteristics), **negative example**, **process**, **set** (one set of characteristics, one definition), **shallow technique** (one definition, one usage), and **token**. Many other concepts have long 3rd-column entries which contain perfect definitions etc amongst less correct text.

5.4.2.4 Cross References (Third Column)

The "SEE ALSO" cross referencing has worked well. The cross references are not intrusive due to their number and do indeed point the reader to other terms in the glossary. The expansion of acronyms (given as part of the third column entry) in the "SEE ALSO" part appears to be particularly useful, although as the writer of this thesis perhaps the author is not best placed to judge this.

Because cross references are made using string-within-string operations, acronyms within acronyms are reported. This is usually good, because very often where this occurs the longer acronym does indeed utilise the shorter one within it (e.g. KEP / KE). However, it occasionally causes a bad cross reference. For example, the reference to **information extraction** in **TIE routine** is incorrect. In general, however, the cross reference precision and recall are both very high, i.e. almost all of the cross references given are correct, and there are very few omissions.

There are several 3rd-column entries which comprise nothing but a SEE ALSO string, i.e. where the 2nd-column concept itself contains an acronym but where no definition etc was found for the concept. These cross references are in general helpful, but in some cases following them does not lead to an increase in reader understanding of the original concept. KEP does not currently check that the target of the SEE ALSO string does indeed have an explanation attached to it. Although one could arrange for such "dangling links" to be omitted, this is not thought to be a good idea because the glossary is designed to be post-edited. When the editing has been done, the cross references will probably no longer point to an unexplained term. The only disadvantage of this approach is where the human editor decides to delete an entry altogether; in such cases he will also have to search the entire glossary for now-useless SEE ALSO entries to the deleted term, so that they themselves may also be removed. However, if the post-editor program were part of a complete system (e.g. one in which the original section of explanation text could be brought up in a window for cut-and-paste operations, as suggested earlier) then it would be possible to build an "intelligent term delete" operation into the editor, capable of automatically removing all defunct cross-reference text on deletion of the referred-to term.

5.4.3 Concluding Remarks on the Thesis Test

It is clear that the KEP program has made a good attempt at producing a glossary for the first four chapters of this thesis. The error rate is acceptable and the glossary obviously captures the main specialist terms used. If one were to pick up this glossary not having first seen the body of the thesis, one would be able to deduce that the source document was to do with natural language processing, technical terms, acronyms, knowledge extraction, and a novel program called KEP (witness the very large amount of explanation text for KEP, KE etc). Although there are occasional errors, and the third column is rather sparsely filled, a few minutes of post-editing would transform the glossary into an accurate and comprehensive document. Production of the glossary using KEP's assistance would have taken a considerably shorter amount of time spent by the writer *at the keyboard*, especially if the writer had left the decision to create a glossary until after the document had been written.

However, the long time taken by KEP to compile the glossary would have required the user to start the process and then retire to some other activity whilst it was being made. This is not necessarily a problem, but since earlier in this thesis negative comments were made regarding the utility of programs which take a long time to run, it is necessary to defend this statement. Several points contribute towards this defence. Firstly, a major purpose of this research is to determine whether the novel pattern-matching method can in principle succeed. This is a question that can be answered regardless of running time. Secondly, KEP has not yet been optimised for speed (clearly a desirable activity before use as a commercial program). Thirdly, one must not neglect the probable increase in speed which would come from using a faster computer. CPU clock rates are increasing year on year, and are predicted to continue to rise rapidly over the next few years. Given a 2-order of magnitude (100-fold) increase in processing speed, a sixty hour run drops to 36 minutes. A 3-order of magnitude increase drops a 60-hour run to just

over three and a half minutes. Furthermore, as larger amounts of fast memory are incorporated into computers, the need for many time-costly disk accesses decreases. (Indeed, the day may shortly come when rotating magnetic disks are entirely replaced by solid-state non-volatile memory.)

Fourthly, this is not the only hardware-oriented improvement that could speed up KEP. Since KEP processes one sentence at a time, it is theoretically possible to process each sentence in parallel. In the ideal scenario, each sentence in the text would be assigned its own processor, and a copy of the whole text would also be made available to each processor. Each processor would then (1) find all TTs in its sentence, using the copy of the rest of the text for TT look-ahead, (2) find all acronyms and their expansions in its sentence (or preceding sentence), and (3) process the sentence for each of the four relation types in turn (triggering and then extraction attempt if triggered). This last step could itself be parcelled out to four processors so that the four relations too could be processed in parallel – this would represent the most parallel scenario possible. (In fact, TTs and acronyms could also be found in parallel, although linking of TTs to acronyms is required at some stage.) Finally, the controlling processor would build the glossary by merging identical 2nd-column entries (terms) found by the separate processors, and sorting alphabetically. The duration of the entire run should be little more than the time taken to process the sentence with the largest number of tokens and relation instances in it, plus glossary-building time. Thus, given a system with enough processors, the run might never take more than a few minutes, however long the text. Given recent suggestions for massively-parallel architectures based upon thousands of cheap processors, this is not out of the question. Parallelisation also opens up the possibility therefore of increasing the 16-token limit by several tokens, dependent on the maximum acceptable sentence-processing time. Thus extractions lost due to the token-limit might be brought back into the fold.

5.5 Summary of Evaluation Results

The evaluations presented in this chapter have demonstrated that KEP performs well in many of its functions. The sentence-end detector works correctly in approximately 95% of cases, and of the technical terms extracted from a text, almost 90% are good. However, up to 80% of all terms may be missed, or about 50% if only "text relevant" terms are considered. KEP performs exceptionally well when extracting acronyms and their expansions, reaching high precision and recall rates often greater than 80%. TTs and acronyms form the bulk of the glossary produced by KEP, and even without any third-column explanations provide an accurate and practically useful output.

KEP succeeds in triggering on relevant sentences without filtering out a significant number of those which ought to have passed through. Surprisingly, an attempt to then filter out presentational sentences proved to be of little use, due to the scarcity of such cases. Switching off apposition triggering also affected recall figures by only a small amount, but had the advantages of increasing precision (due to a

fall in false positives) and greatly reducing running time. Grammatical issues such as sentence structure (e.g. fronted and cleft sentences) also proved to be of little import.

On the actual extraction performance, KEP's success rate varies considerably with the relation type targeted. KEP is particularly good at finding and extracting definitions, but its performance on the other three relation types is mixed. Examples proved particularly difficult to extract, probably due to their essentially complex usage. Relaxing strictly-defined categories to allow all "useful" extractions to be counted showed that KEP is able to find many useful facts from a text, and link them to the correct term. Glossaries produced by KEP are sparsely filled in the explanation column, but this is to be expected for a system which does not use an external knowledge base.

KEP's performance has been exemplified using a large part of this thesis as its input. This demonstrates the usefulness of the output and shows that the novel pattern-matcher used within the program performs well. Although KEP currently takes a long time to run on a large file, it has been argued that this is not a relevant factor in this research.

In the following chapter, further discussions of KEP's performance as revealed by the above evaluations are provided, followed by consideration of possible future applications not yet discussed in detail.

6. Discussion and Future Directions

6.1 Introduction

In the previous chapter the performance of KEP was evaluated both in terms of the individual functions and overall for glossary creation. In this chapter the performance of KEP is further discussed, together with possible ways to improve it. The aim is to consider whether the novel pattern-matching approach taken may ultimately be enhanced to the point at which it reliably extracts all of the terms, acronyms and facts from an input text, so that it may be routinely used, for example, as part of an advanced word processing system. Do the limitations of shallow systems such as that embodied in KEP mean that they are doomed to remain below a "glass ceiling" of achievement? Will it ultimately prove necessary to add modules which access knowledge bases? Will full parsing ultimately be required, or can an acceptable level of performance be achieved without it? Is KEP best used as a "first pass" system? These are important questions at a time when shallow NLP systems are *in vogue*. It may well be that all the difficulties relating to deep systems described in the first chapter of this thesis have to be faced - there may be no alternative but to tackle them and solve them. On the other hand, it may be that tasks previously thought to require deep processing turn out to be solvable in a shallow manner.

In addition to the above-mentioned discussions, some practical uses of KEP as it stands (or as it might stand after a conceivable course of development) are considered. These include marking of student essays and automatic building of large knowledge bases.

6.2 Further Discussions and Future Enhancements

6.2.1 Categorisation of Relation Syntaxes

One of the major obstacles encountered by the pattern-matching approach to conceptual relation extraction as described in this thesis has been that of determining which of the relation types is actually present in an extraction. Wrong-relation and no-relation extractions do occur. This problem has been highlighted at various places in this thesis; it arose when discussing the *is a* problem (see also below), the apposition-pattern detector, and in discussions of the nature of definitions, examples, partitions and hypernyms. Let us now inspect the issue further with a view to resolving it, if possible.

Bowker (1995) has pointed out the now-obvious fact that "a variety of phrases can be used to express any relation type" but does not detail these phrases and the relations to which they apply. Should such knowledge be comprehensively collected, its use in a KE system would require inspection of those cases where the same phrase (lexical pattern) was used for several relations. Therefore, the author proposes the following classification of conceptual relation *lexical patterns*:

Class A If present in text, these always indicate the presence of a specific conceptual relation, and nothing else. Example: **is a type of** (hypernymy)

Class B Sometimes indicate the presence of a specific relation, but are sometimes used in ways which do not indicate the presence of any relation type. Example: **is defined by** (definition, and general usage)

Class C Sometimes indicate the presence of a specific relation, but sometimes indicate one or more other relation types. Example: **is composed of** (partition, material)

Class D Sometimes indicate the presence of a specific relation, but sometimes indicate one or more other relation types, or are sometimes used in ways which do not indicate the presence of any relation type. Example: **is a** (hyponymy, definition, exemplification, instance, etc plus general usage - see sentences **b** through **d** in section 4.6.7.)

Within this categorisation scheme the classes with letters closer to the end of the alphabet are those which require more processing in order to decide upon which relation is actually being expressed (if any) and hence to extract the knowledge if it is there. At present, KEP processes one relation type to completion before moving on to the next, and does not attempt to discover whether the syntactical pattern found really does represent an instance of the relation type currently being looked for. Knowledge of the class of the lexical pattern found in a sentence (as above) would be useful: if the class were anything other than Class A, a new function might be called to determine the relation (if any) present, with subsequent processing being dependent on the output of the function. The function would in effect provide the necessary semantic input. Hahn (1989) has also recognised this situation, stating that "The problem with syntactic approaches is that they are capable of performing formal recognition tasks on the syntactical processing level, but significantly lose impact when further semantic processing is required. ... Syntactic approaches are completely indifferent with respect to this kind of distinction and require subsequent semantic filtering of some sort".

Placing lexical patterns into the above framework is important because it should provide not only linguistically interesting data but also directly useable information for relation extraction. Knowledge of a specific lexical pattern that always indicates the presence of a specific relation (class A patterns) would enable us to avoid the need for a more complex (detailed) extraction in that case. If such lexical patterns exist, they should be discovered, so that KEP may use them without further ado. As has been shown in the evaluation for BNC text B1G in the previous chapter, there are indeed such patterns (e.g. *can be defined as* for the definition relation). Table 20 lists all the forms present in B1G (as detected both by KEP and by hand), arranged into the ABCD classes as defined above. In this table, bullet points are used to indicate where a new form starts, since they do not all fit onto a single line.

	Definition	Hypernymy	Exemplification	Partition
Class A	• can be defined as • defined the concept of	-	• An example is • One example is • One example of • An example of this would be • A leading example of • As a second example of • As a last example of • e.g.	• is an essential component in • is a vital element of • second main component of • key components of • contains three main modules viz. • basic components of … comprise • has the following main elements
Class B	• was set up and labelled	-	• For example • for example • As an example • A further example • By way of example • such as	• three components: • contains modules • comprises three interrelated modules
Class C	• essentially use • consists simply of	• is the third most important • is one of the • a classification of	• such factors as • such elements as • include • includes • including • for instance • For instance	• include • includes • is shared by • consisted of • major feature of
Class D	• is a • is an • can often be conceptualized as	• is a • is an • as both a… and • as well as being an	• is a • (…) • like	-

Table 20. Lexical Patterns found in text 'B1G' arranged by Class and Relation

In the table, the lexical forms have been placed according to how the author believes they are used in British English text as a whole, not by how they have actually been used in the text B1G alone. For this reason, the classifications given in the table are not always clear-cut – it is sometimes difficult to place a phrase with certainty into a class because there is always the possibility of a more general usage not brought to mind by the classifier. (Only after an exercise based upon a very large number of texts would one be in a position to base the classification on actual occurrence – to do this for text B1G alone would fail to illustrate the classification method being proposed.) Exemplifications proved the most difficult to classify. One might argue that the phrase **for instance** always introduces an exemplification, and so should really be classed as Type A. However, there are situations where the phrase seems to be used more as a method for smoothing the flow of discourse.

Class A forms allow the extraction of relation instances without danger of getting the wrong relation type (definition, exemplification, partition, hypernymy), and these forms can only appear in one of the top-level boxes. Where the same lexical form occurs in more than one box on any level (which can only occur in classes C and D) there is the possibility of finding the wrong relation type. Where the lexical form occurs in classes B or D there is the possibility of finding a relation instance where none actually occurred (no-relation usages of the forms). This classification scheme is quite complex and might be simplified by reducing it to only two classes, A and non-A. This simplified scheme would still be very useful, since it would allow the distinction between lexical forms which identified an instance of a specific relation with complete certainty, and those cases where there was some doubt about whether a relation was present, and if so, which one it was.

What is to be done about those forms which are not in Class A ? Leaving aside the *is a* problem (class D), which is discussed separately shortly, then there are two tests that a future version of KEP might try. The first test is to attempt to verify that the relation present is indeed possibly one of those recognised by KEP (definition, exemplification, partition, hypernymy). Assuming that this test answers 'yes', (and assuming that one could actually build such a tester – which might be done using a fuller version of the table above) then the second test is to determine which of the relation types it is. This second test makes the assumption that one class can definitely be picked, but given this assumption one can think of possible approaches to the test. For example, certain syntaxes might be tied to one relation type in preference to another, possibly using occurrence counts against each entry in a full version of Table 20. In the phrase *high-level languages, such as PASCAL* one could prefer the existence of the exemplification relation over the hypernym relation, since this syntax makes it more likely that the subject of the sentence is high-level languages (the reader is referred to the discussion in Chapter 4 regarding how a human reader may spot an exemplification). The sentence does indeed tell us that PASCAL *is a type of* high-level language (see Hearst (1992)) but this is obviously not its main purpose.

Beyond this, things become more difficult. Tests such as Skuce et al.'s definition-presence test rely upon the semantic knowledge of a human tester. It is difficult to see how such knowledge could be built into a shallow KE program (i.e. how could one build a KB to ask whether the text answers the question "What is the meaning of ..." ?). However, the simplest cases of relation indeterminacy might be resolved using a MR thesaurus or semantic net KB. If a system had available to it the knowledge that *dachshund* and *corgi* were types of dog, then in the phrase *dachshunds, poodles and corgis are dogs* the existing knowledge of the hypernym relations *dachshund-dog* and *corgi-dog* could be used to infer (a) that the phrase contains solely hypernym instances (rather than examples etc), and therefore (b) the new knowledge that a *poodle is a type of dog*. This example deals with the most difficult lexical form of all, the *is a* form. It demonstrates the usefulness of prior knowledge in recognising the actual relation present.

6.2.2 Resolution of the "is a" Problem

The "is a" problem has been mentioned at several points throughout this thesis. It is the extreme case of the Class D lexical form, as introduced above. The previous section introduced one possible way of resolving what type of relation it is being used for (if any), but this is such an important topic that it should be considered further. Let us consider again the troublesome sentences introduced in Chapter 4:

b A byte is a contiguous group of eight bits.
c A television set is a modern marvel.
d There is a way to do this.

It can be seen immediately that the sentence **d** presents no problems; the existential form may be rejected with a simple negative trigger (*There is a*). The problems relate to the phrases *a contiguous group of eight bits* and *a modern marvel* in sentences **b** and **c** respectively. Why does one of these two phrases give the meaning of something, and the other not? Part of the answer must lie in the specificity of the phrases. Many things could be described as a modern marvel, but very few things as a contiguous group of eight bits. Therefore one way of tackling this problem might be to detect phrases which could be applied to many concepts. Thus simple generalised AN combinations such as *modern marvel*, *wondrous thing*, *important approach* might be ruled out. Note that this is essentially the same problem as the detection of 'duff' terms (see section 5.3.2.2). In addition to the detection of general terms, one could also search for clues that make a term more *specific* to the associated concept. For example, in sentence **b** there is a semantic hint linking the phrase *a contiguous group of eight bits* to the concept *byte*; there is clearly a thesaural relation between *bit* and *byte*. Thus a method using a combination of general-phrase finder and specific-phrase finder may provide the solution to the *is a* problem as it occurs in the above samples, and in examples similar in style to these although longer or more complex in their phrase structure.

Although it is conceivable that a general-phrase finder (akin to the 'duff'-term finder described earlier) might be constructed without an external KB or thesaurus, this is not conceivable for the specific-phrase case. Assuming that it is necessary to do both tasks to resolve the situation (so that the phrase may be labelled as specific, general, or undetermined) it seems inevitable that some form of KB/thesaurus will be required to solve the *is a* problem. Thus it can be concluded that KEP will not be able to resolve these cases properly unless external knowledge resources of some kind are provided.

6.2.3 Dealing with Episodes

In the first chapter the idea of episodic knowledge was introduced. Of concern here is episodic knowledge of the non-instance type, such as in the phrase *the wheel brace was used to remove the wheel nuts*. Here, *wheel brace* might be a recognised TT and so the elucidation *used to remove the wheel nuts* might be extracted. Clearly this is a historical episode. The knee-jerk solution is to avoid using *was* as a token in the pattern matcher. This might work in many cases, but might also result in a drop in recall,

due to phrases such as *was a tool used to remove wheels*. Therefore a better solution would be to engage a function specifically designed to detect historical episodes.

One approach might be to look for clues which indicate that a *specific* object within the discourse was treated (past tense) in some way. Consider the sentence *The solution was diluted with 100ml dilute sulphuric acid* , where *solution* has been marked as a TT. The task here is to avoid an extraction such as: **Concept:** *solution* **Definition:** *diluted with 100ml dilute sulphuric acid.* Here, not only does the participle *diluted* indicate a past event, but the use of the definite article in *The solution* indicates that the sentence is discussing a specific solution which was introduced earlier in the text, and which took part in some specific historical event. Thus the episode-detector function would use such clues to flag the sentence as being concerned with a specific instance of the *solution* concept taking part in a one-off event.

However, the task may not prove as simple as the above would suggest. In a sentence such as *In olden days, the anvil was used to make horse-shoes* the definite article in "the anvil" indicates "anvils in general" or "the tool called the anvil". Here, it would be nice to know that an anvil was something *used to make horse-shoes*. However, note that this is not a full definition of an anvil as tested by Skuce et al. It says something *about* anvils, but it does not answer the question "What is the meaning of anvil?". The latter would however be true for the sentence *In olden days, the anvil was a tool used to make horse-shoes.* In this case the elucidation *a tool used to make horse-shoes* does not start with a past participle. Thus the function might well function correctly in this case.

Clearly this is a complex area that begs investigation. However it would seem that there may well be sets of simple syntactic tests that could reduce the false positive extraction rate, and this subject will therefore be a priority future task.

6.2.4 Possible Effects of Text Type on Performance

The formal evaluations reported in the last chapter do not cover many texts of different types as discussed in Chapter 3. Despite this, some qualitative observations can be made concerning the effect of textual genre on extraction performance. The main judgements arising from the two large test texts 'B1G' and Chapters 1 – 4 of this thesis are as follows (ordered in the same way as the relevant sections in Chapter 3):

(1) **Expository vs. Historical text:** (Refer to section 3.2.2) Where texts report on historical episodes (e.g. on experiments with new configurations of GIS in 'B1G') there appears to be an increase in KEP's tendency to find bad "is a" extractions, both for the present and past tenses. The example for the concept "range" given on page 160 is such a case. The style of scientific writing which dictates

that incidents should be reported in the passive voice must shoulder a lot of the blame for this, because phrases of the form *X were Y* are common (e.g. *Experiments were carried out...*).

(2) **Informational vs. Presentational text:** (Refer to section 3.2.3) The exemplification relation seems particularly useful for presentational aspects of text. This may be related to its use to elucidate complex concepts, as discussed earlier. Very often the phrase "for instance" is used to introduce a presentational example, and in many cases one feels that the writer was not clear as to exactly what was being exemplified – it is almost as if the phrase "for instance" is being used to mean "let me explain further".

(3) **Generic vs. Specific text:** (Refer to section 3.2.4) Where specific objects are *people*, the extractions may still prove of interest, despite the non-generic nature of the concept - e.g. *Mrs. Thatcher*. However, most specific extractions seem to be associated with singleword technical terms (SWTTs) such as *system*, where it is some specific system that is being described. Since a word such as *system* might well be a TT in some text (e.g. one discussing the "systems approach") then clearly this is going to be a difficult problem to solve. (See also the discussion in the previous section regarding the problem of generic SWTTs.)

(4) **Fact-rich vs. Fact-poor text:** (Refer to section 3.2.5) The spread of attempted and successful extractions by sentence number indicates that even within a single text there are areas particularly dense in facts. Definitions in particular seem to be concentrated at the beginnings of text subsections. (BNC text 'B1G' is in fact a collation of smaller documents all on the same topic. Definitions appear mostly at the start of these.)

(5) **Declarative vs. Procedural text:** (Refer to section 3.2.6) Text 'B1G' did contain descriptions of procedures, often as contiguous multi-sentence sections. These sections gave rise to few extraction attempts, as expected, since the objects taking part in procedures do not tend to be defined within them.

A quantative evaluation of the effect of text type on performance must use many separate test texts, and has been placed on the "future work" list. However, it is already clear that useful techniques might arise from the above. For example, a text-skimmer looking for definitions would do well to look most closely at the first paragraph or two of any chapter. Also, any KE program intending to find and extract procedural information *must* have the capability to extract from contiguous groups of sentences – it is very rare to find single-sentence procedures. It would also need to be able to spot numbered lists, since these seem to be used very often.

6.2.5 Multi-Sentence Relation Instances

The extractor described in this thesis processes one sentence at a time. Leaving aside endophoric links between sentences, there are still cases where a good extraction could have been performed if the relation had not been spread across several sentences. The BNC text B1G contains instances of partition and exemplification spread over several sentences. Here, numbered lists were provided following an initial sentence of the general form "There are three parts to an X.". Since the knowledge to be extracted is stated explicitly in the text, it is frustrating that the opportunity to get it is missed. Therefore a priority area for future research will be the detection of such multi-sentence relation instances. This should detect numbered lists in both arabic and roman numerals, as well as lists labelled by letters. The roman numeral case may also be used to aid in the rejection of acronym candidates previously wrongly identified.

Nishida et al. (1986) also proposed doing multi-sentence KE. The intended approach was to look for small sections of text (up to three contiguous sentences in length) discussing a local topic. This topic was to be detected automatically, e.g. by term repetition, or clues such as the wording of headings. Facts about the topic expressed using inter-sentence relations were then to be extracted, using parsing, anaphora resolution and case-frame filling. The relations to be searched for were similar in flavour to RST relations but also included definitions and examples. Nishida et al. also intended to search for causation (see below). However, the paper discussed an ambitious system under construction which was not described in detail (e.g. the intended extraction methods were not detailed in any depth, and the methods of detecting the local topics of sections of text were barely described). Indeed, a literature search indicates that this system has not in fact been built and tested between 1986 and today, since no further references to it were found. (Although it is possible that it was built, it is highly unlikely that such an important achievement would have gone unreported in the IE/KE literature.)

Thus a multi-sentence approach to KE for conceptual/intersentence relations does not yet appear to have been realised. In any case, the Nishida et al. approach would appear to have been a deep one, as evidenced by the need for full parsing. It is also not clear as to whether the system would have been domain specific or NDS. Thus the extension of KEP's pattern-matching approach to multi-sentence units of text would indeed appear to be a novel method worth pursuing.

6.2.6 Following Simple Anaphoric Links

At present, simple anaphoric links are identified but not followed. This topic has been considered in some depth in Bowden, Halstead and Rose (1996d), where an algorithm designed to detect simple links from *this*-anaphora (and other simple anaphora) to terms in the preceding sentence is given. However, as the evaluation in the previous chapter has shown, it is unusual for there to be a simple easily-resolvable case of *this*-anaphora. Many of the instances where a concept was extracted as "this" require semantic processing. Although it is the case that in more cases than not the resolution does indeed lie within the

preceding sentence, it is rare that it can be simply snipped out. This, combined with the relative scarcity of *this*-concepts in the extractions performed by KEP (with the possible exception of exemplifications), has lead to the conclusion that solving simple anaphors should not be accorded high priority. It is indeed a truly difficult task, which even if it were solved completely, would not result in significant improvements in KEP's recall rate.

6.2.7 Subdivision of Relation Types

The four relation types used by KEP are fairly broad-brush categorisations. The partition relation may be subdivided into several part-whole types (see discussion starting on page 73) and the definition relation is probably not "pure" since definitions may be couched in terms of the concept's parts, uses, forms, history etc. Examples have also been classified in various ways (as discussed from page 74 onwards).

The difficulties with partition in particular lead to the suggestion that this relation should be split into its distinct sub-types, each of which would be handled separately. This would naturally give rise to shared-pattern problems similar to those given in Table 20. Once again the problem of deciding which relation subtype is present would arise, and again it might be that external knowledge would be required. For example, for a Place/Area partition, such as in *The Everglades are part of Florida*, an external KB would provide the knowledge that *Florida is a type of Place* and thereby allow the identification of the correct relation subtype.

Examples have been classified in various ways but the *negative example* is nearly always one of the subtypes suggested. KEP presently attempts to filter these out, but one could argue that a negative example of a concept is a useful piece of knowledge to put in a glossary. Thus one might look separately for negative examples, using trigger phrases such as *is a poor example* or *is not a*. Clearly, *is not a* must raise problems similar to those arising from *is a*.

The hypernym relation appears to be unusually pure in that there do not appear to be subtypes. By its very nature, the hypernym/hyponym relation describes objects which are in some way versions of other objects. The actual relationship will depend upon the objects themselves. One object is either a type of other object, or it is not. (The exact details of *how* one object is seen as a version of the other are not always explained in the elucidatory text.) Thus the hypernym relation is actually a very broad one, and yet, despite this, a fundamental one. This fundamental nature of the hyponym relation has been remarked upon by many researchers, such as Hearst (1992), who noticed that it is probably unusually easy to extract from running text. Thus it is unlikely that this particular relation will need to subdivided.

6.2.8 Allowing Terms in Pattern Matching

The evaluation in the previous chapter has shown that there are circumstances in which the pattern-matching technique employed by KEP cannot extract a concept/elucidation pair *in principle*. For example, consider the following sentence section, taken from Hearst (1992):

> "...most European countries, especially France, England and Spain."

Here, although *European country* might be a TT already found by the term acquisition stage, the pattern matcher would fail to make an extraction because of the lack of punctuation before the concept. Although one could conceivably use the word *most* as a token, this would not be a sensible idea given the multitude of potential modifiers etc that could be placed here. What is ideally required is a pattern matcher which allows terms to take part in the pattern matching stage, rather than use them after this stage for validation of fragments. For the above example, the pattern to match the sentence might then look like this:

X T , e T , T + T .

Here, the symbol **T** represents a technical term (in either its plural or singular form) and **e** represents *especially*. This template would allow the desired extraction, although using TTs in the pattern-matcher itself would require complex changes to the algorithm currently used. This is certainly a feasible solution to the problem but it does depart from the initial philosophy of having a very simple pattern-match against any sentence in a non domain specific form (for not only are TTs *domain* specific, but they are also in a sense *text* specific as currently found). This approach would not however make KEP as a whole any less non domain specific as long as the TT extraction method itself remained NDS.

6.2.9 Parsing of Elucidation Fragments

KEP does not presently parse the elucidation fragment in a concept/elucidation extraction. The mechanisms described in section 4.6.15 attempt to ensure that this part of an extraction is good, but as was mentioned at the start of this thesis, one of the ideas behind the pattern-matching approach adopted was that it might be possible to parse fragments of sentences when necessary.

Taking the example introduced above, then a fragment such as *<some text> most European countries* might be parsed using a TT-aware parser. If this parser were able to detect quantifiers such as *most*, *all*, *some* etc then the TT *European country* could be extracted from the fragment as the concept being elucidated before the token *most*. Thus the need for awareness of all quantifiers, i.e. the need for a MRD containing part of speech information, would be pushed back into the fragment processing stage. This is a valid future approach but does take KEP away from the aim of not using "deep" NLP tools. Having

said this, it is probable that a sentence-fragment parser is much more simple a device than a whole-sentence parser, and this viewpoint has been convincingly argued for by McDonald (1992) amongst others. Thus this suggestion is worthy of future investigation.

It is likely that some conceptual relations will lend themselves more readily to some form of parsing than others. Definitions, for instance, may be particularly amenable to some form of parsing, or at the very least *sectioning* (possibly using a pattern-matching approach). Several researchers have described the structures typically used for definition (e.g. Selinker, Trimble and Trimble (1976), Darian (1981), Swales (1981), Flowerdew (1991, 1992a, 1992b), Pearson (1996)). A common semantic pattern identified by these researchers (in the author's terminology) is as follows:

Concept = *hypernym* + distinguishing characteristics

This semantic pattern (see also Section 5.3.6.3) may be realised as many syntactic patterns. Some examples, using the same bold/italic/underlining as the above so as to identify the semantic sections, are:

> **The tiger** is a *large cat* which lives in India.
> **Tanks** are *armoured military vehicles* designed to rapidly breach enemy defences.
> **Alpha particles** are *nuclear fragments* comprising two protons plus two neutrons.
> A **conservative field** is *one* in which there is no change in energy round any closed loop.

Note that the plus sign in the definition semantic template maps to a variety of semantic relationships: geographical location ("which lives"), purpose ("designed to"), partition or composition ("comprising"), characteristics ("in which"), and any others as required to make a distinction between the particular concept being defined and other members of the hypernym class. The hypernym itself is signalled by a class designation ("large cat" etc) which may be replaced by an anaphoric or null element ("one").

These forms of definition implicitly follow the object oriented (OO) paradigm, in which a sub-class (concept's class) gains (*inherits*, in OO terminology) most of its characteristics from a parent class (hypernym) except that the set of characteristics must be slightly different in order to distinguish the daughter class (see e.g. Parsons (1997) for a readable introduction to the OO philosophy). This fact has been recognised by researchers attempting to build hierarchical semantic nets from collections of definitions of the above form. For example, where a dictionary follows this basic pattern for most of its definitions, it may be possible to parse these into a form which can be integrated with such a structure. Litkowski and others are attempting to do just this, using the 1913 edition of Webster's 2nd International Dictionary[14]. In the case of KEP, the idea is not so much to cut the definition (i.e. the elucidation part) into its semantic pieces, but rather to recognise the whole text string as a definition for inclusion in the

[14] The Dictionary Parsing Project (DPP) is described at http://www.isi.edu/natural-language/dpp/

glossary, i.e. for validating the elucidation part. As was discussed in Section 4.6.11, no validation is currently performed on the cut-out elucidation parts, so this identification of the definition substructure certainly allows scope for doing this in the future.

6.2.10 Re-wording of Elucidations

The entries in the third column of the glossary produced by KEP comprise snippets of text cut from the source sentence. In many cases these are acceptable, but in other cases a minor re-wording of the explanation word improve the readability. Two surface forms may be equivalent but one much better for a glossary than another. For example, for the concept *iteration* one might prefer *the use of loops* over *using loops* as the explanation. However, the latter might be the form which was present in the original sentence. The question thus arises as to whether it is possible (a) to identify cases where a minor change to surface form would be desirable, and (b) to make this change.

Although it is possible to envisage shallow transformation methods for commonly occurring phrases (such as the alteration of the word *using* to the phrase *the use of* if it starts an explanation) it is clear that a thorough treatment of this requirement will not be possible using this approach. In some cases a complex syntactical transformation would be required, based upon the meaning of the sentence as a whole. Shallow rule-based word rearrangements and word-ending changes will not be sufficient, since the semantics of the sentence will determine the transformation. In addition, syntactical information may be required. For example, not all verbs may be transformed from an active to a passive form (e.g. stative verbs).

It remains to be seen whether a purely mechanical transformation process can be applied which does more good than harm to the glossary third column entries. However it is quite clear that this problem can never be completely solved using such an approach. The use of syntactical knowledge such as verb subcategorisation data, possibly collected automatically over many runs of the program on large amounts of text (see e.g. Manning (1993)) may help, but ultimately an approach which builds a representation of meaning must be used.

6.2.11 Additional Conceptual Relations

Future enhancements of KEP should consider adding new conceptual relation types, such as **instance**, **causation**, **nomination** (sometimes called **appellation**) and the **material** relation (see e.g. the list in section 3.3 which starts on page 72). This is a relatively straightforward process involving the addition of code and external files, the population of the latter being achieved as described earlier. However, the "is a" problem and other problems arising due to shared syntaxes are likely to worsen. It is already the case that the same extraction is sometimes made by two different relation types. Code has already been written to prevent duplicated 3rd-column glossary entries, but this does not solve the original problem (it

merely hides it). Therefore the categorisation A,B,C,D task described above will assume increasing importance as more and more relation types are added.

A conceptual relation which might prove of particular use, and for which much data is already available, is that of **causation**. This relation is used to describe *cause and effect* descriptions, as well as *reason and result*. These sub-categories are subtly different but sufficiently close as to allow them to be treated together. (The latter type is closer to *justification* or *purpose* - see Vander Linden and Martin (1995).)

Furthermore, experimental corpus studies have already been performed in order to obtain lists of indicator phrases for this relation. Flowerdew (1996) uses a categorisation scheme of three divisions: reason-result, means-result and grounds-conclusion (personal communication of paper being prepared). Lists of explicit linguistic devices used to express each of these divisions are provided, arranged by part of speech (nouns, conjunctions, complex prepositions, prepositions, verbs, adjective phrases and adverbs). Two corpora were used: a collection of texts from the MicroConcord Academic Corpus Collection entitled *Global Warming: The Greenpeace Report*, and a group of 80 student assignments discussing environmental topics. (The main motivation for this work lay within computer aided language learning (CALL)).

Xuelan and Kennedy (1992) have also studied the causation relation, making a study of devices used to signal causation explicitly in the LOB corpus. Results from this study are presented as two tables: devices for cause/reason, and devices for result/effect. Xuelan and Kennedy distinguish between implicit and explicit causation markers. Explicit causatives include phrases such as **cause**, as in *sulphite preservations can **cause** rashes and abdominal pain.* Implicit causatives may be hidden inside certain verbs, such as **destroy** in *the earthquake **destroyed** the building*. There is certainly causation hidden in this sentence, because it is semantically equivalent to *the earthquake **caused** the destruction of the building*. KEP will only deal with explicit causation since it cannot perform such semantic transformations, or even detect the need to attempt them. This would require a lexicon of verbs (and other parts of speech) capable of expressing causation implicitly, together with semantic processing probably requiring world knowledge. Of the 130 explicit causative devices listed by Xuelan and Kennedy, about 40% of them are labelled as *unambiguous*, i.e. they always indicate causation whenever they are seen. Thus they are Class A markers in the categorisation scheme suggested above. The full list is given in Figure 21. Note the large number of phrases involving the character string 'consequen', an obvious positive trigger for this relation. (The phrases in the figure are arranged in decreasing order of occurrence counts, when read left to right and top to bottom.)

because	why	therefore	effect	reason	result
because of	for (that) reason		as a result of	consequence	
consequently	result in	bring about	result from	thanks to	
in the light of	on account of	outcome	give rise to		
on the ground(s) that	thereby	by reason of	what with	accordingly	
by virtue of	in consequence		bring on	engender	
on the ground(s) of	on the strength of		with the result that		
inasmuch as	in consequence of		occasion	as a consequence of	
consequent on/upon	corollary		underlie	as a consequence	
for reasons of	in consideration of		on that account	on that score	
seeing that	upshot	with the consequence that		by consequence	
consequential to	cos	from reasons	mainspring		
the whys and wherefores	by courtesy of		give occasion to	seeing as	

Figure 21. Explicit unambiguous causation markers, after Xuelan and Kennedy

6.2.12 Use of MRD for Third Column Entries

It has been almost an article of faith in this thesis that the KEP program should not need to access external knowledge resources. The reasons for this include the desire to maintain KEP's NDS credentials and the desire for speed in certain sub-functions (such as plural noun singularisation). However, the sparse nature of the third column in the glossary output has been remarked upon (see section 5.3.6.11) and so one should at least consider ways of improving this situation, even if external knowledge resources are required to do so.

One solution might be to use a MRD to provide 3rd-column definitions. This could be done just for terms having no 3rd-column entry, or for all terms including those already having elucidations. Two questions arise. Firstly, will many of the 2nd-column terms actually have dictionary entries? Secondly, will these entries prove to be correct for the domain of the text? (If there is more than one entry for the term in the MRD, what should be done?)

In order to test the first question, the page of the glossary output given as Figure 20 on page 171 has been processed manually using Chambers English Dictionary (Schwarz et al. (1988)). Since the figure contains a real page of glossary output, which includes singleword terms, 2-word terms and 3-word terms, including some bad terms, it is a good test of the idea of using an MRD. This test used only one dictionary, but any future realisation of the method would need to examine several MRDs to determine if one were better than most, or if more than one MRD might be needed.

Most dictionaries use single orthographic words as head terms. Thus for 2- and 3-word terms, in a realised version of this test, extra processing would be required to detect the term. For example, if the term to be looked up was *chain reaction*, in the printed version of Chambers this appears under the head

term **chain**, as **chain reaction**. Therefore a search operation would be needed within the head term's entry, and following this other unwanted data removed, such as pronunciation and etymological information. (Unlike in some dictionaries, Chambers prints the whole term rather than replace the head word with a symbol such as a tilde, i.e. as in ~ **reaction**. However, stress markings would still need to be removed.) Since multi-word entries do therefore exist in most dictionaries, the worry that it might not be possible to find 2- and 3-word terms at all is dismissed. This is not to say, however, that all 2nd-column entries in KEP's glossary output will be present in the dictionary.

Of the 2nd-column terms in Figure 20 the following have no entries in Chambers: data source, data storage, data structure, data uncertainty, data volume, *database for hazard*, database management system, database management, database view, decision support system, decision support, deep repository, defence system, derived polygon. (The italicised terms in this list are bad TTs, wrongly extracted by the term acquisition stage.) Thus 14 of the 17 glossary entries in Figure 20 would not be assisted using a machine-readable version of Chambers. Of the 2nd-column terms in Figure 20 the following do have entries in Chambers: data, database, design. Thus only 3 of the 17 entries might be assisted by this approach.

These results indicate that a word dictionary is not the same thing as a term dictionary. It is conceivable that this is not the case for very large dictionaries, but clearly terms such as *derived polygon* and *deep repository* are tightly domain specific. This returns us full circle to the argument used to justify the NDS term acquisition method employed by KEP: lists of technical terms are not used to create the 2nd-column entries because they would be (a) numerous (one lexicon per domain, whenever a new domain was recognised), (b) huge, and (c) require constant updating. For the same reasons, term dictionaries are not a practical proposition (for they are merely term lists with added definitions).

The second question raised above concerned the correctness of any definition found from a dictionary. In the case above, all three definitions were good, although in the case of the term *design* the entry is long and general. Similarly, the entry for *data* is good, but does not mention computers, a major topic of the source text. The entry for *database* is just what one would require: *a large body of information stored in a computer, which can process it and from which particular pieces of information can be retrieved when required.* It is doubtful whether *design* is actually a good term to be placed in the glossary for the source text. Thus the MRD usage has only come up with one really good extra definition for this page of the glossary output. Note also that the problem of multiple word senses has not arisen in this small test; this is likely to be more of a problem for single-word terms, where some strategy would be required to find the most relevant definition.

The test results above indicate that the use of MRDs to assist in 3rd-column filling is not likely to be a particularly useful strategy. *NDS* MRDs will not assist much, and *DS* MRDs are impractical in a NDS

system such as KEP. Thus MRDs at first sight do not appear to be the solution to the sparse elucidation column[15]. But perhaps this is a little unfair, since a human reader would know that the source text was about GIS (from the text's title or its introductory paragraphs) and would naturally reach for a dictionary of computers/geography/science, all of which are available. Therefore before dismissing MRD use completely, it is worth exploring this issue a little further.

The use of e.g. a dictionary of science does, of course, represent the use of a *domain specific* dictionary, or at least a dictionary which is *more* domain specific than a word dictionary such as Chambers, for there are degrees of NDS-ness. (A dictionary of science is DS to science, but a dictionary of physics is more DS, and a dictionary of particle physics is still more DS.) However, if the numbers of such dictionaries were low, and if they were available in MRD form (or indeed as *term banks*[16]), then their use might produce better results than the small manual test performed above. With the addition of a topic detector, the KE system might select the relevant MRD. This might even be done by comparing term lists (terms from the text vs. terms from each MRD in turn), although there would not be much point in doing this unless the KE system were required to state the topic of the text, since terms would have to be looked up in one of the dictionaries in any case (i.e. a simple search of all dictionaries would be just as fast if not considerably faster). It is almost certainly the case that *database management system*, *data volume*, and *data structure* occur in most dictionaries of computers. Any dictionary of geology worth its salt will list *deep repository*, a term which probably also occurs in any good dictionary of nuclear engineering. Provided the set of dictionaries can be kept small (so that long search times do not occur), many more definitions might be found. The problem of neologisms remains, but this might not be critical if one is able to rely upon regularly-updated dictionaries from the publishers (e.g. by supplements) so that recently coined terms (not coined within the *current* document) are defined. This approach to the 3rd-column sparseness problem is certainly one which should be investigated in the future.

6.3 Future Applications

In this section some future applications of an enhanced KEP are considered. Although they are a diverse set, they are all based upon KEP's current abilities. However, they would all require a substantial amount of design and coding effort, and for this reason have not been attempted within the timescales of the current research.

---

[15] Béjoint (1988) has considered the intrusion of technical terms into general dictionaries from the point of the lexicographer, and discusses how such dictionaries do not begin to contain such terms until they have reached a certain word-size; this upholds these results.

[16] The term *term bank* is a relatively new one used by terminographers/terminologists to mean computerised, structured lists of DS terms.

6.3.1 Text Summarisation

The KEP term summaries output has already been described in some depth in this thesis (see e.g. section 4.6.6). It is possible that this may be used as the basis for a directed text summary. In a directed summary, the user asks the system to summarise what the text has to say about a particular topic (term). Unlike the term summaries output, a text summary should be a readable section of text without evident gaps - in other words it must flow. This is not at present a feature of the term summaries output, except by accident, and so extra processing would be required to add connecting sentences or alter the ends of sentences adjoining "gaps" in the narrative. Thus some natural language generation (NLG) code would be required.

Term summaries may also provide another method of filling the third column of the glossary. This method would be to apply text summarisation techniques to the term summary entries, so as to provide a general statement about the concept. This method would not provide individual relation extractions (as attempted by KEP currently) and so would not be useful for automatic semantic net building with its requirement to use individual link types (see section 6.3.5). However, it might prove successful for automatic dictionary construction, descriptions of concepts in systems such as HyperTutor, and a smoothly readable column-3 glossary entry.

Whole text summarisation is a different matter. This involves the identification of the important sentences/paragraphs in the text, where by "important" is meant those that are closely related to the essential topic(s) of the text. The word processor used to create this thesis (Microsoft Word 7.0) incorporates a text summariser which picks out the most important sentences based upon counts of topic words made from the entire text[17]. However, when applied to this thesis the results were disappointing; a 10% summary (about 35 pages long) missed out many important sections, and even presented *parts of* tables and figures, clearly not a useful approach.

It is possible that KEP could identify the most important sentences and paragraphs, using term counts and local term densities. This is an interesting idea that could be implemented and tested relatively simply. All sentences in the output would be marked with an importance metric derived from the number of different TTs within and near it, and from the "quality" of those terms (a TT used many times in the text as a whole being regarded as of higher quality than one only use a few times). In addition, relation instance densities could be used. An importance-bargraph would be drawn for the entire text and the user's chosen summary length (half the size of the original text, down to a single paragraph, three sentences etc) used to set an importance threshold. Sentences having importance scores above this

[17] Brief details are given in the Help system of the WP; for commercial reasons, full algorithmic details are not provided by Microsoft.

threshold would then be included in the summary. Again, NLG methods might be required to smooth-over any gaps.

Commercial text summarisers do already exist. The Microsoft Word summariser has already been mentioned, and BT's program *NetSumm* provides text summarisation on a pay-per-text basis over the Internet. However, as is the case with many commercial products, the methods upon which NetSumm has been based have not been published, although outlines of its capabilities have been given. There is, however, a fully described method for creating text "abridgements". This is the method of Hoey (1991). The method relies upon identifying *central* and *marginal* sentences in a text (all non-central sentences being marginal). Hoey suggests that the central sentences written in order form an abridgement (essentially a summary made by picking the most important sentences). Central sentences are identified as those having more *bonds* to other sentences than the average. A sentence is regarded as having a bond to another sentence if it has three or more cohesive links to that other sentence. Since many cohesive link types are largely lexical, there is the possibility of automating the process. Although not currently detected as such by KEP, it may be possible to build automatic cohesive link detectors (see e.g. Jobbins and Evett (1995)) and so automate the abridgement process. Hoey's method could be used in conjunction with data extracted by KEP to reinforce the identification of central sentences, or to create a super-category of "very central" sentences to allow more than one level of text shortening. For example, Hoey-central sentences which also contained a definition of a concept (as detected by KEP) could be marked as "very central". Thus there are exciting prospects for hybrid text summarisers.

6.3.2 Automatic Index Creation

Indexes of the sort found at the back of books are simpler entities than the glossary. They require terms and page numbers. The terms must however be permuted (e.g. *chain reaction* and *reaction, chain*) so that the user may find the term easily. Since KEP already finds terms, and since it knows which sentences they occurred in, an automatic index creator could be built with a small amount of design and coding effort.

The input texts processed by KEP do not necessarily contain page numbers. Where page numbers are absent, KEP could generate them based upon a standard page size (lines, characters-per-line etc). With page numbers present (or generated) a mechanism would be required to link sentence numbers to page numbers (so that TT page numbers could be found). This might however give rise to the odd incorrect page number, where the term existed in a part of the sentence which crossed a page boundary. Thus it would be preferable to link terms directly to page number.

Term permutation is not a difficult task. The patterns of permuted terms are simple and few. The basic term patterns of Justeson and Katz (1995) given in Table 14 would each be allowed a set of

permutations, so that e.g. **N1 P N2** would become **N2 , P N1** (e.g. *use of loops* becomes *loops, use of*). The permuted terms would be added to the term list and the index creation function would build the index in a manner similar to that of the glossary maker. The index would simply be another KEP output file. Thus it would be a relatively simple matter to build a completely automatic index generator into KEP. This is a feature which even the latest WPs do not appear to have[18]; the indexer in Microsoft Word version 7.0 requires the user to manually mark all index terms for inclusion. Thus it would represent a useful new word processor feature.

6.3.3 Student Assignment Marking

It was mentioned in the introductory chapter that an original motivation for the work described in this thesis was the desire to automate certain aspects of student assignment and examination marking. Although this has not in practice been a major goal, it is worth considering how KEP as it currently exists might be useful in this direction.

Although there is a lot more to an essay than a collection of technical terms, it is possible that there is a correlation between the set of TTs used in an essay and the quality of that essay. This is speculation, and the hypothesis would of course require extensive testing. For example, on an essay on telecommunications satellites one might expect to see the terms *GEO*, *LEO*, *geosynchronous Earth orbit*, *apogee*, *perigee*, *Arthur C. Clarke*, *footprint*, *launch site* etc. Some of these terms might be deemed essential by the human marker, i.e. thought to be so important that they must be present in any good essay. Others might be regarded as less than essential but desirable. Thus it is possible that a "term profile" could assist the human marker in identifying the scope covered by an essay, if not its quality.

As an initial test of this hypothesis, and also to test its efficacy as a potential new IR method, a post-processor program (*ir.c*) was developed to compare two sets of terms (derived from KEP glossary output files) using a variety of similarity metrics designed to calculate the degree of Venn-diagram style overlap. Thirty-one BNC texts were selected, three of which were ostensibly on the same topic (based upon their BNC subject classification). One of these three texts was taken as the reference text (i.e. as the "ideal essay" or the "paper of interest") and the other thirty were taken as search-space texts. KEP was run for all 31 texts and similarity metrics calculated between the reference text and the other texts one by one. The search-space texts were then ranked by similarity. It was found that the two search-space texts most similar to the reference text were the other two same-topic texts. Furthermore, about 14 of the other texts were marked as "completely unrelated" to the reference text (as indeed was the case). These experiments are fully detailed in a paper available from the author (Bowden, in preparation). They

[18] This is surprising given that as long ago as 1983 Dillon and Gray (1983) put forward a fully automatic indexer, and in 1988 Salton (1988) suggested syntactical means of identifying index terms.

demonstrate that the method does appear to be able to detect "closeness" of topic/contents based upon extracted technical terms.

This method is similar in approach to Allott's APN method (Allott, Fazackerley and Halstead (1994)) which attempts to detect concepts present in a single sentence based upon nodes (phrases) present or absent. Allott's activation passing network allows the presence of nodes at one level to "fire" a higher-level node. The examiner sets up an APN "answer" to a question, and the students' single-sentence answers are judged against this. By marking certain technical terms as essential, the flavour of such an APN could be simulated by KEP, albeit at a whole-text level rather than at the level of a single sentence.

Clearly, any method which marked an essay based solely upon term or acronym lists would be open to abuse. For example, an essay answer which was just a simple list of domain TTs would get high marks. Thus it would be advisable to combine this approach with other factors, such as relation instances extracted by KEP. Where students had been taught a precisely-worded definition, KEP might detect this in the essay. Similar comments apply to examples of concepts. In addition, the numbers of definitions, hypernyms, examples etc present would be of interest. One might also combine these KEP-methods with commercially available programs such as grammar and spelling checkers.

The above represents a whole area of research in its own right and this is not the best place to explore it. There are many practical problems which would have to be overcome before evaluation could start (for example, students would have to submit machine-readable work) but it is an interesting area for future exploration.

6.3.4 Engineering Project Estimation

Many engineering disciplines require design documentation at the early stages which describe the elements of the finished product. Furthermore, these documents are used by human experts in an attempt to cost the project, in terms of time, man-years, and pounds sterling. This is a skilled task which requires both a systematic approach and knowledge of past projects. It is not surprising, therefore, that attempts have been made to automate this process from the requirements specifications. One of the most obvious approaches is to look for "abstractions" i.e. terms in the documents which represent objects to be built and/or used. These abstractions are similar to technical terms in that they are usually represented by simple noun-adjective combinations. However, they may also exist at higher levels semantically wherein synonyms are used to identify the same abstraction. For example, "buy materials" and "purchase materials" are essentially the same abstraction. The AbstFinder program of Goldin and Berry (1994) is one example of a program which attempts to find abstractions in engineering specifications (see page 137 for previous discussion).

Some simple experiments have been performed using KEP's TT stage to see if this can provide lists of abstractions from technical specifications. Early results are promising (see Bowden, Hargreaves and Langensiepen (in preparation) for full details). These initial tests show that the TT lists give good coverage but omit many singleword TTs (SWTTs). To remedy this, an experimental SWTT extractor was added to KEP's TT extractor. This function has not been described in the body of the thesis since it is a very recent addition which has not yet been fully developed and evaluated. It works in a similar manner to the existing algorithm except that the patterns searched for are much simpler – i.e. there is only one pattern, the trivial pattern **N** for noun. Plural to singular conversion is still required, however, since individual nouns do occur in either number.

However, a new aspect for this SWTT code is that the count threshold may be varied. The Justeson and Katz threshold of count = 2 may be used, or some other threshold such as count = 3, or count = (mean count) + 1. Experiments are underway to determine which threshold produces the best results in terms of the balance between recall and precision. For example, a count threshold of (mean count)+1, where *mean count* is the average number of times a given noun occurs in the text, gives a list of SWTTs which are mostly good (i.e. precision is high). Unfortunately, it also appears to miss some other good SWTTs which occur less than the threshold number of times in the text (i.e. recall is low). This threshold does however have the advantage of adjusting itself to the length of the text (actually to the number of nouns used in the text, which approximates to the latter). Other thresholds give different balances between precision and recall.

Clearly, this method will not combine TTs such as *buy material* and *purchase material* into one single abstraction. For this a dictionary of synonyms would be required. In a DS application this might be provided by the user, as was the case for AbstFinder. The mechanism would seem to be fairly straightforward given such a synonym list. However, note also that *buy* and *purchase* are in fact verbs, and so not found by KEP's TT stage at present. Therefore preliminary studies need to be performed to see what percentage of the combined abstractions come from actions rather than objects, so that such patterns may be searched for as necessary. Whatever, this does seem an interesting potential application for KEP.

6.3.5 Building a Permanent KB

At present, the conceptual extractions made by KEP are not made available to future runs of the program on new texts. This was a deliberate decision made in keeping with the philosophy of being domain independent, expounded throughout this thesis. However, there is no reason why KEP should not maintain a permanent semantic-net KB held on disk file. This KB could be used in one of two ways. Firstly, it could be used merely as an output store. New facts would be merged into the semantic net in such a way that repetition would enhance the certainty of a particular fact, or expand knowledge of a

certain class of objects, or introduce an entirely new class. The semantic net need not be a single interconnected entity; multiple nets might be constructed, by domain.

Semantic nets have already been mentioned in several places in this thesis. They are indeed useful as KB structures, but their exact forms vary from system to system. For example, in the 'wit' system, Reimer (1989) used a variation having only one link type (*is a*, used both for both hyponymy and instance) with complex nodes having slots to effectively hold other conceptual information, these slots being created dynamically by a KE system (see section 2.4.2.3). In WordNet (Miller et al. (1990)), a large-scale semantic net project, *is a* links (hyponyms) and *has part* links (meronyms) are used to link *synsets*, i.e. concepts represented as sets of synonyms. Some semantic nets use only one type of link, e.g. within *composition graphs*, which use only *has part* links (see e.g. Magnan and Oussalah (1995)). In KEP, although no semantic net is created, the conceptual output is better thought of in terms of a semantic net having simple concept nodes (i.e. nodes have only a label, the concept name) but with a multitude of link types (*is a type of*, *has example*, *has definition*, *has part*, *causes* etc).

The updating of an existing semantic net is not a simple task, but this has not prevented several researchers from considering the task. For example, Virkar and Roach (1986) describe a text processor designed to extract knowledge for the DIE system (an expert system concerned with drug interactions). The system processes pharmacology abstracts and uses a pattern-matching IE method based on DS "text grammars". The difficulties of updating an existing KB are mentioned although not fully detailed. Also considering the task is Szpakowicz (1990) who describes a potential system for extracting mini semantic nets from text and integrating them with an existing semantic net KB. The paper describes only preliminary work on the program, but the broad approach is to generate pieces of semantic net (having *Object* or *Activity* nodes, joined by *is_a* etc links) from the manual of the 'Quiz' software product. These mini nets are then integrated into the existing KB. Although short on detail, Szpakowicz does propose trying to generate an extraction from each separate sentence (as with KEP), although he also suggests paragraph-sized extractions. Garigliano, Morgan and Smith (1993) discuss updating of semantic nets in the LOLITA system (a large multi-function project which includes NLG, MT, and text summarisation). Finally, there is Reimer's 'wit' system for extracting and storing data about computer printers (Reimer (1989)), which was described earlier. Thus it is clear that many researchers are actively considering this interesting area.

The KEN file (Knowledge Extraction Network) created by KEP as an interface to the HyperTutor system of Edwards, Powell and Palmer-Brown (1995) is a step in the direction of a write-only permanent semantic net KB. In this case, the maintenance of the semantic net(s) lies within the remit of HyperTutor rather than of KEP. The HyperTutor/HypeLab system uses semantic nets as KBs of tutoring knowledge, termed *curriculum graphs* within this product. The form of these curriculum graphs is close to that envisaged for KEP's potential semantic net storage. However, there are differences. For example, in

HypeLab/HyperTutor the partition and exemplification links map well, but the definition of a concept is held as a description within the concept node.

Link Type Group	Forward Link Type	Reverse Link Type
Being	has a type has an instance has an example	is a type of is an instance of is an example of
Including	has a part has a procedure	is a part of is a procedure of
Doing	performs carries out	is performed by is carried out by
Using	requires	is required by
Causing	produces	is produced by
Showing	has a picture has a diagram has a video clip has a simulation	is a picture of is a diagram of is a video clip of is a simulation of
Similarity	has a reference has an association	is referred to by is associated with
Quantifying	has a size has number	is size of is number of
Qualifying	has a characteristic	is a characteristic of

Table 21. Link Types in HypeLab/HyperTutor (from Bowden and Edwards (1996))

Furthermore, the set of link types used within HypeLab/HyperTutor is larger than that currently available to KEP, and each is to be used in a specific way by the curriculum graph creator during the authoring process. Table 21 shows the HypeLab/HyperTutor link types (from Bowden and Edwards (1996)).

Note that there is a corresponding reverse link type for each forward link. This is so that a learner may navigate around the curriculum graph using semi-NL phrases as described by Long, Powell and Palmer-Brown (1995). The specific purposes of some of the link types given in Table 21 can result in situations where KEP would not produce the same curriculum graph as a human author. For example, consider again the test text of Figure 8 and the short output file derived from it, Figure 9 on page 84. Assuming that KEP was able to detect the **Sorting** concept (by one of the single-word TT extraction methods discussed in this thesis) then this concept would appear in the output files. Figure 22 represents the semantic net drawn from the short output file. Here, the **Sorting** node is not attached to the rest of the network (it should be connected to **Sort Routine** via a *performs* link) and **Criterion** is attached to **Sort Routine** via a *has_a_part* link (but a human HypeLab author would be expected to use the *has_a_quality* link). Thus a certain amount of human intervention will be required in any combined KEP/HypeLab system in order to ensure that the curriculum graphs suggested by KEP adhere to the HypeLab/HyperTutor authoring conventions.

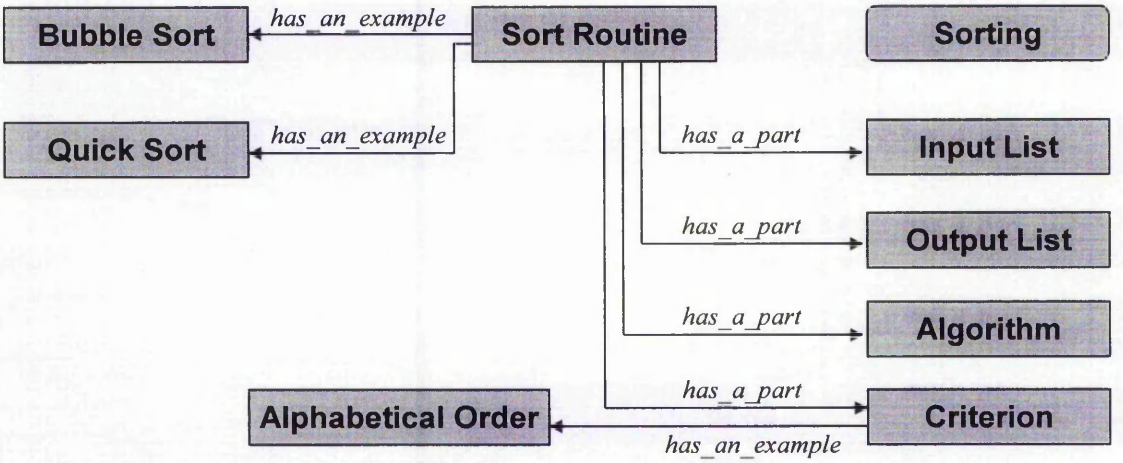

Figure 22. Example of a KEP-generated Curriculum Graph

The KEP to HypeLab/HyperTutor interface is an ASCII file written to disk by KEP at the end of processing. An example of this has already been given - see Figure 10. Note that *concepts* are distinguished from *processes*, and that this is illustrated Figure 22 by having the **Sorting** process drawn as a rounded box. Whereas KEP does not distinguish between types of concept extracted, in HypeLab concepts are subdivided into **concepts**, **processes**, **media**, **qualities** and **quantities**. Thus for example, *making the tea* would be regarded as a **process**, whereas *teapot* would be a **concept**. An example involving a **quality** has already been discussed. This leads to a mismatch between KEP's broad-stroke output and the node requirements of HypeLab/HyperTutor. In order to minimize the amount of human intervention required after automatic curriculum graph creation, this mismatch needs to be reduced as far as possible. The suggested resolution is as follows. KEP concepts will be categorised by an expert HypeLab author, into the categories as described above. These will then be examined to see if any obvious tag patterns arise which might signify e.g. a process as opposed to a concept. For example, the presence of present participles or gerunds might indicate processes (e.g. ***making*** *the tea*). However, it may be that this simple approach is not capable of distinguishing between certain pairs of concept types. If this proves to be the case, future studies will need to apply semantic and possibly pragmatic knowledge to resolve this problem. In the short term, manual intervention would still be required.

The second way in which a pre-existing KB may be used by a future KEP is to aid in the current extraction run. This need not be as simple as looking up an unknown acronym, say, to see if it is already known. A more subtle approach may be needed (after all, the acronym STD means quite different things in an article on telecommunications and one on human reproduction). Thus the KB might first be used to identify the general subject area (e.g. by counting extraction "hits" over various areas within the net). This knowledge of the domain could then be used to disambiguate conflicting meanings.

Given a mechanism for correcting (or ignoring) errors within the semantic net KB, a closed-loop learning system would be created. This mechanism might be "fuzzy" in the sense that it is statistical rather than boolean. The more text processed by KEP, the greater the knowledge stored, and the better KEP would become at obtaining new knowledge. This positive feedback would allow increasingly good performance over time, although it would probably become asymptotic to a performance ceiling due to KEP's inherent limitations regarding semantics etc.

In such a system, very many KEP processes could all contribute to a single KB. One might envisage web-crawling versions of KEP, all reporting back to a single huge KB. This is not as fantastical as it might first appear; the mechanisms for traversing the world wide web (WWW) are already well established and in use by the large commercial search engines (AltaVista, Yahoo, Infoseek etc). These companies already extract indexing terms automatically from WWW texts for storage on large central servers. The extraction of acronyms, technical terms and indeed facts could be regarded merely as a logical expansion of this process.

6.4 Concluding Discussions

In the introductory paragraph to this chapter a number of questions were posed relating to the maximum performance which might be extracted from the shallow NDS approach tested by KEP. Let us now answer these questions one by one. *Do the limitations of shallow systems such as that embodied in KEP mean that they are doomed to remain below a "glass ceiling" of achievement?* Undoubtedly, yes. Cutting text as it stands from a document will never achieve 100% recall simply because in some cases the exact text is not there to extract. Minor rearrangements of text snippets may be possible, but there will always be cases where only a full understanding could generate a correct concept. Issues relating to endophors, apposition, the *is a* problem, and the existence of complex concepts ensure this. *Will full parsing ultimately be required, or can an acceptable level of performance be achieved without it?* Again, full parsing is probably going to be necessary for many of the functions required in full understanding. However, the author does believe that an acceptable level of performance has been demonstrated, if the answer to the question which follows is 'yes': *Is KEP best used as a "first pass" system?* This is indeed the intention. It was never thought that a perfect glossary could be produced using the approach detailed in this thesis. *Will it ultimately prove necessary to add modules which access knowledge bases?* Yes, for full text understanding, but if the user is happy to accept less than optimal recall and precision rates, then a KB is optional. External resources such as DS MRDs might assist as discussed in section 6.2.12, and indeed are by definition necessary if knowledge from *outside the text* is to be placed in the glossary.

The answers to these questions might seem a trifle disappointing. They confirm the suspicion that only a proper deep treatment will do the job fully. But this was not the ultimate question posed at the start of this thesis. The question posed in the beginning was: *How far can a specific set of shallow techniques go*

for NDS knowledge extraction? The answer to this question is *a surprisingly long way.* The usual initial reaction of those who see the glossary output for the first time is one of puzzlement. They ask: how could a "dumb system" create such a thing without somehow understanding the text? How does it know what to "put in"? How does it find what acronyms stand for, especially when the letters in the acronym "don't match"? Where does it get the text in the third column from? As with all the best tricks, things do not seem so amazing once the methods are explained, or once the output is examined in detail. Nevertheless, the initial puzzlement is an indicator. It indicates that the KEP program appears to be intelligent at first glance, because glossary creation is assumed to be a task that requires intelligence. Recalling Rich and Knight's definition of AI as "the study of how to make computers do things which, at the moment, people do better", and the author's own definition of an AI program as one which is *apparently* intelligent, this makes the KEP program a contender for the title of "AI program".

This thesis started with the premise that *shallow* NLP systems can often be successful and should always be used where possible, because *deep* systems are usually beset with a range of problems arising from the thorough approach taken. The motivation for attempting a shallow system included the desire to build a system that actually works, if not perfectly, then at least to a certain degree of usefulness. Furthermore, it was hoped that taking such a shallow approach would reveal the points at which a deep approach really becomes necessary, i.e. the places in which the shallow approach fails. These limits of the shallow approach have indeed been identified, as discussed above. In summary, the evaluation of KEP has shown that deep approaches do not become necessary until instances of conceptual relations become implicit or distributed in the text. Fortunately, it is the nature of e.g. definitions that they are very often *not* implicit or spread out over several sentences (i.e. they are very often explicit and contained within a single sentence), and so KEP is able to use its shallow approach in many cases. This is in itself an interesting *linguistic* result; by building and evaluating KEP, the nature of e.g. definitions has been explored and the above linguistic situation discovered.

The shallow/deep theme was developed in the first chapter of this thesis, and backed up by descriptions of both deep and shallow systems in the second chapter. The evaluations and discussions in chapters 5 and 6 sought to establish whether a shallow approach could be taken to the NDS task of fact extraction from explanatory text, and if so, how successful this shallow approach could be. The KEP system embodies one such shallow approach, and although it may not be the only such approach, it does use an almost-inevitable method for such systems i.e. pattern matching. The KEP system was developed in an incremental fashion in which problems (such as the need to have a function to give the singular form of a plural noun) were tackled as they arose, using shallow methods.

The results of the exercise reported in this thesis show that a shallow, NDS approach can make good progress in the KE task. They also show that the chosen method cannot, even in theory, hope to extract all facts of the chosen varieties (definitions, examples, parent-class information and lists of the

component parts of objects). Evaluations have shown that one of the target relations is used by writers of explanatory texts in such a way that instances of it are very difficult to extract (examples), whereas others can be more straightforward (definitions). There appear to be two main reasons for the failure of the method in some cases. Firstly, it is difficult to *detect* a definition, example etc in text without a full understanding of that text. Although in many cases this detection is indeed possible using shallow triggering methods, there will always be cases where semantic processing is necessary. Secondly, once a definition, example etc has been detected, it is not always possible to determine *exactly what is being defined* etc. This was found to be particularly true for the exemplification relation. Again, semantic information is needed. It was also suggested that in order to do the job properly the KE system would have to build a representation of the meaning of the text as it scanned through it, much as a human reader builds an understanding construct in the mind as the text is read. Detection of relation instances and determination of what is being elucidated may not in practice be separate processes; both may need to be performed at the same time during the making of such a mental construct.

Despite this, the study has shown that a practically useful shallow knowledge extractor can be built, and so this part of the motivation for attempting a shallow approach has been satisfied. The glossary output format provided by KEP provides a good starting point for the construction of a glossary of terms for a text not written with one. This glossary contains three components which are each in themselves useful, in this and other applications:

(1) The acronym extractor is a novel tool that could be used to automate construction of dictionaries of abbreviations. It could be incorporated in a simple fashion into any text processing program that needed such a function. With its high recall and precision figures it could make a real contribution to any such program.

(2) The technical term (TT) extractor finds lists of specialist terms in an explanatory text. Although partly based on a method first suggested by others, it has been enhanced using hypernym detection methods to tackle the case of single-word terms, a problem acknowledged by Justeson and Katz (1995) and others. Furthermore, it incorporates attempts to avoid obviously wrong terms (so-called 'duff' terms), another novel aspect. Unlike other attempts, it also utilises part of speech tags as given by a tagger program. The result is a new TT extractor with good recall and precision that provides capabilities which are demonstrably useful.

(3) The conceptual relation extractor is a novel attempt to build a shallow fact extractor. Using its pattern-matching approach it is able to find concepts and their elucidations where explicitly stated within single sentences. Since these are frequently used by writers of explanatory texts, the method has some degree of success. Although others have suggested the use of textual patterns to find occurrences of definition, exemplification etc (e.g. Ahmad and Fulford (1992)), such methods have

not to date been built into an actual computer program for evaluation. KEP appears to be the first program to do this.

The way in which these three types of knowledge extraction have been combined is also novel. The author is not aware of any other attempts to create a glossary automatically, i.e. without a writer having to pre-identify a set of glossary terms. Cross referencing acronyms to automatically-acquired TTs provides a more comprehensive list of specialist terms than is possible with other automatic current TT extraction methods. It may also give rise to better ways of doing tasks such as "more of the same" document IR. The addition of the third glossary column, the pattern-matched definitions etc, adds another dimension to the endeavour. Glossaries exist to explain terms within the document, and so this aspect is desirable. Even where the third column entry is not strictly the right sort of phrase for the specific relation being reported, it often contains knowledge useful enough to be placed there. With the addition of a modest amount of cross-referencing between glossary entries (so that, for example, a reader can understand an acronym present in a definition without having to search the glossary for it) the result is an editable block of text which provides a firm basis for a complete glossary. It is not claimed that KEP's glossary is the finished article; it is, however, a good starting point for someone faced with the huge task of creating a comprehensive glossary for a large extant document.

The research reported upon in this thesis has given rise to several exciting potential applications in diverse fields, as discussed in the latter part this chapter. In addition, several areas for future KEP development have been highlighted. Thus there is much interesting research to be done. At the time of writing the author is engaged in collaborative research with two other groups (automatic IR on the Internet, automatic abstraction-finding for software project estimation). Thus there is much scope for both the author and future researchers to follow up, improve and extend the work reported upon in this thesis.

References

References are given in alphabetical order. Where one author (or author combination) has more than one entry, date of publication order is used. Where more than one such paper occurs in one year, lowercase letters are appended to the year in order to distinguish the papers. Note that where there are four or more authors, the author list is referred to in the body of the thesis as first author *et al*; the full author list is always given below.

AGARWAL, R. and TANNIRU, M. R. *Knowledge extraction using content analysis* Knowledge Acquisition **3** pp 421 - 444 (1991)

AHMAD, K. *Document Management: The role of terminology* Invited talk, 4th day of Document Conference, Ede, The Netherlands (1995)

AHMAD, K. *A Terminology Dynamic and the Growth of Knowledge: A Case Study in Nuclear Physics and in the Philosophy of Science* Procs. TKE'96, Vienna (1996)

AHMAD, K. and COLLINGHAM, S. *Renewable Terminology* Procs. EURALEX'96, Goteborg, Sweden (1996)

AHMAD, K. and FULFORD, H. *Knowledge Processing 4: Semantic Relations and Their Use in Elaborating Terminology* CS Report No. CS-92-07 University of Surrey, Guildford (1992)

AHO, A.V., HOPCROFT, J. E. and ULLMAN, J. D. *The Design and Analysis of Computer Algorithms* Addison-Wesley (1974)

ALLEN, J. *Natural Language Understanding* 2nd edition, Benjamin/Cummings (1995)

ALLOTT, N., FAZACKERLEY, P. and HALSTEAD, P. *Automated Assessment: Evaluating a Knowledge Architecture for Natural Language Processing* Procs EXPERT SYSTEMS '94 (Cambridge, England 12-14 Dec. (1994)

ALSHAWI, H. *Processing Dictionary Definitions with Phrasal Pattern Hierarchies* Computational Linguistics **13** 3-4 (1987)

ANDERSEN, P. M., HAYES, P. J., HUETTNER, A. K., SCHMANDT, L. M., NIRENBURG, I. B. and WEINSTEIN, S. P. *Automatic Extraction of Facts from Press Releases to Generate News Stories* Procs. 3rd Conf. on Applied Natural Language Processing, Morristown (NJ) (1992)

APPELT, D. E., HOBBS, J. R., BEAR, J. ISRAEL, D. and TYSON, M. *FASTUS: A Finite-State Processor for Information Extraction from Real-world Text* Procs. 13th International Joint conference on A.I., Chambery, France (1993)

BEAR, J. *A Morphological Recognizer with Syntactic and Phonological Rules* Procs. COLING-86, Univ. Bonn (1986)

BÉJOINT, H. *Scientific and Technical Words in General Dictionaries* International Journal of Lexicography **1** 4 p. 354 (1988)

BOWDEN, P. R., HALSTEAD, P. and ROSE, T. G. *Knowledge Extraction and Text Analysis Using Conceptual Relation Markers* Procs. LEDAR (Language Engineering for Document Analysis and Recognition), one-day workshop as part of AISB'96 Workshop Series, 2nd April (1996) University of Sussex, Brighton, England (1996a)

BOWDEN, P. R., HALSTEAD, P. and ROSE, T. G. *Extracting Conceptual Knowledge from Text Using Explicit Relation Markers* Procs. 9th European Knowledge Engineering Workshop (EKAW-96), Lecture Notes in Artificial Intelligence no.1076, Springer Verlag (1996b).

BOWDEN, P. R., HALSTEAD, P. and ROSE, T. G. *Dictionaryless English Plural Noun Singularisation Using A Corpus-Based List of Irregular Forms* In *Corpus-based Studies in English - Papers from the Seventeenth International Conference on English Language Research on Computerized Corpora (ICAME 17) Stockholm, May 15 - 19 1996* (Rodopi) (1996c)

BOWDEN, P. R., HALSTEAD, P. and ROSE, T. G. *Endophor Resolution in a Pattern-Matching Knowledge Extraction System* (paper presented at IndiAna workshop, DAARC'96 conference, University of Lancaster) (1996d)

BOWDEN, P. R. and EDWARDS, M. A. *Knowledge Extraction from Corpora for Pedagogical Applications* (paper presented at TALC'96 conference, University of Lancaster) (1996)

BOWDEN, P. R., EVETT, L. and HALSTEAD, P. *Automatic Acronym Acquisition in a Knowledge Extraction Program* Procs. COMPUTERM workshop (ACL-COLING'98), Montreal (1998)

BOWDEN, P. R. *The Use of Automatically Generated Technical Terms in a Document Topic Similarity Metric* (in preparation)

BOWDEN, P. R., EVETT, L. and HALSTEAD, P. *A Corpus-Based Search for the Forms of the Partition Relation* (in preparation)

BOWDEN, P. R., HARGREAVES, M. A. and LANGENSIEPEN, C. S. *Estimation Support by Lexical Analysis of Requirements Documents* (in preparation)

BOWKER, L. *LSP Corpora in NLP: Some Fundamentals and Approaches in the Discipline of Terminology* Procs. CSNLP '95, Dublin (1995)

BROSS, I. D. J., SHAPIRO, P. A. and ANDERSON, B. B. *How Information Is Carried in Scientific Sub-Languages* Science **176** (1972)

BROWN, G. and YULE, G. *Discourse Analysis* CUP Cambridge Textbooks in Linguistics (1983)

BROWNING, D. C. (ed.) *Roget's Thesaurus The Everyman Edition* Pan Books (1978)

BURKERT, G. *Lexical semantics and terminological knowledge representation* In SAINT-DIZIER, P. and VIEGAS, E. (Eds) *Computational Lexical Semantics* Cambridge University Press (1995)

BURNARD, L. (ed.) *Users Reference Guide for the British National Corpus (Version 1.0)* Oxford University Computing Services (1995)

BURSTEIN, J. and KAPLAN, R. *Parsing Sentence Fragments in Computer-Assisted Test Scoring* In WILSON, A. and McENERY, T. (Eds.) *Corpora in Language Education and Research: A Selection of Papers from Talc94* UCREL, Dept. of Linguistics and Modern English Language, Lancaster University, England (1994)

CARDIE, C. *Empirical Methods in Information Extraction* A.I. Magazine Winter (1977 p. 65 (1997)

CARNE, C., FURNEAUX, C. and WHITE, R. *Corpora, Genre Analysis and Dissertation Writing: An Evaluation of the Potential of Corpus-Based Techniques in the Study of Academic Writing* Procs. TALC'96, University of Lancaster (1996)

CHARNIAK, E. *Statistical Language Learning* Bradford Books (MIT Press) (1996)

CHODOROW, M. *Extracting Semantic Hierarchies from a Large On-Line Dictionary* Procs. Association for Computational Linguistics pp.299-304 (1985)

CHOMSKY, N. *Syntactic Structures* Mouton (1957)

CHURCH, K. and PATIL, R. *Coping with Syntactic Ambiguity, or how to put the block on the box on the table* Computational Linguistics **8** pp 139 – 149 (1982)

COHEN, D. I. A. *Introduction to Computer Theory* John Wiley and Sons (1986)

COLLIER, R. *An Historical Overview of Natural Language Processing Systems that Learn* Artificial Intelligence Review **8** 17 - 54 (1994)

'COMPUTING' newspaper Notebook Section: *The Ratcliff-Obershelp Algorithm* Computing p.24 (20th August 1992)

CREVIER, D. *AI - The Tumultuous History of the Search for Artificial Intelligence* Basic Books (Harper Collins) (1993)

CROWE, J. *Shallow techniques for the segmentation of news reports.* Procs. LEDAR (Language Engineering for Document Analysis and Recognition), one-day workshop as part of AISB'96 Workshop Series, 2nd April (1996) University of Sussex, Brighton, England (1996)

CRUSE, D. A. *Lexical Semantics* CUP (1986)

CRYSTAL, D. *The Cambridge Encyclopedia of Language* CUP (1987)

DAGAN, I. and ITAI, A. *Automatic Processing of Large Corpora for the Resolution of Anaphora References* Procs. COLING '90 (Helsinki) (1990)

DAILLE, B. *Combined Approach for Terminology Extraction: Lexical Statistics and Linguistic Filtering* UCREL paper no. 5., UCREL, Dept. of Linguistics and Modern English Language, University of Lancaster (1995)

DAINTITH, J., ILLINGWORTH, V., MARTIN, E. and STIBBS, A. (Eds) *The Oxford Dictionary of Abbreviations* (paperback edition), OUP, (1993)

DARIAN, S. *The Role of Definitions in Scientific and Technical Writing: Forms, Functions and Properties* English Language Research Journal **2** pp 41 – 56 (1981)

DeJONG, G. *Prediction and Substantiation: A New Approach to Natural Language Processing* Cognitive Science **3** pp 251 – 273 (1979)

DILLON, M. and GRAY, A. S. *FASIT: A fully automatic syntactically based indexing unit* JASIS **34**:2 pp 99 - 108 (1983)

EDWARDS, M. A., POWELL. H., and PALMER-BROWN, D. *A Hypermedia-based Tutoring and Knowledge Engineering System* In Proceedings. ED-MEDIA '95 Graz, Austria (1995)

ENGUEHARD, C. *Acquisition of a Terminology from Colloquial Texts* Procs. Computational Linguistics for Speech and Handwriting Recognition, one-day workshop as part of AISB'94 Workshop Series, 12th April (1994) University of Leeds, England (1994)

FLOWERDEW, J. L. *Pragmatic modifications on the "representative" speech act of defining* Journal of Pragmatics **15** pp 253 – 264 (1991)

FLOWERDEW, J. L. *Definitions in Science Lectures* Applied Linguistics **13** pp 201 – 221 (1992a)

FLOWERDEW, J. L. *Salience in the Performance of One Speech Act: The Case of Definitions* Discourse Processes **15** pp 165 – 181 (1992b)

FLOWERDEW, L. *CALL Materials Derived from Integrating 'Expert' and 'Interlanguage' Corpora Findings* Pre-publication paper, Language Centre, Hong Kong University of Science and Technology, Clearwater Bay Road, Kowloon, Hong Kong. (1996)

FRIEDMAN, C., HRIPCSAK, G., DuMOUCHEL, W., JOHNSON, S. B. and CLAYTON, P.D. *Natural language processing in an operational clinical information system* Natural Language Engineering **1** 1 pp 83 - 108 (1995)

GARIGLIANO, R., MORGAN, R.G. and SMITH, M.H *The LOLITA System as a Contents Scanning Tool* Procs. 13th International Conference on A.I., Expert Systems and Natural Language Processing, Avignon, France (1993)

GARSIDE, R., LEECH, G. and SAMPSON, G. (Eds) *The Computational Analysis of English* Longman (1987)

GOLDIN, L. and BERRY, D. M. *AbstFinder, A Prototype Natural Language Text Abstraction Finder for Use in Requirements Elicitation* Procs First Int. Conf. on Requirements Engineering, Colorado Springs, CO IEEE Computer Society pp 84 - 93 (1994)

GREENBAUM, S. and QUIRK, R. *A Student's Grammar of the English Language* Longman (1990)

GRISHMAN, R. *Computational Linguistics An Introduction* Cambridge University Press (1986)

GROSS, M. *On The Failure Of Generative Grammar* Language **55** 4 p859 (1979)

GROSZ, B.J. and SIDNER, C.L. *Attention, Intentions, and the Structure of Discourse* Computational Linguistics **12** 3 (1986)

HAHN, U. *Making Understanders Out of Parsers. Semantically Driven Parsing as a Key Concept for Realistic Text Understanding Applications* International Journal of Intelligent Systems **4** (1989)

HAHN, U., KLENNER, M. and SCHNATTINGER, K. *A Quality-Based Terminological Reasoning Model for Text Knowledge Acquisition* Procs. 9th European Knowledge Engineering Workshop (EKAW-96), Lecture Notes in Artificial Intelligence no.1076, Springer Verlag (1996).

HALLIDAY, M. and HASAN, R. *Cohesion in English* Longman (1976)

HAPESHI, K. *Simulating the Development of the Language Lexicon* AISB Quarterly (no. 90) (Winter 1994/95 issue)

HARRÉ, R. *The Philosophies of Science* OUP (1972)

HARTLEY, R. V. *The Transmission of Information* Bell System Technical Journal **3** (1928)

HAYES, P. J. and MOURADIAN, G. V. *Flexible Parsing* American Journal of Computational Linguistics **7** no. 4 pp232 - 242 (1981)

HAYES, P. J., ANDERSEN, P. M., NIRENBURG, I. B. and SCHMANDT, L. M. *TCS: A Shell for Content-Based Text Categorization.* Sixth IEEE AI Applications Conference, Santa Monica (1990)

HEARST, M. A. *Automatic Acquisition of Hyponyms from Large Text Corpora* Procs. COLING-92, Nantes, France (1992)

HEARST, M.A. *Multi-Paragraph Segmentation of Expository Texts* Report No. UCB/CSD 94/790 Computer Sciences Division (EECS) University of California, Berkeley, California 94720 (1994)

HOBBS, J.R. *Coherence and Coreference* Cognitive Science **3** pp 67 - 90 (1979)

HOBBS, J. R., APPELT, D. E., BEAR, J., TYSON, M. and MAGERMAN, D. *Robust Processing of Real-World Natural Language Texts Systems* (In JACOBS, P. S. (ed.) *Text-Based Intelligent Systems* Lawrence Erlbaum Associates (1992))

HOEY, M. *Patterns of Lexis in Text* OUP (1991)

HULS, C., BOS, E. and CLAASEN, W. *Automatic Referent Resolution of Deictic and Anaphoric Expressions* Computational Linguistics **211** (1995)

JACOBS, P. S. *A knowledge framework for natural language analysis* Procs. 10th Int. Conf. on Artificial Intelligence, Milan (1987)

JACOBS, P. S. *To Parse or Not to Parse: Relation-Driven Text Skimming* Proc. 13th Int. Conf. on Computational Linguistics (COLING-90) pp 194-198, Helsinki (1990)

JACOBS, P. S. (ed.) *Text-Based Intelligent Systems* Lawrence Erlbaum Associates (1992)

JENSEN, K. and BINOT, J. *Disambiguating Prepositional Phrase Attachments By Using On-Line Dictionary Definitions* Computational Linguistics **13** 3-4 pp 251-260 (1987)

JENSEN, K. and BINOT, J. *Dictionary Text Entries as a Source of Knowledge for Syntactic and Other Disambiguations* Proc. 2nd Conf. on Natural Language Processing, Feb. (1988, Austin, Texas, USA. (1988)

JOBBINS, A. and EVETT, L. J. *Automatic Identification of Cohesion in Texts: Exploiting the Lexical Organisation of Roget's Thesaurus* In Proceedings ROCLING VIII , Republic of China (Taiwan) (1995)

JOBBINS, A. C. RAZA, G., EVETT, L. J. and SHERKAT, N. *Postprocessing for OCR: Correcting Errors Using Semantic Relations* Procs. LEDAR (Language Engineering for Document Analysis and Recognition), one-day workshop as part of AISB'96 Workshop Series, 2nd April (1996) University of Sussex, Brighton, England (1996)

JOHANSSON, S., ATWELL, E., GARSIDE R. and LEECH, G. *The Tagged LOB Corpus* Norwegian Computer Centre for the Humanities, University of Bergen, Norway (1986)

JUSTESON, J. S. and KATZ, S. M. *Technical Terminology: some linguistic properties and an algorithm for identification in text* Natural Language Engineering **1** 1 pp 9 - 27 (1995)

KARMILOFF-SMITH, A. *Beyond Modularity – A Developmental Perspective on Cognitive Science* MIT Press (1992)

KEENAN, F. *Large Vocabulary Syntactic Analysis for Text Recognition* Ph.D. thesis, Nottingham Trent University (1993)

KIERAS, D. E. *A model of reader strategy for abstracting main ideas from simple technical prose* Text **2** (1-3) pp 47 - 81 (1982)

KITA, K., KATO Y., OMOTO T. and YANO Y. *Automatically Extracting Collocations from Corpora for Language Learning* In WILSON, A. and McENERY, T. (Eds.) *Corpora in Language Education and Research: A Selection of Papers from Talc94* UCREL, Dept. of Linguistics and Modern English Language, Lancaster University, England (1994)

KUHN, T. S. *The Structure of Scientific Revolutions* 2nd Edn. University of Chicago Press (1970)

KUPIEC, J. M. *Robust Part-of-Speech Tagging Using a Hidden Markov Model* Computer Speech and Language **6** pp 225 - 242 (1992)

LAURISTON, A. *Automatic Recognition of Complex Terms: Problems and the TERMINO solution* Terminology **1** 1 pp147 – 170 (1994)

LEECH, G., GARSIDE, R. and BRYANT, M. *CLAWS4: The Tagging of the British National Corpus* Procs. 15th International Conference on Computational Linguistics (COLING 94) Kyoto, Japan pp 622 - 628 (1994)

LEIDNER, J. *Evaluating Taggers for English: Some Evidence* Technical Report no. CLUE-TR-971101, Abteilung für Computerlinguistik, Friedrich-Alexander-Universität, Erlangen-Nürnberg (1997)

LEHNERT, W., CARDIE, C., FISHER, D., McCARTHY, J., RILOFF, E. and SODERLAND, S. *Description of the CIRCUS System as used in MUC-4* Procs. Fourth Message Understanding Conference (MUC-4), San Francisco (1992)

LENAT, D. *A Large-Scale Investment in Knowledge Infrastructure* Comms. ACM, **38** 11 pp.33-38 (1995a)

LENAT, D. *Steps to Sharing Knowledge* Procs. KB+KS'95 (Knowledge Building and Knowledge Sharing (1995)), Ed. MARS, N.J.I. (University of Twente, Enschede, The Netherlands) IOS Press (1995b)

LENAT, D. and GUHA, R. V. *Building Large Knowledge-Based Systems* Addison-Wesley (1990)

LONG, G., POWELL, H. and PALMER-BROWN, D. A Syntax-free NLP Interface for an Intelligent Tutoring Environment Procs. CSNLP '95 (4th International Conference on the Cognitive Science of Natural Language Processing, Dublin City University) (1995)

LOU, W. and FOXLEY, E. *STAMS - A Simple Text Automatic Marking System* Procs. Computational Linguistics for Speech and Handwriting Recognition, one-day workshop as part of AISB'94 Workshop Series, 12th April (1994) University of Leeds, England (1994)

LYONS, J. *Semantics* Cambridge University Press (1977)

LYTINEN, S. L. and GERSHMAN, A. ATRANS: *Automatic Processing of Money Transfer Messages* Procs. 5th Nat. Conf. on AI, Philadelphia (1986)

MAGNAN, M. and OUSSALAH, C. *Exceptions in Composition Graphs* Procs. KB+KS'95 (Knowledge Building and Knowledge Sharing (1995), Ed. MARS, N.J.I. (University of Twente, Enschede, The Netherlands) IOS Press (1995)

MANN, W. C. and THOMPSON, S. A. *Rhetorical Structure Theory: Toward a functional theory of text organisation* Text **8** 3 pp. 243 - 281 (1988)

MANNING, C. D. *Automatic Acquisition of a Large Subcategorisation Dictionary from Corpora* Procs. 31st Annual Meeting of the ACL (1993)

MARKOWITZ, J., AHLSWEDE, T. and EVENS, M. *Semantically Significant Patterns In Dictionary Definitions* Procs. 24th Ann. Meeting of the Association for Computational Linguistics, Columbia University, New York, USA. (1986)

MARTIN, W. *Concept-Oriented Parsing of Definitions* Procs.COLING-92, Nantes Aug 23-28 (1992)

McDONALD, D. D. *Robust Partial-Parsing Through Incremental, Multi-Algorithm Processing* (In JACOBS, P. S. (ed.) *Text-Based Intelligent Systems* Lawrence Erlbaum Associates (1992))

McENERY, T. and WILSON, A. *Corpus Linguistics* Edinburgh University Press (1996)

MEYER, C. F. *A corpus-based study of apposition in English* In *English Corpus Linguistics* (Eds. K. Aijmer and B. Altenberg) Longman (1991)

MEYER, I. and MACKINTOSH, K. *The Corpus from a Terminographer's Viewpoint* International Journal of Corpus Linguistics, **1** 2 pp 257 – 285 (1996)

MICHENER, E. R. *Epistemology, Representation, Understanding and Interactive Exploration of Mathematical Theories* PhD thesis, MIT (1977)

MICHOS, S. E., STAMATATOS, E., FAKOTATIS, N. and KOKKINAKIS, G. *Identification of Functional Style in Unrestricted Texts Based on a Three-Level Stylistic Description.* Procs. LEDAR (Language Engineering for Document Analysis and Recognition), one-day workshop as part of AISB'96 Workshop Series, 2nd April (1996) University of Sussex, Brighton, England (1996)

MILLER, G.A., BECKWITH, R., FELLBAUM, C., GROSS, D. and MILLER, K. J. *Introduction to WordNet: An online lexical database* International Journal of Lexicography **3** 4 pp235-244 (1990)

MITTAL, V. O. and PARIS, C. L. *Categorizing Example Types in Instructional Texts: the need to consider context* Procs. AI-ED 93 (Edinburgh, 23 - 27 Aug.) (1993)

MOORE, J. D. and POLLACK, M. E. *A Problem for RST: The Need for Multi-Level Discourse Analysis* Computational Linguistics **18** 4 pp 537-544 (1992)

MORRIS, J. and HIRST, G. *Lexical Cohesion Computed by Thesaural Relations as an Indicator of the Structure of Text* Computational Linguistics **17** 1 (1991)

MYERS, G. K. and MULGAONKAR, P. *Automatic Extraction of Information from Printed Documents* Procs. 4th Ann. Symposium on Document Analysis and Retrieval, UNLV (University of Nevada, Las Vegas) (1995)

NAKAGAWA, H. *Extraction of Index Words from Manuals* Procs RIAO'97, Montreal (1997)

NG, H. T. and ZELLE, J. *Corpus-Based Approaches to Semantic Interpretation in Natural language Processing* A.I. Magazine Winter 1997 p. 45 (1997)

NISHIDA, F., TAKAMATSU, S., TANI, T. and KUSAKA, H. *Text Analysis and Knowledge Extraction* COLING-86, Univ. Bonn (1986)

NKWENTI-AZEH, B. *Positional and combinational characteristics of terms: Consequences for corpus-based terminography* Terminology **1** 1 (1994)

NOBLE, H.M. *Natural Language Processing* Blackwell Scientific (1988)

NORRIS, J. *Compound Nominal Generation for Information Retrieval: The COMMIX System* Procs. LEDAR (Language Engineering for Document Analysis and Recognition), one-day workshop as part of AISB'96 Workshop Series, 2nd April (1996) University of Sussex, Brighton, England (1996)

OAKES, M. P. and PAICE, C. D. *Term Extraction for Automatic Abstracting* Procs. COMPUTERM workshop (ACL-COLING'98), Montreal (1998)

OUESLATI, R., FRATH, P. and ROUSSELOT, F. *Tools for Acquistion and Exploitation of Terms* Procs. LEDAR (Language Engineering for Document Analysis and Recognition), one-day workshop as part of AISB'96 Workshop Series, 2nd April (1996) University of Sussex, Brighton, England (1996)

PALMER, D. D. and HEARST, M.A. *Adaptive Sentence Boundary Disambiguation* Report No. UCB/CSD 94/797 Computer Sciences Division (EECS) University of California, Berkeley, California 94720 (1994)

PARSONS, D. *Object Oriented Programming with C++* Letts Educational (1997)

PEARSON, J. *The Expression of Definitions in Specialised Texts: A Corpus-based Analysis* Procs. EURALEX '96, Goteborg, Sweden (1996)

PINKER, S. *The Language Instinct* Penguin (1994)

POLYA, G. *How To Solve It - A New Aspect of Mathematical Method* Princeton University Press (1945)

POPPER, K. *Conjectures and Refutations* 4th Edition (revised) Routledge and Kegan Paul (1972)

QIAO, Hong Liang *The mapping between the parsing annotation schemes of the Lancaster Parsed Corpus and the Susanne Corpus* ICAME Journal **19** (1995)

QUIRK, R., GREENBAUM, S., LEECH, G. and SVARTVIK, J. *A Comprehensive Grammar of the English Language* Longman (1985)

RADEMANN, T. *Using online electronic newspapers in modern English-Language press corpora: Benefits and Pitfalls* ICAME Journal **22** (1998)

RAU, L. F. and JACOBS, P. S. *Integrating Top-down and Bottom-up Strategies in a Text Processing System* Proc. 2nd Conf. on Applied Natural language Processing pp 129 - 135 Morristown, NJ, USA (ACL) (1988)

RAU, L. F., JACOBS, P. S. and ZERNIK, U. *Information Extraction and Text Summarization Using Linguistic Knowledge Acquisition* Information Processing and Management **25** 4 pp 419 - 428 (1989)

RICH, E. and KNIGHT, K. *Artificial Intelligence* McGraw-Hill (1991)

RILOFF, E. *Automatically Generating Extraction Patterns from Untagged Text* Procs. 13th Nat. Conference on A. I., Menlo Park, California pp 1044 – 1049 (1996)

RISTAD, E. S. *The Language Complexity Game* MIT Press (1993)

REIMER, U. *Automatic Knowledge Acquisition from Texts: Learning terminological Knowledge via Text Understanding and Inductive Generalization* Procs. 5th AAAI-sponsored Knowledge Acquistion for Knowledge-Based Systems Workshop, Banff, Canada (1989)

ROSE, T.G. and EVETT, L.J. *A large vocabulary semantic analyzer for handwriting recognition* AISB Quarterly, No. 80 (1992)

ROSE, T.G. and EVETT, L.J. *Text Recognition Using Collocations and Domain Codes* Procs. First Annual Workshop on Very Large Corpora, Ohio state University, Columbus, USA (1993a)

ROSE, T.G. and EVETT, L.J. *Semantic analysis for large vocabulary cursive script recognition* Procs. 2nd Int. Assocn. for Pattern Recognition, IAPR Conference on Document Analysis and Recognition, Tsukuba Science City, Japan (1993b)

ROUSSELOT, F., BARTHELEMY, T., De BEUVRON, F., FRATH, P., and OUESLATI, R. *Terminological Competence and Knowledge Acquisition from Texts* 9th European Knowledge Engineering Workshop (EKAW-96), Poster Presentation (1996).

SAGER, N., FRIEDMAN, C., LYMAN, M. S. et al. *Medical Language Processing: Computer Management of Narrative Data* Addison-Wesley (1987)

SALTON, G. *Syntactic Approaches to Automatic Book Indexing* Procs. 26th Ann. Meeting of the ACL, Buffalo, New York (1988)

SAMPSON, G. *Evidence against the "Grammatical" / "Ungrammatical" Distinction* in Corpus Linguistics and Beyond: Procs. 7th Int. Conf. on English Language Research on Computerised Corpora, ed. W. Meijs (Rodopi, Amsterdam) (1987)

SCHANK, R. *Dynamic Memory* Cambridge University Press (1982)

SCHANK, R. and ABELSON, R. *Scripts Plans Goals and Understanding* Lawrence Erlbaum Associates (1977)

SCHWARZ, C., DAVIDSON, G., SEATON, A. and TEBBIT, V. (eds.) *Chambers English Dictionary* Chambers/Cambridge University Press (1988)

SELINKER, L., TRIMBLE, R. M. T. and TRIMBLE, L. *Presuppositional Rhetorical Information in EST Discourse* TESOL Quarterly **10** 3 pp 281 – 290 (1976)

SHANNON, C. E. *A Mathematical Theory of Communication* Bell System Technical Journal **27** (1948)

SKUCE, D., MATWIN, S., TAUZOVICH, B., OPPACHER, F. and SZPAKOWICZ, S. *A logic-based knowledge source system for natural language documents* Data & Knowledge Engineering **1** pp 201 - 231 (1985)

SMITH, N. and WILSON, D. *Modern Linguistics - The Results of Chomsky's Revolution* Penguin (1990)

SOWA, J. F. *Conceptual Structures - Information Processing in Mind and Machine* Addison-Wesley (1984)

SPARCK JONES, K. and WILKS, Y. (Eds) *Automatic Natural Language Processing* Ellis Horwood (1983)

SPARCK JONES, K. *So what about parsing compound nouns?* In SPARCK JONES, K. and WILKS, Y. (Eds) *Automatic Natural Language Processing* Ellis Horwood (1983)

STEDE, M. *The Search for Robustness in Natural Language Understanding* Artificial Intelligence Review **6** pp 383-414 (1992)

SUMMERS, D. *Longman/Lancaster English Language Corpus Criteria and Design* (1991)

SUTCLIFFE, R. F. E., BOERSMA, P., BON, A., DONKER, T., FERRIS, M. C., HELLWIG, P., HYLAND, P., KOCH, H., MASEREEUW, P., McELLIGOT, A., O'SULLIVAN, D., RELIHAN, L., SERAIL, I., SCHMIDT, I., SHEAHAN, L., SLATER, B., VISSER, H. and VOSSEN, P. *Beyond Keywords: Accurate Retrieval from Full Text Documents* Procs. 2nd Language Engineering Convention, London 16-18 Oct. (1995)

SWALES, J. *Definitions in Science and Law – Evidence for Subject-Specific Course Components* Fachsprache **3** p 106 (1981)

SZPAKOWICZ, S. *Semi-Automatic Acquisition of conceptual structure from technical texts* Int. Journal Man-Machine Studies **33** pp 385 - 397 (1990)

TAYLOR, L., GROVER, C. and BRISCOE, T. *The Syntactic Regularity of English Noun Phrases* Procs. COLING-86, Univ. Bonn (1986)

TUCKER, A., NIRENBURG, S. and RASKIN, V. *Discourse and Cohesion in Expository Text* Procs. COLING-86, Univ. Bonn (1986)

VANDER LINDEN, K. and MARTIN, J. H. *Expressing Rhetorical Relations in Instructional Text: A Case Study of the Purpose Relation* Computational Linguistics **21** 1 (1995)

VIRKAR, R. S. and ROACH, J. W. *Direct Assimilation of Expert-Level Knowledge by Automatically Parsing Research Paper Abstracts* International Journal of Expert Systems **1** 4 (1988)

WILKS, Y. *An Intelligent Analyzer and Understander of English* Comms. ACM **18** 5 pp264-274 (1975)

WILSON, A. and McENERY, T. (Eds.) *Corpora in Language Education and Research: A Selection of Papers from Talc94* UCREL, Dept. of Linguistics and Modern English Language, Lancaster University, England. (1994)

WINOGRAD, T. *Language as a Cognitive Process: Syntax* Addison-Wesley (1983)

WINSTON, M. E., CHAFFIN, R. and HERRMAN, D. *A Taxonomy of Part-Whole Relations* Cognitive Science **11** pp 417-444 (1987)

WOTHKE, K. *Machine Learning of Morphological Rules by Generalization and Analogy* Procs. COLING-86, Univ. Bonn (1986)

XUELAN, F. and KENNEDY, G. *Expressing Causation in Written English* RELC Journal **23** 1 (1992)

YANG HUIZHONG *A New Technique for Identifying Scientific/Technical Terms and Describing Science Texts* Literary and Linguistic Computing **1** 2 (1986)

YOUNG, S. R. and HAYES, P. J. *Automatic Classification and Summarization of Banking Telexes* Procs. 2nd Conf. on AI Applications, IEEE Computer Society, Dec. (1985)

ZERNIK, U. and JACOBS, P. *Tagging for Learning: Collecting Thematic Relations from Corpus* Procs. 13th Int. Conf. on Computational Linguistics (COLING-90) (1990)

ZHU, G. and SHADBOLT, N. *Mining Knowledge: The Partial Parsing of Texts* WWW Home page, AI Research Group, Dept. of Psychology, University of Nottingham (1995)

Appendix A – Nomenclature of KE-related Fields

Information Retrieval (IR) This is a discipline in which *whole documents* are retrieved from databases of documents, e.g. by keyword searching. NLP is of interest to IR researchers who wish to create natural language search tools, for example. IR is **not** usually used to mean the extraction of information from within a single document.

Knowledge Acquisition (KA) The term KA is usually applied by *expert systems* practitioners to the stage of representation of knowledge after its elicitation from human experts. Such knowledge must then be further processed, e.g. by KE systems (see discussion on this subject in Agarwal and Tanniru (1991)).

Knowledge Extraction (KE) KE is the process of extracting knowledge (facts) from text. This is the topic of this thesis.

Message Understanding (MU) Reference is often made in the literature to MU applications. Message Understanding (MU) is a branch of NLP closely related to KE and refers to computerised understanding of messages such as naval signals, newswire stories etc. The Message Understanding Conferences (MUC) are regular international competitions designed to foster practical MU progress via public testing of rival computer programs. MUC applications are discussed in sections 2.3 and 2.4 of this thesis.

Information Extraction (IE) Information extraction is a term often applied in MU applications, to mean KE. Here, however, the 'K' is really information, because of the possibly transient nature of the information gleaned from the message. (See section 1.1 for a discussion of the relative merits of the terms 'knowledge' and 'information'.)

Data Mining This is the discovery of interesting data as a result of trawling through large existing databases. The information uncovered was always 'there' in the database, but in an implicit form. Data mining does not usually apply to textual resources and tends to extract information rather than knowledge. It is therefore not of direct interest here.

Content Analysis or **Topic Analysis** or **Text Classification** This discipline aims at automating the decision as to which subject area a particular text belongs to, i.e. at identifying the topic of a piece of text. In the sense that it extracts some kind of information from a text (albeit at a very high level) it may be regarded as a type of IE. However, it is not an area which is of direct interest in this report.

Text Summarisation This is a fairly new discipline aimed at automatically producing summaries of texts. Again, this may be regarded as a form of IE, since the "essence" of a text is extracted. However, this is an area not of direct relevance to the work reported here.

Stylistics A branch of NLP/computational linguistics which examines writing styles, either by genre or on an individual by individual basis. Although not of direct relevance in this report, stylistics can aid in KE and content analysis since it finds patterns of surface forms used to express knowledge in text (see e.g. Michos et al. (1996)).

Terminology Extraction Of direct relevance to this thesis, this field aims to extract words and phrases from documents which are characteristic of the concepts or abstractions discussed in the text. See e.g. Justeson and Katz (1996) and the discussions of alternative systems in Chapter 5.

Appendix B - Term Summaries Output Sample

The example below of part of a term summaries output was derived from BNC text 'B1G'. Only a small percentage of the full term summaries listing has been reproduced, since the full listing is very much longer than the original text, because several different terms may occur in any one sentence.

```
KEP VERSION 98

********** TERM SUMMARIES OUTPUT FOR USER-ENTERED IDENTIFIER 'B1G' **********


At present, this file merely shows blocks of sentences which contain each term.
However, these blocks do contain 'gapping' sentences where the gaps are small.



------------------------------------------------------------------------------
GIS             geographical information systems
------------------------------------------------------------------------------

   1 The areal interpolation problem : estimating population using remote sensin
     g in a GIS framework Mitchel Langford , David .
   2 Maguire and David .
   3 Unwin Introduction !
   4 Data integration is one of the fundamental GIS operations ( Burrough (1986 )
      .
.
.
.
  22 A common problem in geographical information systems ( GIS ) , and one whic
     h has been known about for many years in the context of choropleth mapping
     , is that of producing maps from population data aggregated over selected a
     rbitrary areal units .
.
.
.
  43 In a truly integrated GIS framework ( Jackson and Mason (1986 ) it is almost
      certain to be the case that other potentially useful information is availa
     ble .
.
.
.
  47 In this chapter , we develop a method similar to Flowerdew 's in which we u
     se GIS techniques to enable areal interpolation to be informed by the distr
     ibution of land -cover types , as inferred from a classified Landsat Themat
     ic Mapper ( TM ) image , in both the source ( (1981 Census wards ) and targe
     t ( National Grid kilometre squares ) units .
.
.
.
  98 ERDAS GIS functions were used to overlay the ward boundaries on to the clas
     sified image and then count the number of pixels of each recognized land -c
     over type within each ward .
.
.
.
 172 Given that the remotely sensed data add a great deal of information to thes
     e processes this is hardly surprising , but this general approach is relati
     vely easy to carry out in a GIS environment .
.
.
.
 178 The absence of facilities within GIS software for handling the effects of i
     nput data uncertainty and possible error propagation by GIS operations crea
     tes a question mark over the safe utilization of many aspects of the techno
     logy .
 179 The problem arises because it is thought that the positional errors and att
     ribute uncertainties which are characteristic of all spatial databases , ma
     y be propagated and amplified by GIS operations and thus adversely affect s
     ome or all applications .
 180 These input data uncertainties are attributable to a number of sources rang
```

ing from errors in the original cartographic map documents through to the e ffects of the GIS operations themselves .

187 The advent of GIS has significantly changed all this .
188 The ease of use and flexibility of GIS allow the user to perform operations on map data that were previously impossible on a large scale .
189 The typical end -user of GIS output will probably care or know little about the cartographic and uncertainty characteristics of the map data being use d , while the GIS itself has no procedures for handling the varying accurac y and reliability of the digital map data being processed .
(190 A GIS gives the user complete freedom to combine , overlay and analyse data from many different sources , regardless of scale , accuracy , resolution and quality of the original map documents and without any regard for the ac curacy characteristics of the data themselves .
(191 The mixing of geographical information from different map scales and source s is a key aspect of GIS functionality , but it does raise the question as to what effects the combination of different levels of data uncertainty has on both the output maps and on the data derived from spatial query and ana lysis .
(192 It must be recognized that there are many good reasons for wishing to combi ne data in these ways , but a major problem arises because GIS packages fai l to offer any means of keeping track of the effects of error propagation a nd how it affects the results .

(196 This chapter is concerned with developing methods able to provide estimates of the confidence regions around GIS map -based outputs by taking into acc ount certain selected sources of uncertainty affecting spatial databases .
(197 A Monte Carlo simulation -based approach is used as a general means of esti mating the effects of input data uncertainty on the map outputs after an ar bitrary sequence of GIS operations .
(198 The objective is to identify and handle the effects of data uncertainty in a GIS by defining uncertainty envelopes to create " credibility regions " a round the results .
(199 This is considered to be the minimum needed to allow a GIS to function in a mixed data environment .
200 Sources of error in GIS Error and uncertainty are common features of cartog raphic information , so it is hardly surprising that these aspects are also present in digital versions of analogue maps .
201 It follows , therefore , that no map -related spatial data exist which are wholly error -free .
202 There are many different causes of uncertainty and those which are explicit ly due to GIS -based manipulations of geographic information are merely a m ore recent problem .
203 However , it is also obvious that the power of GIS has the potential dramat ically to increase both the magnitude and importance of errors in spatial d atabases .

214 Digitizing error Despite the availability of hardware for the automated con version of geographic data from paper maps to digital form (e.g. optical s canners) much data input to GIS is still done by hand using a digitizing t able .
215 As a result of human and other complicating factors involved , a high level of error is often present in digital map data .
216 Manual digitizing is consequently recognized as a significant source of map error in GIS (Otawa (1987 ; Keefer etalia (1988) .
217 However , error introduced into digital map databases through the digitizin g process is often ignored because the characteristics of digitizing error have not been fully defined and because no practical means of handling inpu t data uncertainty exist within proprietary GIS software .

246 Errors in digital overlay analysis Much of the functionality of GIS lies wi th their ability to overlay one or more digital maps for the purposes of Bo olean or network analyses .
247 This kind of map analysis used to be done manually (before the advent of p ractical GIS) by overlaying transparent map sheets , establishing the requ ired spatial relationships and drawing the new map on a clean top sheet wit h felt pens (McHarg (1969) .

252 These questions need to be answered , at least in part , before GIS can rea lize their full potential .

.
.
264 Elimination procedures are available in GIS software to remove sliver polyg
ons on the basis of minimum area (e.g. ARC/INFO , ESRI (1987) .
.
.
.
286 Expanding on this it is possible to identify five major tasks in the study
of error propagation within GIS .
287 These are : 1 .
288 The development of mathematical models to represent the uncertainty charact
eristics of digital map databases ; 2 .
289 The development of procedures for estimating the effects of input data unce
rtainties and their propagation through GIS ; 3 .
290 The application of these models and techniques to a representative range of
case studies to derive empirical estimations of likely error levels in GIS
output ; 4 .
291 The development of techniques to utilize output data uncertainty estimates
; and 5 .
292 The incorporation of the technology as standard GIS tools .
.
.
.
304 Here , a Monte Carlo based simulation procedure for estimating the impact o
f error in GIS is proposed and developed .
.
.
.
312 The randomized input map data are then subject to an arbitrary sequence of
GIS operations .
.
.
.
316 If the GIS output is merely numeric , then the distribution of M results gi
ves some indication of the effects of input data uncertainty .
.
.
.
320 In a vector GIS the set of M different output maps would be rasterized and
a count made of the frequency that each cell appears in the final map .
.
.
.
324 This simulation procedure is totally independent of the error models used a
nd the nature and sequence of the GIS operations employed .
325 The GIS component can include all manner of map manipulation , evaluation a
nd statistical procedures .
.
.
.
330 The resulting " error audits " would also provide a platform for illustrati
ng to vendors the importance of installing error estimators in GIS software
.
331 A practical means of identifying approximate levels of output uncertainty a
lso requires that some basic recommendations are made about how this variab
ility can be retained , used and passed on to subsequent operations and app
lications using the data .
332 Some US research groups appear to be tackling part of this problem by taggi
ng databases with error information .
333 Attention might be better focused on more technical questions such as how t
his error information may be used in GIS and spatial analysis .
334 For example , development of spatial retrieval techniques and nearest neigh
bour analyses which can operate with fuzzy data .
335 Other issues relate to investigating how this uncertainty information can b
est be presented to the user .
336 Finally , the simulation approach and associated error models should be cap
able of being incorporated into standard GIS software .
.
.
.
340 Some of the terminology , therefore , relates to the ARC/INFO GIS software
(see Table 6.1) .
341 The basic sequence of operations The nature of the simulation methodology i
s outlined in Fig. 6.2 .
342 The simulation of input data uncertainty involves replacing the determinist
ic input data values by those from a probability distribution that reflects
an appropriate error model .
343 The GIS operations are then performed and the results saved for evaluation
.
344 The GIS software used in this case study is the widely used ARC/INFO packag
e (ESRI (1987) .

350 Carry out the GIS operations on the perturbed coverage ; 3 .

372 The GIS overlay operations follow next .

376 The object of the final analysis is to create a data set which contains , f
or each raster , the number of times it is simulated to be inside the areas
resulting from the sequence of GIS operations .

381 The GIS operations constitute little more than a sequence of map overlays i
n the form of a Boolean search .
382 This process is re -examined here with particular attention to northern Eng
land to reduce the computational resources required .
383 The object of the overlay exercise is to determine potentially feasible sit
es for the dumping of waste material from the nuclear industry , in particu
lar , low - and intermediate -level radioactive waste .
384 This chapter is not itself concerned with the mechanics or politics of the
matter , the example chosen is merely a convenient one for purposes of illu
strating a very common GIS procedure .

473 This chapter has introduced some of the issues surrounding the propagation
of error in GIS and described the preliminary application of a Monte Carlo
approach to assessing their effects .

478 However , with faster hardware likely to be on the market in the near futur
e and the possibility of the emergence of parallel GIS machines , there is
some justification for believing that extra effort is both worth while and
acceptable .
479 The challenge is to resolve the outstanding questions and perfect the techn
ology as soon as possible .
480 Perusal of Fig. 6.5 indicates that it does appear to work , and that it off
ers a pragmatic solution that could probably be developed further into a ge
neral -purpose GIS " error button " .
481 User interfaces Jonathan .
482 Raper Introduction !
483 Geographical information systems (GIS) make considerable demands on the u
ser : the wide variety of data types recorded in digital maps , the complex
data structures used to organize them and the range of operations availabl
e , amount to a formidable obstacle for most users with standard requiremen
ts .
484 As such , the quality of interfaces to GIS has taken on a considerable impo
rtance in terms of awareness , training and usage , both to the providers o
f GIS software and users of GIS alike (Rhind , etalia (1989) .
485 However , there are many aspects to the definition of an interface for syst
ems as complex as GIS , and the solutions to this problem are developing ex
tremely rapidly at the time of writing .
486 Accordingly , this chapter aims to set out the requirements for a fully con
figured GIS interface , and profiles the development of a new GIS user inte
rface system called UGIX .
487 This model is also used to define a research agenda for the next 5 years ;
the reader may judge the accuracy of this analysis by the commercial realit
y of available systems during the early and mid -(1990s .
488 There is a wide recognition that the problems of poor interfaces are of con
siderable importance to the development of GIS .
489 The UK Government Committee of Enquiry chaired by Lord Chorley (DoE (1987)
on the Handling of Geographic Information suggested in recommendation 59 t
hat GIS technology projects be promoted since the report noted that the exi
sting interfaces to GIS systems were poor .
490 The shortcomings of GIS user environments can be divided into two groups :
1 .

494 The quality of GIS user interfaces is also an important factor in the accep
tance , uptake and efficiency of the integrated GIS which are currently on
the market .
495 In a recent study by Willis and Nutter ((1990) of 136 publicly funded util
ities and municipalities in the UK 57 per cent stated that they were inhibi
ted in their GIS developments by " a lack of staff with the right expertise

" .

496 At a time when there is a national and international shortage of staff skil
led in the computer handling of geographical data (Rhind and Mounsey (1989
) , " ease of use " is a vital criterion for the selection of an appropriat
e GIS .

497 It is generally accepted that a system which is easy to use can help cut re
cruitment and training costs , and help retain staff .

498 Organizations which are in the process of implementing GIS strategies also
appreciate that the " ease of use " factor is a key control over how quickl
y GIS programmes can be implemented , and therefore the speed with which fi
nancial targets for paying off capital costs can be met .

499 It may also be true that " ease of use " can influence the quality of work
done and the effectiveness of a GIS as a decision support system .

500 To illustrate this in the negative , Beard ((1989) for example , showed ho
w " use error " in GIS was an important but neglected aspect of quality con
trol in GIS .

501 In summary , the user interface is a vital element of any GIS .

502 Long ignored as an esoteric aspect of GIS design while GIS development was
driven by the need to extend functionality , the user interface is now begi
nning to attract its due attention .

503 However , the implementation of a GIS user interface involves considerably
more than the improvement of the human -computer interaction (HCI) proces
s .

504 Since GIS are conceptually complex and involve diverse operations ranging f
rom data modelling to geometric transformations , improving the HCI can not
be a complete solution to the improvements of GIS use .

505 Consideration also needs to be given to the embedding of knowledge , task d
efinitions and database view manipulations into such interfaces .

506 A key assumption , therefore , which remains to be tested is whether a GIS
user interface should condition GIS use .

507 While there are many who would argue that a measure of technical knowledge
is desirable in those who use a GIS and a protection against the misuse of
a powerful tool , it must now be established that maximum " achievable " us
e of a software system owes much to the creation of a structured use enviro
nment , with logic controls built into the interface .

.
.
.

518 Work by Smith etalia ((1983) and Sneiderman ((1983) developed the concept
s behind moving and selecting screen representations , which has become imp
ortant to all graphics -oriented applications (such as GIS) .

.
.
.

541 Several GIS have already begun to use GUIs to make their systems more user
friendly using the standard platform interface tools as described above .

.
.
.

545 Hence , the use of a GIS with a GUI can only improve user productivity in "
use " factors , for example by increasing the speed of use and reducing er
rors , and may only help the user with a previously substantial knowledge o
f GIS .

546 Thus , due to the sheer complexity of spatial data and the operations avail
able , this can only be a partial solution to the general problem of user i
nteraction with a GIS (Gould (1989) .

ED enumeration district

---

9 Hearnshaw etalia ((1989) , for example , highlight the problem in the cont
ext of Leicestershire , a county in the Midlands of England , and show the
difficulties of linking enumeration district (ED) , ward , parish and pos
tcode data .

.
.
.

(19 Where the source zones nest hierarchically into the target zones , for exam
ple UK administrative EDs nest exactly in wards , transfer of data from the
source units to the target units is one of simple aggregation .

.
.
.

35 Martin ((1988 , (1989) describes a simple algorithm that uses the ED centro
ids , with a spreading function to allocate people to neighbouring grid squ
ares , which incorporates Tobler 's idea .

.

408 The population data arc rather more problematic , since they are generated from the " centroids " of census enumeration districts (EDs) ; the bounda ries of the census EDs do not conveniently follow a grid , it is assumed th ey have a similar fuzzy tolerance to the other coverages .

451 These data had to be estimated from census ED information .
452 The aggregation process merely allocates to a grid square the populations o f the ED whose " centroid " happens to fall in that grid square .
453 In the case where two or more EDs fall in the same grid square , then the g rid square 's population is the sum of those of the individual EDs .

459 One might buffer the ED centroids , assign populations to the resulting zon es , then determine the population densities .
460 The choice of buffer distance would need to be the subject of some experime nt .
461 The use of Theissen polygons has been suggested , although the statistical properties of processes giving rise to such areas are poor surrogates for d igitized ED boundaries .
462 The varying size of EDs in urban and rural areas , and the discontinuous na ture of the population distribution inside the larger rural EDs is also dif ficult to parametrize .

1351 Within the public domain in the UK we must rely in general on data from the most recent Population Census , the lowest level being that for enumeratio n districts (EDs) .
1352 As is well known (Rhind (1983) these contain on average perhaps 150 househ olds and 400 people .
1353 The boundaries of EDs have not been widely digitized , unlike the higher -l evel electoral wards .
1354 However , the centroids of EDs are available as 100 m grid references and a rtificial ED " polygons " can be created if necessary .
1355 In some instances such data will suffice ; in many cases they will have to suffice as nothing better is available .
1356 But in some areas EDs are very extensive physical units and the shapes will be quite distorted .
1357 Furthermore , the scales at which population estimates are often required m eans that even EDs are too coarse for risk assessments .

1429 This may be quite absurd in some cases , notably where population is spatia lly clustered within a physically large ED .

TM thematic mapper

---

47 In this chapter , we develop a method similar to Flowerdew 's in which we u se GIS techniques to enable areal interpolation to be informed by the distr ibution of land -cover types , as inferred from a classified Landsat Themat ic Mapper (TM) image , in both the source ((1981 Census wards) and targe t (National Grid kilometre squares) units .

62 Obtaining a land -cover classification A Landsat TM image of Leicestershire recorded on a cloud -free day in July (1984 constitutes the basic data sour ce .
63 This area was selected for study because of the authors ' familiarity with it , the adequate rural -urban contrast and the availability of a suitable image .
64 The ground resolution of a TM image is such that a pixel has about a 30 m s ide , which seems appropriate for the scale of analysis used .

ERDAS earth resources data analysis system

---

66 The full seven bands of image data were loaded into an ERDAS (Earth Resources Data Analysis System) software system and ground control point information entered to rectify the image to National Grid coordinates .

.
.
.

97 These data were transferred into ERDAS and rasterized .
98 ERDAS GIS functions were used to overlay the ward boundaries on to the classified image and then count the number of pixels of each recognized land -cover type within each ward .

.
.
.

1280 Merchant etalia ((1987) for example , use the image processing system ERDAS to construct an index of vulnerability to groundwater pollution ; this uses variables such as depth to water table , soil type , slope and so on (see Estes et a .

RGB red-green-blue

---

79 The third component , despite carrying only 6 per cent of the original variance , revealed intra -urban differentiation with a clarity unseen in any RGB (red -green -blue (colour)) , composite taken from the original bands .

.
.
.

83 When these components were displayed as an RGB composite they continued to show good urban differentiation , closely matching both our local knowledge of the area and the ground truth data that were collected .

RMS root mean squares

---

153 Table 5.4 provides a summary of the overall fit in the form of the root mean squares (RMS) of the differences for each map .
154 All point to precisely the same problems as were identified when evaluating the original model fits .
155 The average ward population is 9488 (Table 5.1) and so the RMS errors are comparatively small (the range of RMS errors is from 815.13 for the Shotgun ordinary least squares (OLS) to 1035.29 for the Simple Poisson) .
156 There is little difference in the values obtained using OLS and Poisson regression and the Simple models produce only slightly higher errors .
157 Figure 5.12 shows the differences given using the Shotgun model which has the lowest RMS value .

OLS ordinary least squares

---

155 The average ward population is 9488 (Table 5.1) and so the RMS errors are comparatively small (the range of RMS errors is from 815.13 for the Shotgun ordinary least squares (OLS) to 1035.29 for the Simple Poisson) .
156 There is little difference in the values obtained using OLS and Poisson regression and the Simple models produce only slightly higher errors .

NCGIA national center for geographic information and analysis

---

(195 Such is its importance that the National Center for Geographic Information and Analysis (NCGIA) in the USA , has placed this issue first in its list of research priorities (NCGIA (1989) .

.
.

573 The importance of the cognitive structuring of space is expressed by Mark a
nd Frank ((1989) , who set out the research agenda for the " Spatial Langu
ages " Research Initiative of the US National Center for Geographic Informa
tion and Analysis (NCGIA) .

---

target zone

(19 Where the source zones nest hierarchically into the target zones , for exam
ple UK administrative EDs nest exactly in wards , transfer of data from the
source units to the target units is one of simple aggregation .

33 The target zones are then overlain and the interpolated value is transferre
d into the zones .

---

geographical information

---

22 A common problem in geographical information systems (GIS) , and one whic
h has been known about for many years in the context of choropleth mapping
, is that of producing maps from population data aggregated over selected a
rbitrary areal units .

(191 The mixing of geographical information from different map scales and source
s is a key aspect of GIS functionality , but it does raise the question as
to what effects the combination of different levels of data uncertainty has
on both the output maps and on the data derived from spatial query and ana
lysis .

483 Geographical information systems (GIS) make considerable demands on the u
ser : the wide variety of data types recorded in digital maps , the complex
data structures used to organize them and the range of operations availabl
e , amount to a formidable obstacle for most users with standard requiremen
ts .

587 However , attempts to translate the Geographical Information Systems Tutor
(GISTutor) (Raper and Green (1989) into a number of European languages h
ave encountered two main difficulties : first , the local adoption of Engli
sh for spatial terms (and therefore concepts ?

709 Like Gatrell and Vincent and Shepherd before him , he is concerned with the
lack of detailed geographical information particularly in this case with r
eference to the years between censuses .

---

information system

---

22 A common problem in geographical information systems (GIS) , and one whic
h has been known about for many years in the context of choropleth mapping
, is that of producing maps from population data aggregated over selected a
rbitrary areal units .

483 Geographical information systems (GIS) make considerable demands on the u
ser : the wide variety of data types recorded in digital maps , the complex
data structures used to organize them and the range of operations availabl
e , amount to a formidable obstacle for most users with standard requiremen

ts .

587 However , attempts to translate the Geographical Information Systems Tutor (GISTutor) (Raper and Green (1989) into a number of European languages h ave encountered two main difficulties : first , the local adoption of Engli sh for spatial terms (and therefore concepts ?

604 This may indicate that a different approach is required for " public " spat ial information systems used by " spatial professionals " .

621 (A) containing the screen interfaces , dialogues and spatial command proc essor ; (B) containing a help and information system for a GIS ; and (C) an expert system shell or high -level system access module .

842 In addition , plans for regional research centres and for the data and info rmation systems for IGBP (known as IGBP -DIS) had been announced .

904 It originates from an Italian request to the Council of Ministers in (1973 t o identify environmentally " balanced " and " unbalanced " areas in the Com munity ; the first attempts to do this were unsuccessful and , though by (19 81 it was clear that a new approach based on an environmental information s ystem was the most promising one , funding for this was not secured until 1 985 .

925 This is conceived not as a set of hardware but as a comprehensive informati on system focused on the needs identified by the ESSC (see above) and ant icipating somewhat those of IGBP .

1228 They propose a national radiological spatial information system and , in a pilot project in Cumbria , have integrated several layers of data within AR C/INFO to show what this might involve (Fig. 10.2) .

1401 Is it , we wonder , possible to contemplate bringing together some of these data sources to form a National Online Health Information System (NOHIS) , to parallel that for employment and unemployment (Townsend etalia (1987) ?

1469 This large system , designed for a SUN workstation has the following main e lements : an intelligent user interface ; an information system including k nowledge bases , databases , inference machine and database management syst em ; a simulation system ; a DSS .

---

population data

---

22 A common problem in geographical information systems (GIS) , and one whic h has been known about for many years in the context of choropleth mapping , is that of producing maps from population data aggregated over selected a rbitrary areal units .

23 In the UK , the decennial Census of Population records the number of indivi dual people and households , but almost all population data are reported as various aggregations which ensure that data about individuals can not be r ecovered .

124 Although this Shotgun model gives the best fit to the observed ward populat ion data , it is logically flawed in at least two respects .

.
.
408 The population data arc rather more problematic , since they are generated
from the " centroids " of census enumeration districts (EDs) ; the bounda
ries of the census EDs do not conveniently follow a grid , it is assumed th
ey have a similar fuzzy tolerance to the other coverages .
.
.
.
450 The grid square population data used here are a case in point .
.
.
.
1359 It should be pointed out , however , that the National Radiological Protect
ion Board currently uses 1 km grid square resolution population data from t
he (1971 Census in its radiological protection studies (Hallam etalia (1981
) .

<TERM SUMMARY LISTING TRUNCATED HERE>

Appendix C - Definition Templates, Tokens and Triggers

The full list of definition templates arising due to the experiments described in this thesis is given below. This is essentially a copy of the file defpats.txt . The corresponding token file (deftoks.txt) is also given. Also listed here are the triggers (positive and negative) used to mark sentences as potentially containing definitions (files deftrigs.txt and defntrigs.txt).

```
DEFINITION-TEMPLATES.
C,0,X.
C-0-X.
C=0.
C(X)=0.
X,C-0-X.
X,C=0.
X,C=0,X.
C=0,X.
XdC,X:0.
0=cC.
0=czC.
C=dz0.
Cd0.
XCd0.
X,Cd0.
XdCz0.
XCd"0".
XCd"0",X.
Cd"0".
Cd"0",X.
XCd:"0".
XCd:"0",X.
Cd:"0".
Cd:"0",X.
XCd:0.
Cd:0.
Xd"C"z"0".
Xd"C"z"0",X.
d"C"z"0".
d"C"z"0",X.
C=d0.
C=d=0.
C=Xd0.
d:C=0.
wdCz0.
wXd:C=0.
dCz0.
wvdCz0.
wdC=0.
wdC(X)=0.
wC=0.
wCz0.
"C"=0.
X"C"=0.
XtC,X,=0.
XtC=0.
XnCh0.
X,+Cn0.
X,hCn0.
X,hC-0.
C-0.
Cn0.
Cn0,X.
```

```
TOKENS FOR DEFINITION===^dummy header line
define===d
defined===d
defined the concept===d
defined the concept of===d
defined the concept of a===d
defined the concept of an===d
defines the concept===d
defines the concept of===d
defines the concept of a===d
defines the concept of an===d
defines as===d
defined as===d
can be defined as===d
can also be defined as===d
is defined as===d
is defined to be===d
are defined as===d
describes===d
can be described as===d
Definition===d
definition===d
DEFINITION===d
We define===d
we define===d
let us define===d
Let us define===d
we===w
We===w
we can===w
We can===w
we may===w
We may===w
we might===w
We might===w
let us===w
Let us===w
sometimes===v
often===v
usually===v
occasionally===v
always===v
another===x
again===x
good===x
bad===x
useful===x
helpful===x
as the===z
as a===z
as an===z
as===z
is====
are====
is a====
is an====
is simply====
to be====
was====
are====
were====
will be====
ought to be====
would be====
could be====
```

```
should be====
may be====
might be====
and===+
but===>
not===|
that===t
is called===n
is named===n
is known as===n
is often called===n
is often known as===n
which===h
which exhibits===h
which exhibit===h
in which case===h
where===h
.===.
!===!
,===,
(===(
)===)
:===:
;===;
?===?
"==="

DEFINITION TRIGGER PHRASES~
N.B. USE SPACES AS NECESSARY!~
 define~
define~
Define~
 definition~
definition~
Definition~
DEFINITION~
 is ~
 are ~
 designat~
Designat~
 is called~
 is named~
 regarded as~
Regarded as~
 denoted~
Denoted~
 elucidat~
Elucidat~
 that is~
 i.e. ~
 ie ~
 - ~
 or "~
Known as ~
 known as ~
Called a ~
Called an ~
 called a ~
 called an ~

DEFINITION NEGATIVE TRIGGER PHRASES~
N.B. USE SPACES AS NECESSARY!~
well defined~
badly defined~
wrongly defined~
wrong definition~
```

poor definition~
bad definition~
ill defined~
ill-defined~
not defined~
Not defined~
It is possible~
it is possible~
It is feasible~
it is feasible~
It is certain~
it is certain~
It is even~
it is even~
It is likely~
it is likely~
It is then~
it is then~
is to~
is thus~
is such that~
is due to~
is caused~
There are~
there are~
There is~
there is~
is a type of~

Appendix D - Example KEP Long Output

A full KEP long-format output file (kep.out) is listed here. This corresponds to the short test text given as Figure 8. (For space reasons this output has been reproduced in a small font size.)

```
KEP VERSION 091

KEP LONG OUTPUT FILE FOR INPUT FILE '/research5/pmr/kep/kep.in'

OTHER IDENTIFIER ENTERED BY USER IS '4.4test'

Technical term/acronym detection was selected.
Term look-ahead distance was set to 10 sentences.

Triggering was selected.

Ignore-'is'-triggers was selected.

Extraction was selected.
Only technical term bearing sentences are to be processed.

Input was pre-tagged with '_' as tag attachment.

LINE STRUCTURE OF INPUT TEXT:-
LINE TEXT
---- ----
   0
   1  Sorting_VVG is_VBZ the_AT0 action_NN1 of_IO arranging_VVG data_NN0 items_NN2
into_II some_DD specific_AJ0 order_NN1 ._.
   2  We_PPIS2 can_VM define_VVI a_AT0 sort_NN1 routine_NN1 (_( SR_FO )_) to_TO be_VBI
a_AT0 function_NN1 which_DDQ orders_VVZ a_AT0 list_NN1 of_IO items_NN2 according_II21
to_II22 some_DD criterion_NN1 ._.
   3  Examples_NN2 of_IO SRs_NP0 include_VV0 the_AT0 bubble_NN1 sort_NN1 and_CC the_AT0
quick_AJ0 sort_NN1 ._.
   4  Sort_NN1 routines_NN2 are_VBR composed_VVN of_IO four_MC elements_NN2 :_:
input_NN1 list_NN1 ,_, output_NN1 list_NN1 ,_, sort_NN1 criterion_NN1 and_CC sort_NN1
algorithm_NN1 ._.
   5  An_AT0 example_NN1 of_IO a_AT0 sort_NN1 criterion_NN1 is_VBZ alphabetical_AJ0
order_NN1 ._.
   6  A_AT0 sort_NN1 routine_NN1 is_VBZ a_AT0 type_NN1 of_IO data_NN0 rearrangement_NN1
algorithm_NN1 ,_, or_CC DRA_NP0 ._.
   7  In_II these_DD2 ,_, data_NN0 elements_NN2 are_VBR not_XX themselves_PPX2
altered_VVN ,_, but_CCB their_APPGE order_NN1 of_IO presentation_NN1 is_VBZ changed_VVN
to_TO assist_VVI the_AT0 calling_AJ0 application_NN1 ._.

Searching for lines containing untagged words...

SENTENCE STRUCTURE OF INPUT TEXT:-
SENT TEXT
---- ----
   0 Sorting is the action of arranging data items into some specific order .
   1 We can define a sort routine ( SR ) to be a function which orders a list of
     items according to some criterion .
   2 Examples of SRs include the bubble sort and the quick sort .
   3 Sort routines are composed of four elements : input list , output list , so
     rt criterion and sort algorithm .
   4 An example of a sort criterion is alphabetical order .
   5 A sort routine is a type of data rearrangement algorithm , or DRA .
   6 In these , data elements are not themselves altered , but their order of pr
     esentation is changed to assist the calling application .

BLANK SENTENCE COUNT: 0

TOTAL SENTENCE COUNT: 7


LOOKING FOR TECHNICAL TERMS...

Removing old tttest.out file...
Opening new tttest.out file...
Removing old ttnola.out file...
Opening new ttnola.out file...
```

```
NOTE! compile-time flag use_basic was TRUE, so termtag.txt NOT USED.
Initialising global fixed TT array...

Candidate term 'data_NN0 items_NN2 ' found in sentence 0.
Term 'data item' NOT seen before.
This term occurred only once in sentence 0, so
Sentence look-ahead found no match for 'data_nn0 it'
Therefore candidate term has not been stored in the term array.


Candidate term 'specific_AJ0 order_NN1 ' found in sentence 0.
Term 'specific order' NOT seen before.
This term occurred only once in sentence 0, so
Sentence look-ahead found no match for 'specific_aj0 or'
Therefore candidate term has not been stored in the term array.


Candidate term 'sort_NN1 routine_NN1 ' found in sentence 1.
Term 'sort routine' NOT seen before.
This term occurred only once in sentence 1, so
Look-ahead has detected match for 'sort_nn1 ro' in sentence 3
FILLING TERM SLOT 0 with 'sort routine'


Candidate term 'bubble_NN1 sort_NN1 ' found in sentence 2.
Term 'bubble sort' NOT seen before.
This term occurred only once in sentence 2, so
Sentence look-ahead found no match for 'bubble_nn1 so'
Therefore candidate term has not been stored in the term array.


Candidate term 'quick_AJ0 sort_NN1 ' found in sentence 2.
Term 'quick sort' NOT seen before.
This term occurred only once in sentence 2, so
Sentence look-ahead found no match for 'quick_aj0 so'
Therefore candidate term has not been stored in the term array.


Candidate term 'Sort_NN1 routines_NN2 ' found in sentence 3.
Term already stored.

Candidate term 'input_NN1 list_NN1 ' found in sentence 3.
Term 'input list' NOT seen before.
This term occurred only once in sentence 3, so
Sentence look-ahead found no match for 'input_nn1 li'
Therefore candidate term has not been stored in the term array.


Candidate term 'output_NN1 list_NN1 ' found in sentence 3.
Term 'output list' NOT seen before.
This term occurred only once in sentence 3, so
Sentence look-ahead found no match for 'output_nn1 li'
Therefore candidate term has not been stored in the term array.


Candidate term 'sort_NN1 criterion_NN1 ' found in sentence 3.
Term 'sort criterion' NOT seen before.
This term occurred only once in sentence 3, so
Look-ahead has detected match for 'sort_nn1 cr' in sentence 4
FILLING TERM SLOT 1 with 'sort criterion'


Candidate term 'sort_NN1 algorithm_NN1 ' found in sentence 3.
Term 'sort algorithm' NOT seen before.
This term occurred only once in sentence 3, so
Sentence look-ahead found no match for 'sort_nn1 al'
Therefore candidate term has not been stored in the term array.


Candidate term 'sort_NN1 criterion_NN1 ' found in sentence 4.
Term already stored.

Candidate term 'alphabetical_AJ0 order_NN1 ' found in sentence 4.
Term 'alphabetical order' NOT seen before.
This term occurred only once in sentence 4, so
Sentence look-ahead found no match for 'alphabetical_aj0 or'
Therefore candidate term has not been stored in the term array.


Candidate term 'sort_NN1 routine_NN1 ' found in sentence 5.
Term already stored.
```

```
Candidate term 'data_NN0 rearrangement_NN1 ' found in sentence 5.
Term 'data rearrangement' NOT seen before.
This term occurred only once in sentence 5, so
Sentence look-ahead found no match for 'data_nn0 re'
Therefore candidate term has not been stored in the term array.


Candidate term 'rearrangement_NN1 algorithm_NN1 ' found in sentence 5.
Term 'rearrangement algorithm' NOT seen before.
This term occurred only once in sentence 5, so
Sentence look-ahead found no match for 'rearrangement_nn1 al'
Therefore candidate term has not been stored in the term array.


Candidate term 'data_NN0 rearrangement_NN1 algorithm_NN1 ' found in sentence 5.
Term 'data rearrangement algorithm' NOT seen before.
This term occurred only once in sentence 5, so
Sentence look-ahead found no match for 'data_nn0 rearrangement_nn1 al'
Therefore candidate term has not been stored in the term array.


Candidate term 'data_NN0 elements_NN2 ' found in sentence 6.
Term 'data element' NOT seen before.
This term occurred only once in sentence 6, so
Sentence look-ahead found no match for 'data_nn0 el'
Therefore candidate term has not been stored in the term array.


Candidate term 'their_APPGE order_NN1 ' found in sentence 6.
Term 'their order' NOT seen before.
This term occurred only once in sentence 6, so
Sentence look-ahead found no match for 'their_appge or'
Therefore candidate term has not been stored in the term array.


Candidate term 'calling_AJ0 application_NN1 ' found in sentence 6.
Term 'calling application' NOT seen before.
This term occurred only once in sentence 6, so
Sentence look-ahead found no match for 'calling_aj0 ap'
Therefore candidate term has not been stored in the term array.


Creating hypernym singleword terms from 2-word terms...

                ********** PROBABLE TECHNICAL TERMS **********


Probable technical term 'sort routine'.
  This term occurred 3 times (in sentences 1 3 5 ).

Probable technical term 'sort criterion'.
  This term occurred 2 times (in sentences 3 4 ).

(Probable terms all occur at least twice.)

                ********** UNCONFIRMED TECHNICAL TERMS **********


(Unconfirmed TTs are TTs which the look-ahead mechanism thought
occurred twice but which in fact only occurred once. They do NOT
include TTs which only occurred once and which did NOT get stored
by the look-ahead code - for these you have to look
at the output file tttest.out, which contains all the text
fragments of length 2 and 3 words which KEP thought were
candidates for TT-hood. Note that this file does not contain
fragments of length 4 or more, and that it does contain duplicate
entries. However, it is sorted. It also allows you to see the
various morphological forms that a TT occurred in.)


LOOKING FOR PROBABLE DUFF TERMS...
Done.

TERM ACQUISITION PHASE COMPLETED - 2 TERMS FOUND.

                                - INCLUDING 0 N-P-N TERMS.
```

```
LOOKING FOR ACRONYMS...


Candidate acronym 'SR' found in sentence 1.
   1 We can define a sort routine ( SR ) to be a function which orders a list of
     items according to some criterion .
Acronym 'SR' NOT seen before
FILLING ACRONYM SLOT 0 with 'SR'
Attempting to find expansion for 'SR'...
POSSIBLE ACRO EXPANSIONS
 -------------------------
Candidate no. 4 (sort_NN1 routine_NN1 ) has score 20
Candidate no. 6 (define_VVI a_AT0 sort_NN1 routine_NN1 ) has score 16
BEST SCORER is 'sort_NN1 routine_NN1 ' on 20 points.
This candidate had the following attributes:
  The acronym itself was bracketed in some way.
  Furthermore, the bracketed acronym immediately followed the candidate expansion.
  The acronym was exactly generated from the expansion's word-initial letters.
 Stored expansion 'sort routine' in acro slot 0
Attempting to link acronym to a technical term...
Linked acronym 0 (SR - sort routine) to term 0 (sort routine)

Candidate acronym 'SRs' found in sentence 2.
   2 Examples of SRs include the bubble sort and the quick sort .
Acronym 'SR' seen before - in slot 0

Candidate acronym 'DRA' found in sentence 5.
   5 A sort routine is a type of data rearrangement algorithm , or DRA .
Acronym 'DRA' NOT seen before
FILLING ACRONYM SLOT 1 with 'DRA'
Attempting to find expansion for 'DRA'...
POSSIBLE ACRO EXPANSIONS
 -------------------------
Candidate no. 7 (data_NN0 rearrangement_NN1 ) has score 8
Candidate no. 8 (rearrangement_NN1 algorithm_NN1 ) has score 8
Candidate no. 12 (data_NN0 rearrangement_NN1 algorithm_NN1 ) has score 10
Candidate no. 16 (data_NN0 rearrangement_NN1 algorithm_NN1 , ) has score 10
Candidate no. 17 (rearrangement_NN1 algorithm_NN1 , or_CC ) has score 8
Candidate no. (19 (type_NN1 of_IO data_NN0 rearrangement_NN1 algorithm_NN1 ) has score 7
Candidate no. 21 (data_NN0 rearrangement_NN1 algorithm_NN1 , or_CC ) has score 10
Candidate no. 24 (type_NN1 of_IO data_NN0 rearrangement_NN1 algorithm_NN1 , ) has score
7
BEST SCORER is 'data_NN0 rearrangement_NN1 algorithm_NN1 ' on 10 points.
This candidate had the following attributes:
  The acronym was exactly generated from the expansion's word-initial letters.
 Stored expansion 'data rearrangement algorithm' in acro slot 1
Attempting to link acronym to a technical term...
No link made.


                    ********** ACRONYMS **********


Probable acronym 'SR'.
  This acronym stands for 'sort routine'.
  This expansion is also a technical term as found above.
  The acronym occurred 2 times (in sentences 1 2 ).

Probable acronym 'DRA'.
  This acronym stands for 'data rearrangement algorithm'.
  The acronym occurred 1 times (in sentences 5 ).

ADDING UNLINKED ACRONYM EXPANSIONS AS NEW TERMS...

Acronym 'DRA', expansion 'data rearrangement algorithm', has no term link.
Term no. 2, 'data rearrangement algorithm' is now stored.

LOOKING FOR ACRONYMS WITHIN TECHNICAL TERMS (for capital correction)...
...looking for acronym 'sR' in all technical terms...
...looking for acronym 'dRA' in all technical terms...

LOOKING FOR UNKNOWN CAPITALISED WORDS STARTING TERMS (for capital correction)...

LOOKING FOR STRAY CAPITALS (for capital correction)...

LOOKING FOR STRAY LOWERCASE LETTERS IN WORDS (for capital correction)...

                ***** ACRONYM STATISTICS *****

There were 2 possible acronyms found, 2 with expansions found in the text.
```

```
2 of the expanded acronyms were 'exact':
      SR --> sort routine
      DRA --> data rearrangement algorithm

There were 0 expanded acronyms where the expansion was in some way bracketed:

There were 1 expanded acronyms where the acronym was in some way bracketed:
      SR --> sort routine
...and 1 of these had the bracketed acronym IMMEDIATELY after the expansion:
      SR --> sort routine

0 acronyms had the acronym generated from ALL the capitals in the expansion:

0 acronyms had the acronym generated from the word-initial letters
remaining after all glue-words had been deleted:

0 acronyms had been expanded in the text as hyphenated single words:

0 of the 2 probable acros had no expansion near their FIRST occurrence:

NOTE: The above figures reflect the opinion of the KEP acronym extractor,
rather than that of a human reader - the two may not be exactly the same!
This is because the KEP acronym extractor may fail to find some acronym
expansions, or find one where none actually exists.

Note also that counts given are not exclusive; an acronym
may well be bracketed AND be generated from a candidate which
has had all its glue words deleted etc.



LOOKING FOR HYPERNYMS...

REMINDER! No 'is' triggers will be used in this run!


SENTENCE 0
No hypernym-triggers found, so looking for apposition triggers...
No apposition-type triggers detected either.

Sentence 0 not triggered.
Sentence was:
   0 Sorting is the action of arranging data items into some specific order .

SENTENCE 1
No hypernym-triggers found, so looking for apposition triggers...
Just found first part of apposition pattern at position 30
Just found second part of apposition pattern at position2 35
 X(X) apposition trigger detected...  possible hypernym at sent 1, char 30.
Sentence was:
   1 We can define a sort routine ( SR ) to be a function which orders a list of
      items according to some criterion .
Term no. 0 ('sort routine') exists in sentence '1 '
Tokenising sentence...
TOKENS IN THIS SENTENCE ARE:
Instance no. 1, start word 10, end word 11, pattern 'to be', token '='
Instance no. 2, start word 24, end word 24, pattern '.', token '.'
Instance no. 3, start word 7, end word 7, pattern '(', token '('
Instance no. 4, start word 9, end word 9, pattern ')', token ')'
 Tokenisation stage 1 complete.
There were 4 token-instances in this sentence.
(Sentence will be cut up in 15 different ways.)
 Tokenisation stage 2 complete.
 Tokenisation stage 3 complete.
  (There were 15 allowed and 0 not-allowed tokenisations,
   due to token overlaps.)
  Also, of the allowed ones, 13 did not have a LL element
   made for them, because no extraction would have resulted.
   So there were 2 LL elements actually made.
Full list of tokenisations made for this sentence:
X=X.
  word group 0: We can define a sort routine ( SR )
  word group 1: =
  word group 2: a function which orders a list of items according to some criterion
  word group 3: .
X(X)X.
  word group 0: We can define a sort routine
  word group 1: (
  word group 2: SR
```

```
  word group 3: )
  word group 4: to be a function which orders a list of items according to some
criterion
  word group 5: .
 Tokenisation attempt complete.
Sentence had some tokenisation(s).
Tokenisation X=X. matched template C=0.
This is template match no. 0 for this tokenisation.
Tokenisation X(X)X. matched template C(0)X.
This is template match no. 0 for this tokenisation.
CANDIDATE EXTRACTION No. 1
Concept being defined: 'We can define a sort routine ( SR )'
Hypernym given      : 'a function which orders a list of items according to some
criterion'
Unable to validate the concept in any way.
  CONCEPT HAS INVALID SYNTAX - CANDIDATE REJECTED
CANDIDATE EXTRACTION No. 2
Concept being defined: 'We can define a sort routine'
Hypernym given      : 'SR'
Unable to validate the concept in any way.
  CONCEPT HAS INVALID SYNTAX - CANDIDATE REJECTED
AMALGAMATION CANDIDATES ARE AS FOLLOWS:
RELATION AMALGAMATION DONE.
(0 candidate(s) for this amalgamation.)
AMALGAMATED EXTRACTION:
No amalgamated extraction found.

SENTENCE 2 possible hypernym at sent 2, char 17.
Sentence was:
   2 Examples of SRs include the bubble sort and the quick sort .
Acronym no. 0 ('SR') exists in sentence '2 '
Tokenising sentence...
TOKENS IN THIS SENTENCE ARE:
Instance no. 1, start word 8, end word 8, pattern 'and', token '+'
Instance no. 2, start word 12, end word 12, pattern '.', token '.'
 Tokenisation stage 1 complete.
There were 2 token-instances in this sentence.
(Sentence will be cut up in 3 different ways.)
 Tokenisation stage 2 complete.
 Tokenisation stage 3 complete.
  (There were 3 allowed and 0 not-allowed tokenisations,
   due to token overlaps.)
  Also, of the allowed ones, 3 did not have a LL element
   made for them, because no extraction would have resulted.
   So there were 0 LL elements actually made.
Full list of tokenisations made for this sentence:
 Tokenisation attempt complete.
Sentence had no tokenisations.
Unable to extract this possible hypernym due to no template matches.


SENTENCE 3
No hypernym-triggers found, so looking for apposition triggers...
Just found first part of apposition pattern at position 58
Just found second part of apposition pattern at position2 72
 X,X,X apposition trigger detected...  possible hypernym at sent 3, char 58.
Sentence was:
   3 Sort routines are composed of four elements : input list , output list , so
     rt criterion and sort algorithm .
Term no. 0 ('sort routine') exists in sentence '3 '
Tokenising sentence...
TOKENS IN THIS SENTENCE ARE:
Instance no. 1, start word 3, end word 3, pattern 'are', token '='
Instance no. 2, start word 17, end word 17, pattern 'and', token '+'
Instance no. 3, start word 20, end word 20, pattern '.', token '.'
Instance no. 4, start word 11, end word 11, pattern ',', token ','
Instance no. 5, start word 14, end word 14, pattern ',', token ','
Instance no. 6, start word 8, end word 8, pattern ':', token ':'
 Tokenisation stage 1 complete.
There were 6 token-instances in this sentence.
(Sentence will be cut up in 63 different ways.)
 Tokenisation stage 2 complete.
 Tokenisation stage 3 complete.
  (There were 63 allowed and 0 not-allowed tokenisations,
   due to token overlaps.)
  Also, of the allowed ones, 59 did not have a LL element
   made for them, because no extraction would have resulted.
   So there were 4 LL elements actually made.
Full list of tokenisations made for this sentence:
X=X.
  word group 0: Sort routines
```

```
  word group 1: =
  word group 2: composed of four elements : input list , output list , sort criterion
and sort algorithm
  word group 3: .
X=X,X.
  word group 0: Sort routines
  word group 1: =
  word group 2: composed of four elements : input list
  word group 3: ,
  word group 4: output list , sort criterion and sort algorithm
  word group 5: .
X=X,X.
  word group 0: Sort routines
  word group 1: =
  word group 2: composed of four elements : input list , output list
  word group 3: ,
  word group 4: sort criterion and sort algorithm
  word group 5: .
X,X,X.
  word group 0: Sort routines are composed of four elements : input list
  word group 1: ,
  word group 2: output list
  word group 3: ,
  word group 4: sort criterion and sort algorithm
  word group 5: .
 Tokenisation attempt complete.
Sentence had some tokenisation(s).
Tokenisation X=X. matched template C=0.
This is template match no. 0 for this tokenisation.
Tokenisation X=X,X. matched template C=0,X.
This is template match no. 0 for this tokenisation.
Tokenisation X=X,X. matched template C=0,X.
This is template match no. 0 for this tokenisation.
Tokenisation X,X,X. matched template C,0,X.
This is template match no. 0 for this tokenisation.
CANDIDATE EXTRACTION No. 1
Concept being defined: 'Sort routines'
Hypernym given      : 'composed of four elements : input list , output list , sort
criterion and sort algorithm'
Concept is technical term, and so is valid.
  CONCEPT HAS VALID SYNTAX
Rejected elucidation 'composed of four elements : input list , output list , sort
criterion and sort algorithm'
 because it was probably a different relation.
  ELUCIDATION HAS INVALID SYNTAX - CANDIDATE REJECTED
CANDIDATE EXTRACTION No. 2
Concept being defined: 'Sort routines'
Hypernym given      : 'composed of four elements : input list'
Concept is technical term, and so is valid.
  CONCEPT HAS VALID SYNTAX
Rejected elucidation 'composed of four elements : input list'
 because it was probably a different relation.
  ELUCIDATION HAS INVALID SYNTAX - CANDIDATE REJECTED
CANDIDATE EXTRACTION No. 3
Concept being defined: 'Sort routines'
Hypernym given      : 'composed of four elements : input list , output list'
Concept is technical term, and so is valid.
  CONCEPT HAS VALID SYNTAX
Rejected elucidation 'composed of four elements : input list , output list'
 because it was probably a different relation.
  ELUCIDATION HAS INVALID SYNTAX - CANDIDATE REJECTED
CANDIDATE EXTRACTION No. 4
Concept being defined: 'Sort routines are composed of four elements : input list'
Hypernym given      : 'output list'
Unable to validate the concept in any way.
  CONCEPT HAS INVALID SYNTAX - CANDIDATE REJECTED
AMALGAMATION CANDIDATES ARE AS FOLLOWS:
RELATION AMALGAMATION DONE.
(0 candidate(s) for this amalgamation.)
AMALGAMATED EXTRACTION:
No amalgamated extraction found.

SENTENCE 4
No hypernym-triggers found, so looking for apposition triggers...
No apposition-type triggers detected either.

Sentence 4 not triggered.
Sentence was:
   4 An example of a sort criterion is alphabetical order .

SENTENCE 5 possible hypernym at sent 5, char 21.
```

```
Sentence was:
   5 A sort routine is a type of data rearrangement algorithm , or DRA .
Term no. 0 ('sort routine') exists in sentence '5 '
Tokenising sentence...
TOKENS IN THIS SENTENCE ARE:
Instance no. 1, start word 4, end word 7, pattern 'is a type of', token 't'
Instance no. 2, start word 6, end word 7, pattern 'type of', token 't'
Instance no. 3, start word 4, end word 4, pattern 'is', token '='
Instance no. 4, start word 4, end word 5, pattern 'is a', token 't'
Instance no. 5, start word 14, end word 14, pattern '.', token '.'
Instance no. 6, start word 11, end word 11, pattern ',', token ','
 Tokenisation stage 1 complete.
There were 6 token-instances in this sentence.
(Sentence will be cut up in 63 different ways.)
 Tokenisation stage 2 complete.
 Tokenisation stage 3 complete.
  (There were 27 allowed and 36 not-allowed tokenisations,
   due to token overlaps.)
  Also, of the allowed ones, 17 did not have a LL element
   made for them, because no extraction would have resulted.
   So there were 10 LL elements actually made.
Full list of tokenisations made for this sentence:
XtX.
  word group 0: A sort routine
  word group 1: t
  word group 2: data rearrangement algorithm , or DRA
  word group 3: .
XtX.
  word group 0: A sort routine is a
  word group 1: t
  word group 2: data rearrangement algorithm , or DRA
  word group 3: .
X=X.
  word group 0: A sort routine
  word group 1: =
  word group 2: a type of data rearrangement algorithm , or DRA
  word group 3: .
XtX.
  word group 0: A sort routine
  word group 1: t
  word group 2: type of data rearrangement algorithm , or DRA
  word group 3: .
XtX.
  word group 0: A sort routine
  word group 1: t
  word group 2: data rearrangement algorithm , or DRA
  word group 3: .
XtX,X.
  word group 0: A sort routine
  word group 1: t
  word group 2: data rearrangement algorithm
  word group 3: ,
  word group 4: or DRA
  word group 5: .
XtX,X.
  word group 0: A sort routine is a
  word group 1: t
  word group 2: data rearrangement algorithm
  word group 3: ,
  word group 4: or DRA
  word group 5: .
X=X,X.
  word group 0: A sort routine
  word group 1: =
  word group 2: a type of data rearrangement algorithm
  word group 3: ,
  word group 4: or DRA
  word group 5: .
XtX,X.
  word group 0: A sort routine
  word group 1: t
  word group 2: type of data rearrangement algorithm
  word group 3: ,
  word group 4: or DRA
  word group 5: .
XtX,X.
  word group 0: A sort routine
  word group 1: t
  word group 2: data rearrangement algorithm
  word group 3: ,
  word group 4: or DRA
```

```
  word group 5: .
 Tokenisation attempt complete.
Sentence had some tokenisation(s).
Tokenisation XtX. matched template Ct0.
This is template match no. 0 for this tokenisation.
Tokenisation XtX. matched template Ct0.
This is template match no. 0 for this tokenisation.
Tokenisation X=X. matched template C=0.
This is template match no. 0 for this tokenisation.
Tokenisation XtX. matched template Ct0.
This is template match no. 0 for this tokenisation.
Tokenisation XtX. matched template Ct0.
This is template match no. 0 for this tokenisation.
Tokenisation XtX,X. matched template Ct0,X.
This is template match no. 0 for this tokenisation.
Tokenisation XtX,X. matched template Ct0,X.
This is template match no. 0 for this tokenisation.
Tokenisation X=X,X. matched template C=0,X.
This is template match no. 0 for this tokenisation.
Tokenisation XtX,X. matched template Ct0,X.
This is template match no. 0 for this tokenisation.
Tokenisation XtX,X. matched template Ct0,X.
This is template match no. 0 for this tokenisation.
CANDIDATE EXTRACTION No. 1
Concept being defined: 'sort routine'
Hypernym given      : 'data rearrangement algorithm , or DRA'
Concept is technical term, and so is valid.
  CONCEPT HAS VALID SYNTAX
  ELUCIDATION HAS VALID SYNTAX
CANDIDATE EXTRACTION No. 2
Concept being defined: 'sort routine is a'
Hypernym given      : 'data rearrangement algorithm , or DRA'
Unable to validate the concept in any way.
  CONCEPT HAS INVALID SYNTAX - CANDIDATE REJECTED
CANDIDATE EXTRACTION No. 3
Concept being defined: 'sort routine'
Hypernym given      : 'a type of data rearrangement algorithm , or DRA'
Concept is technical term, and so is valid.
  CONCEPT HAS VALID SYNTAX
Rejected elucidation 'a type of data rearrangement algorithm , or DRA'
 because it was probably a different relation.
  ELUCIDATION HAS INVALID SYNTAX - CANDIDATE REJECTED
CANDIDATE EXTRACTION No. 4
Concept being defined: 'sort routine'
Hypernym given      : 'type of data rearrangement algorithm , or DRA'
Concept is technical term, and so is valid.
  CONCEPT HAS VALID SYNTAX
Rejected elucidation 'type of data rearrangement algorithm , or DRA'
 because it was probably a different relation.
  ELUCIDATION HAS INVALID SYNTAX - CANDIDATE REJECTED
CANDIDATE EXTRACTION No. 5
Concept being defined: 'sort routine'
Hypernym given      : 'data rearrangement algorithm , or DRA'
Concept is technical term, and so is valid.
  CONCEPT HAS VALID SYNTAX
  ELUCIDATION HAS VALID SYNTAX
CANDIDATE EXTRACTION No. 6
Concept being defined: 'sort routine'
Hypernym given      : 'data rearrangement algorithm'
Concept is technical term, and so is valid.
  CONCEPT HAS VALID SYNTAX
  ELUCIDATION HAS VALID SYNTAX
CANDIDATE EXTRACTION No. 7
Concept being defined: 'sort routine is a'
Hypernym given      : 'data rearrangement algorithm'
Unable to validate the concept in any way.
  CONCEPT HAS INVALID SYNTAX - CANDIDATE REJECTED
CANDIDATE EXTRACTION No. 8
Concept being defined: 'sort routine'
Hypernym given      : 'a type of data rearrangement algorithm'
Concept is technical term, and so is valid.
  CONCEPT HAS VALID SYNTAX
Rejected elucidation 'a type of data rearrangement algorithm'
 because it was probably a different relation.
  ELUCIDATION HAS INVALID SYNTAX - CANDIDATE REJECTED
CANDIDATE EXTRACTION No. 9
Concept being defined: 'sort routine'
Hypernym given      : 'type of data rearrangement algorithm'
Concept is technical term, and so is valid.
  CONCEPT HAS VALID SYNTAX
Rejected elucidation 'type of data rearrangement algorithm'
```

```
 because it was probably a different relation.
  ELUCIDATION HAS INVALID SYNTAX - CANDIDATE REJECTED
CANDIDATE EXTRACTION No. 10
Concept being defined: 'sort routine'
Hypernym given      : 'data rearrangement algorithm'
Concept is technical term, and so is valid.
  CONCEPT HAS VALID SYNTAX
  ELUCIDATION HAS VALID SYNTAX
AMALGAMATION CANDIDATES ARE AS FOLLOWS:
Amalgamation candidate 1:
  Concept: 'sort routine'
  Elucidation: 'data rearrangement algorithm , or DRA'
Amalgamation candidate 2:
  Concept: 'sort routine'
  Elucidation: 'data rearrangement algorithm , or DRA'
Amalgamation candidate 3:
  Concept: 'sort routine'
  Elucidation: 'data rearrangement algorithm'
Amalgamation candidate 4:
  Concept: 'sort routine'
  Elucidation: 'data rearrangement algorithm'
Amalgamation code: >3-amalgamation-candidate case:
CODE STILL BEING WRITTEN - USING FIRST ONE FOR NOW!
RELATION AMALGAMATION DONE.
(4 candidate(s) for this amalgamation.)
AMALGAMATED EXTRACTION:
Concept being defined: sort routine
Hypernym given      : data rearrangement algorithm , or DRA

SENTENCE 6
No hypernym-triggers found, so looking for apposition triggers...
Just found first part of apposition pattern at position 10
Just found second part of apposition pattern at position2 53
 X,X,X apposition trigger detected...  possible hypernym at sent 6, char 10.
Sentence was:
    6 In these , data elements are not themselves altered , but their order of pr
      esentation is changed to assist the calling application .
No technical term exists in sentence '6 '

HYPERNYMY TRIGGER STATISTICS
The following positive triggers passed the negative triggering stage:
TRIGGER: '(*)'                          COUNT: 1
TRIGGER: 'includ'                       COUNT: 1
TRIGGER: ',*,'                          COUNT: 2
TRIGGER: 'type of'                      COUNT: 1

5 hypernym instance(s) passed triggering stages.
2 hypernym instance(s) DID NOT pass triggering stages.
(2 positive trigger(s) and 0 negative trigger(s) (not including appositions))

THESIS:
Apposition trigs=3,
s(all)=4
s(passed)=2
s(blocked)=2
actual pass rate=50 percent

3 of those had one or more template matches.
And of those 1 had at least one validated extraction.

 ***** 1 HYPERNYM(S) EXTRACTED *****


LOOKING FOR EXEMPLIFICATIONS...

REMINDER! No 'is' triggers will be used in this run!


SENTENCE 0
No example-triggers found, so looking for apposition triggers...
No apposition-type triggers detected either.

(Triggering: position = 0, nposition = 0)
Sentence 0 not triggered.
Sentence was:
   0 Sorting is the action of arranging data items into some specific order .

SENTENCE 1
No example-triggers found, so looking for apposition triggers...
Just found first part of apposition pattern at position 30
```

```
Just found second part of apposition pattern at position2 35
 X(X) apposition trigger detected...  possible exemplification at sent 1, char 30.
Sentence was:
   1 We can define a sort routine ( SR ) to be a function which orders a list of
     items according to some criterion .
Term no. 0 ('sort routine') exists in sentence '1 '
Tokenising sentence...
TOKENS IN THIS SENTENCE ARE:
Instance no. 1, start word 24, end word 24, pattern '.', token '.'
 Tokenisation stage 1 complete.
There were 1 token-instances in this sentence.
(Sentence will be cut up in 1 different ways.)
 Tokenisation stage 2 complete.
 Tokenisation stage 3 complete.
  (There were 1 allowed and 0 not-allowed tokenisations,
   due to token overlaps.)
  Also, of the allowed ones, 1 did not have a LL element
   made for them, because no extraction would have resulted.
   So there were 0 LL elements actually made.
Full list of tokenisations made for this sentence:
 Tokenisation attempt complete.
Sentence had no tokenisations.
Unable to extract this possible exemplification due to no template matches.


SENTENCE 2 possible exemplification at sent 2, char 1.
Sentence was:
   2 Examples of SRs include the bubble sort and the quick sort .
Acronym no. 0 ('SR') exists in sentence '2 '
Tokenising sentence...
TOKENS IN THIS SENTENCE ARE:
Instance no. 1, start word 1, end word 2, pattern 'Examples of', token 'e'
Instance no. 2, start word 8, end word 8, pattern 'and', token '+'
Instance no. 3, start word 12, end word 12, pattern '.', token '.'
Instance no. 4, start word 4, end word 4, pattern 'include', token '='
 Tokenisation stage 1 complete.
There were 4 token-instances in this sentence.
(Sentence will be cut up in 15 different ways.)
 Tokenisation stage 2 complete.
 Tokenisation stage 3 complete.
  (There were 15 allowed and 0 not-allowed tokenisations,
   due to token overlaps.)
  Also, of the allowed ones, 13 did not have a LL element
   made for them, because no extraction would have resulted.
   So there were 2 LL elements actually made.
Full list of tokenisations made for this sentence:
eX=X.
  word group 0: e
  word group 1: SRs
  word group 2: =
  word group 3: the bubble sort and the quick sort
  word group 4: .
eX=X+X.
  word group 0: e
  word group 1: SRs
  word group 2: =
  word group 3: the bubble sort
  word group 4: +
  word group 5: the quick sort
  word group 6: .
 Tokenisation attempt complete.
Sentence had some tokenisations.
Tokenisation eX=X. matched template eC=0.
This is template match no. 0 for this tokenisation.
Tokenisation eX=X+X. matched template eC=0+1.
This is template match no. 0 for this tokenisation.
CANDIDATE EXTRACTION No. 1
Concept being exemplified: 'SRs'
Example given      : bubble sort and the quick sort
Concept is probable acronym, and so is valid.
NOTE: Concept 'SR' was a probable acronym,
      and so has been replaced by 'sort routine' by validation function.
  CONCEPT HAS VALID SYNTAX
  ELUCIDATION HAS VALID SYNTAX
CANDIDATE EXTRACTION No. 2
Concept being exemplified: 'SRs'
Example given      : bubble sort
Example given      : quick sort
Concept is probable acronym, and so is valid.
NOTE: Concept 'SR' was a probable acronym,
      and so has been replaced by 'sort routine' by validation function.
```

```
  CONCEPT HAS VALID SYNTAX
  ELUCIDATION HAS VALID SYNTAX
  ELUCIDATION HAS VALID SYNTAX
AMALGAMATION CANDIDATES ARE AS FOLLOWS:
Amalgamation candidate 1:
  Concept: 'sort routine'
  Elucidation: 'bubble sort and the quick sort' etc
Amalgamation candidate 2:
  Concept: 'sort routine'
  Elucidation: 'bubble sort' etc
EXEMP/PART Amalgamation code: 2-amalgamation-candidate case:-
conc_cand0 is 'sort routine', conc_cand1 is 'sort routine'
 eluc_cand01_0 is
'bubble sort and the quick sort'
 eluc_cand01_1 is
'bubble sort'
(There may be other elucs in addition to this one.)
 2 concept candidates were identical!
 2 FIRST eluc candidates were greater than 30 percent similar, so using the LCS's...

 LCS? str1 was
'bubble sort and the quick sort '
 and str2 was
'bubble sort '
 - LCS is
'bubble sort'

LCS function was passed a blank string - returning the other one.

RELATION AMALGAMATION DONE.
(2 candidate(s) for this amalgamation.)
AMALGAMATED EXTRACTION:
Concept being exemplified: sort routine
Example given     : bubble sort
Example given     : quick sort

SENTENCE 3
No example-triggers found, so looking for apposition triggers...
Just found first part of apposition pattern at position 58
Just found second part of apposition pattern at position2 72
 X,X,X apposition trigger detected...  possible exemplification at sent 3, char 58.
Sentence was:
   3 Sort routines are composed of four elements : input list , output list , so
     rt criterion and sort algorithm .
Term no. 0 ('sort routine') exists in sentence '3 '
Tokenising sentence...
TOKENS IN THIS SENTENCE ARE:
Instance no. 1, start word 17, end word 17, pattern 'and', token '+'
Instance no. 2, start word 20, end word 20, pattern '.', token '.'
Instance no. 3, start word 3, end word 3, pattern 'are', token '='
 Tokenisation stage 1 complete.
There were 3 token-instances in this sentence.
(Sentence will be cut up in 7 different ways.)
 Tokenisation stage 2 complete.
 Tokenisation stage 3 complete.
  (There were 7 allowed and 0 not-allowed tokenisations,
   due to token overlaps.)
  Also, of the allowed ones, 7 did not have a LL element
   made for them, because no extraction would have resulted.
   So there were 0 LL elements actually made.
Full list of tokenisations made for this sentence:
 Tokenisation attempt complete.
Sentence had no tokenisations.
Unable to extract this possible exemplification due to no template matches.


SENTENCE 4 possible exemplification at sent 4, char 3.
Sentence was:
   4 An example of a sort criterion is alphabetical order .
Term no. 1 ('sort criterion') exists in sentence '4 '
Tokenising sentence...
TOKENS IN THIS SENTENCE ARE:
Instance no. 1, start word 2, end word 2, pattern 'example', token 'e'
Instance no. 2, start word 1, end word 3, pattern 'An example of', token 'e'
Instance no. 3, start word 10, end word 10, pattern '.', token '.'
Instance no. 4, start word 7, end word 7, pattern 'is', token '='
 Tokenisation stage 1 complete.
There were 4 token-instances in this sentence.
(Sentence will be cut up in 15 different ways.)
 Tokenisation stage 2 complete.
 Tokenisation stage 3 complete.
```

```
  (There were 11 allowed and 4 not-allowed tokenisations,
   due to token overlaps.)
  Also, of the allowed ones, 10 did not have a LL element
   made for them, because no extraction would have resulted.
   So there were 1 LL elements actually made.
Full list of tokenisations made for this sentence:
eX=X.
  word group 0: e
  word group 1: a sort criterion
  word group 2: =
  word group 3: alphabetical order
  word group 4: .
 Tokenisation attempt complete.
Sentence had some tokenisations.
Tokenisation eX=X. matched template eC=0.
This is template match no. 0 for this tokenisation.
CANDIDATE EXTRACTION No. 1
Concept being exemplified: 'sort criterion'
Example given      : alphabetical order
Concept is technical term, and so is valid.
  CONCEPT HAS VALID SYNTAX
  ELUCIDATION HAS VALID SYNTAX
AMALGAMATION CANDIDATES ARE AS FOLLOWS:
Amalgamation candidate 1:
  Concept: 'sort criterion'
  Elucidation: 'alphabetical order' etc
Amalgamation code: 1 candidate only - so returning it.
RELATION AMALGAMATION DONE.
(1 candidate(s) for this amalgamation.)
AMALGAMATED EXTRACTION:
Concept being exemplified: sort criterion
Example given      : alphabetical order

SENTENCE 5
No example-triggers found, so looking for apposition triggers...
Just found first part of apposition pattern at position 58
No apposition-type triggers detected either.

(Triggering: position = 0, nposition = 0)
Sentence 5 not triggered.
Sentence was:
   5 A sort routine is a type of data rearrangement algorithm , or DRA .

SENTENCE 6
No example-triggers found, so looking for apposition triggers...
Just found first part of apposition pattern at position 10
Just found second part of apposition pattern at position2 53
 X,X,X apposition trigger detected...  possible exemplification at sent 6, char 10.
Sentence was:
   6 In these , data elements are not themselves altered , but their order of pr
     esentation is changed to assist the calling application .
No technical term exists in sentence '6 '

EXEMPLIFICATION TRIGGER STATISTICS
The following positive triggers passed the negative triggering stage:
TRIGGER: '(*)'                          COUNT: 1
TRIGGER: 'Example'                      COUNT: 1
TRIGGER: ',*,'                          COUNT: 2
TRIGGER: ' example'                     COUNT: 1

5 exemplification instance(s) passed triggering stages.
2 exemplification instance(s) DID NOT pass triggering stages.
(2 positive trigger(s) and 0 negative trigger(s) (not including appositions))

THESIS:
Apposition trigs=3,
s(all)=4
s(passed)=2
s(blocked)=2
actual pass rate=50 percent

2 of those had one or more template matches.
And of those 2 had at least one validated extraction.

 ***** 2 EXEMPLIFICATION(S) EXTRACTED *****


LOOKING FOR DEFINITIONS...

REMINDER! No 'is' triggers will be used in this run!
```

```
SENTENCE 0
No definition-triggers found, so looking for apposition triggers...
No apposition-type triggers detected either.

Sentence 0 not triggered.
Sentence was:
   0 Sorting is the action of arranging data items into some specific order .

SENTENCE 1 possible definition at sent 1, char 7.
Sentence was:
   1 We can define a sort routine ( SR ) to be a function which orders a list of
      items according to some criterion .
Term no. 0 ('sort routine') exists in sentence '1 '
Tokenising sentence...
TOKENS IN THIS SENTENCE ARE:
Instance no. 1, start word 3, end word 3, pattern 'define', token 'd'
Instance no. 2, start word 1, end word 1, pattern 'We', token 'w'
Instance no. 3, start word 1, end word 2, pattern 'We can', token 'w'
Instance no. 4, start word 10, end word 11, pattern 'to be', token '='
Instance no. 5, start word 24, end word 24, pattern '.', token '.'
Instance no. 6, start word 7, end word 7, pattern '(', token '('
Instance no. 7, start word 9, end word 9, pattern ')', token ')'
 Tokenisation stage 1 complete.
There were 7 token-instances in this sentence.
(Sentence will be cut up in 127 different ways.)
 Tokenisation stage 2 complete.
 Tokenisation stage 3 complete.
  (There were 95 allowed and 32 not-allowed tokenisations,
   due to token overlaps.)
  Also, of the allowed ones, 89 did not have a LL element
   made for them, because no extraction would have resulted.
   So there were 6 LL elements actually made.
Full list of tokenisations made for this sentence:
XdX.
  word group 0: We can
  word group 1: d
  word group 2: a sort routine ( SR ) to be a function which orders a list of items
according to some criterion
  word group 3: .
X=X.
  word group 0: We can define a sort routine ( SR )
  word group 1: =
  word group 2: a function which orders a list of items according to some criterion
  word group 3: .
wX=X.
  word group 0: w
  word group 1: can define a sort routine ( SR )
  word group 2: =
  word group 3: a function which orders a list of items according to some criterion
  word group 4: .
wX=X.
  word group 0: w
  word group 1: define a sort routine ( SR )
  word group 2: =
  word group 3: a function which orders a list of items according to some criterion
  word group 4: .
wdX=X.
  word group 0: w
  word group 1: d
  word group 2: a sort routine ( SR )
  word group 3: =
  word group 4: a function which orders a list of items according to some criterion
  word group 5: .
wdX(X)=X.
  word group 0: w
  word group 1: d
  word group 2: a sort routine
  word group 3: (
  word group 4: SR
  word group 5: )
  word group 6: =
  word group 7: a function which orders a list of items according to some criterion
  word group 8: .
 Tokenisation attempt complete.
Sentence had some tokenisation(s).
Tokenisation XdX. matched template Cd0.
This is template match no. 0 for this tokenisation.
Tokenisation X=X. matched template C=0.
This is template match no. 0 for this tokenisation.
```

```
Tokenisation wX=X. matched template wC=0.
This is template match no. 0 for this tokenisation.
Tokenisation wX=X. matched template wC=0.
This is template match no. 0 for this tokenisation.
Tokenisation wdX=X. matched template wdC=0.
This is template match no. 0 for this tokenisation.
Tokenisation wdX(X)=X. matched template wdC(X)=0.
This is template match no. 0 for this tokenisation.
CANDIDATE EXTRACTION No. 1
Concept being defined: 'We can'
Definition given     : 'a sort routine ( SR ) to be a function which orders a list of
items according to some criterion'
Unable to validate the concept in any way.
  CONCEPT HAS INVALID SYNTAX - CANDIDATE REJECTED
CANDIDATE EXTRACTION No. 2
Concept being defined: 'We can define a sort routine ( SR )'
Definition given     : 'a function which orders a list of items according to some
criterion'
Unable to validate the concept in any way.
  CONCEPT HAS INVALID SYNTAX - CANDIDATE REJECTED
CANDIDATE EXTRACTION No. 3
Concept being defined: 'can define a sort routine ( SR )'
Definition given     : 'a function which orders a list of items according to some
criterion'
Unable to validate the concept in any way.
  CONCEPT HAS INVALID SYNTAX - CANDIDATE REJECTED
CANDIDATE EXTRACTION No. 4
Concept being defined: 'define a sort routine ( SR )'
Definition given     : 'a function which orders a list of items according to some
criterion'
Unable to validate the concept in any way.
  CONCEPT HAS INVALID SYNTAX - CANDIDATE REJECTED
CANDIDATE EXTRACTION No. 5
Concept being defined: 'sort routine ( SR )'
Definition given     : 'a function which orders a list of items according to some
criterion'
Unable to validate the concept in any way.
  CONCEPT HAS INVALID SYNTAX - CANDIDATE REJECTED
CANDIDATE EXTRACTION No. 6
Concept being defined: 'sort routine'
Definition given     : 'a function which orders a list of items according to some
criterion'
Concept is technical term, and so is valid.
  CONCEPT HAS VALID SYNTAX
  ELUCIDATION HAS VALID SYNTAX
AMALGAMATION CANDIDATES ARE AS FOLLOWS:
Amalgamation candidate 1:
  Concept: 'sort routine'
  Elucidation: 'a function which orders a list of items according to some criterion'
Amalgamation code: 1 candidate only - so returning it.
RELATION AMALGAMATION DONE.
(1 candidate(s) for this amalgamation.)
AMALGAMATED EXTRACTION:
Concept being defined: sort routine
Definition given     : a function which orders a list of items according to some
criterion

SENTENCE 2
No definition-triggers found, so looking for apposition triggers...
No apposition-type triggers detected either.

Sentence 2 not triggered.
Sentence was:
   2 Examples of SRs include the bubble sort and the quick sort .

SENTENCE 3
No definition-triggers found, so looking for apposition triggers...
Just found first part of apposition pattern at position 58
Just found second part of apposition pattern at position2 72
 X,X,X apposition trigger detected...  possible definition at sent 3, char 58.
Sentence was:
   3 Sort routines are composed of four elements : input list , output list , so
     rt criterion and sort algorithm .
Term no. 0 ('sort routine') exists in sentence '3 '
Tokenising sentence...
TOKENS IN THIS SENTENCE ARE:
Instance no. 1, start word 3, end word 3, pattern 'are', token '='
Instance no. 2, start word 17, end word 17, pattern 'and', token '+'
Instance no. 3, start word 20, end word 20, pattern '.', token '.'
Instance no. 4, start word 11, end word 11, pattern ',', token ','
Instance no. 5, start word 14, end word 14, pattern ',', token ','
```

```
Instance no. 6, start word 8, end word 8, pattern ':', token ':'
 Tokenisation stage 1 complete.
There were 6 token-instances in this sentence.
(Sentence will be cut up in 63 different ways.)
 Tokenisation stage 2 complete.
 Tokenisation stage 3 complete.
  (There were 63 allowed and 0 not-allowed tokenisations,
   due to token overlaps.)
  Also, of the allowed ones, 59 did not have a LL element
   made for them, because no extraction would have resulted.
   So there were 4 LL elements actually made.
Full list of tokenisations made for this sentence:
X=X.
  word group 0: Sort routines
  word group 1: =
  word group 2: composed of four elements : input list , output list , sort criterion
and sort algorithm
  word group 3: .
X=X,X.
  word group 0: Sort routines
  word group 1: =
  word group 2: composed of four elements : input list
  word group 3: ,
  word group 4: output list , sort criterion and sort algorithm
  word group 5: .
X=X,X.
  word group 0: Sort routines
  word group 1: =
  word group 2: composed of four elements : input list , output list
  word group 3: ,
  word group 4: sort criterion and sort algorithm
  word group 5: .
X,X,X.
  word group 0: Sort routines are composed of four elements : input list
  word group 1: ,
  word group 2: output list
  word group 3: ,
  word group 4: sort criterion and sort algorithm
  word group 5: .
 Tokenisation attempt complete.
Sentence had some tokenisation(s).
Tokenisation X=X. matched template C=0.
This is template match no. 0 for this tokenisation.
Tokenisation X=X,X. matched template C=0,X.
This is template match no. 0 for this tokenisation.
Tokenisation X=X,X. matched template C=0,X.
This is template match no. 0 for this tokenisation.
Tokenisation X,X,X. matched template C,0,X.
This is template match no. 0 for this tokenisation.
CANDIDATE EXTRACTION No. 1
Concept being defined: 'Sort routines'
Definition given     : 'composed of four elements : input list , output list , sort
criterion and sort algorithm'
Concept is technical term, and so is valid.
  CONCEPT HAS VALID SYNTAX
Rejected elucidation 'composed of four elements : input list , output list , sort
criterion and sort algorithm'
 because it was probably a different relation.
  ELUCIDATION HAS INVALID SYNTAX - CANDIDATE REJECTED
CANDIDATE EXTRACTION No. 2
Concept being defined: 'Sort routines'
Definition given     : 'composed of four elements : input list'
Concept is technical term, and so is valid.
  CONCEPT HAS VALID SYNTAX
Rejected elucidation 'composed of four elements : input list'
 because it was probably a different relation.
  ELUCIDATION HAS INVALID SYNTAX - CANDIDATE REJECTED
CANDIDATE EXTRACTION No. 3
Concept being defined: 'Sort routines'
Definition given     : 'composed of four elements : input list , output list'
Concept is technical term, and so is valid.
  CONCEPT HAS VALID SYNTAX
Rejected elucidation 'composed of four elements : input list , output list'
 because it was probably a different relation.
  ELUCIDATION HAS INVALID SYNTAX - CANDIDATE REJECTED
CANDIDATE EXTRACTION No. 4
Concept being defined: 'Sort routines are composed of four elements : input list'
Definition given     : 'output list'
Unable to validate the concept in any way.
  CONCEPT HAS INVALID SYNTAX - CANDIDATE REJECTED
AMALGAMATION CANDIDATES ARE AS FOLLOWS:
```

```
RELATION AMALGAMATION DONE.
(0 candidate(s) for this amalgamation.)
AMALGAMATED EXTRACTION:
No amalgamated extraction found.

SENTENCE 4
No definition-triggers found, so looking for apposition triggers...
No apposition-type triggers detected either.

Sentence 4 not triggered.
Sentence was:
   4 An example of a sort criterion is alphabetical order .

SENTENCE 5
No definition-triggers found, so looking for apposition triggers...
Just found first part of apposition pattern at position 58
No apposition-type triggers detected either.

Sentence 5 not triggered.
Sentence was:
   5 A sort routine is a type of data rearrangement algorithm , or DRA .

SENTENCE 6
No definition-triggers found, so looking for apposition triggers...
Just found first part of apposition pattern at position 10
Just found second part of apposition pattern at position2 53
 X,X,X apposition trigger detected...  possible definition at sent 6, char 10.
Sentence was:
   6 In these , data elements are not themselves altered , but their order of pr
     esentation is changed to assist the calling application .
No technical term exists in sentence '6 '

DEFINITION TRIGGER STATISTICS
The following positive triggers passed the negative triggering stage:
TRIGGER: ' define'                    COUNT: 1
TRIGGER: ',*,'                        COUNT: 2

3 definition instance(s) passed triggering stages.
4 definition instance(s) DID NOT pass triggering stages.
(1 positive trigger(s) and 0 negative trigger(s) (not including appositions))

THESIS:
Apposition trigs=2,
s(all)=5
s(passed)=1
s(blocked)=4
actual pass rate=20 percent

2 of those had one or more template matches.
And of those 1 had at least one validated extraction.

 ***** 1 DEFINITION(S) EXTRACTED *****


LOOKING FOR PARTITIONS...

REMINDER! No 'is' triggers will be used in this run!


SENTENCE 0
No partition-triggers found, so looking for apposition triggers...
No apposition-type triggers detected either.

Sentence 0 not triggered.
Sentence was:
   0 Sorting is the action of arranging data items into some specific order .

SENTENCE 1
No partition-triggers found, so looking for apposition triggers...
Just found first part of apposition pattern at position 30
Just found second part of apposition pattern at position2 35
 X(X) apposition trigger detected...  possible partition at sent 1, char 30.
Sentence was:
   1 We can define a sort routine ( SR ) to be a function which orders a list of
      items according to some criterion .
Term no. 0 ('sort routine') exists in sentence '1 '
Tokenising sentence...
TOKENS IN THIS SENTENCE ARE:
Instance no. 1, start word 14, end word 14, pattern 'which', token 'W'
Instance no. 2, start word 1, end word 2, pattern 'We can', token 'w'
```

```
Instance no. 3, start word 22, end word 22, pattern 'some', token 'x'
Instance no. 4, start word 10, end word 11, pattern 'to be', token '='
Instance no. 5, start word 24, end word 24, pattern '.', token '.'
Instance no. 6, start word 7, end word 7, pattern '(', token '('
Instance no. 7, start word 9, end word 9, pattern ')', token ')'
 Tokenisation stage 1 complete.
There were 7 token-instances in this sentence.
(Sentence will be cut up in 127 different ways.)
 Tokenisation stage 2 complete.
 Tokenisation stage 3 complete.
  (There were 127 allowed and 0 not-allowed tokenisations,
   due to token overlaps.)
  Also, of the allowed ones, 127 did not have a LL element
   made for them, because no extraction would have resulted.
   So there were 0 LL elements actually made.
Full list of tokenisations made for this sentence:
 Tokenisation attempt complete.
Sentence had no tokenisations.
Unable to extract this possible partition due to no template matches.


SENTENCE 2 possible partition at sent 2, char 16.
Sentence was:
   2 Examples of SRs include the bubble sort and the quick sort .
Acronym no. 0 ('SR') exists in sentence '2 '
Tokenising sentence...
TOKENS IN THIS SENTENCE ARE:
Instance no. 1, start word 5, end word 5, pattern 'the', token 'T'
Instance no. 2, start word 9, end word 9, pattern 'the', token 'T'
Instance no. 3, start word 8, end word 8, pattern 'and', token '+'
Instance no. 4, start word 4, end word 4, pattern 'include', token 'i'
Instance no. 5, start word 12, end word 12, pattern '.', token '.'
 Tokenisation stage 1 complete.
There were 5 token-instances in this sentence.
(Sentence will be cut up in 31 different ways.)
 Tokenisation stage 2 complete.
 Tokenisation stage 3 complete.
  (There were 31 allowed and 0 not-allowed tokenisations,
   due to token overlaps.)
  Also, of the allowed ones, 30 did not have a LL element
   made for them, because no extraction would have resulted.
   So there were 1 LL elements actually made.
Full list of tokenisations made for this sentence:
XiX.
  word group 0: Examples of SRs
  word group 1: i
  word group 2: the bubble sort and the quick sort
  word group 3: .
 Tokenisation attempt complete.
Sentence had some tokenisations.
Tokenisation XiX. matched template Ci0.
This is template match no. 0 for this tokenisation.
CANDIDATE EXTRACTION No. 1
Concept being partitioned: 'Examples of SRs'
Part given     : bubble sort and the quick sort
Unable to validate the concept in any way.
  CONCEPT HAS INVALID SYNTAX - CANDIDATE REJECTED
AMALGAMATION CANDIDATES ARE AS FOLLOWS:
RELATION AMALGAMATION DONE.
(0 candidate(s) for this amalgamation.)
AMALGAMATED EXTRACTION:
No amalgamated extraction found.

SENTENCE 3 possible partition at sent 3, char 18.
Sentence was:
   3 Sort routines are composed of four elements : input list , output list , so
     rt criterion and sort algorithm .
Term no. 0 ('sort routine') exists in sentence '3 '
Tokenising sentence...
TOKENS IN THIS SENTENCE ARE:
Instance no. 1, start word 7, end word 7, pattern 'elements', token 'p'
Instance no. 2, start word 3, end word 5, pattern 'are composed of', token 'k'
Instance no. 3, start word 4, end word 5, pattern 'composed of', token 'k'
Instance no. 4, start word 6, end word 6, pattern 'four', token '$'
Instance no. 5, start word 3, end word 3, pattern 'are', token '='
Instance no. 6, start word 3, end word 3, pattern 'are', token '='
Instance no. 7, start word 17, end word 17, pattern 'and', token '+'
Instance no. 8, start word 20, end word 20, pattern '.', token '.'
Instance no. 9, start word 11, end word 11, pattern ',', token ','
Instance no. 10, start word 14, end word 14, pattern ',', token ','
Instance no. 11, start word 8, end word 8, pattern ':', token ':'
```

```
 Tokenisation stage 1 complete.
There were 11 token-instances in this sentence.
(Sentence will be cut up in 2047 different ways.)
 Tokenisation stage 2 complete.
 Tokenisation stage 3 complete.
  (There were 895 allowed and 1152 not-allowed tokenisations,
   due to token overlaps.)
  Also, of the allowed ones, 886 did not have a LL element
   made for them, because no extraction would have resulted.
   So there were 9 LL elements actually made.
Full list of tokenisations made for this sentence:
XpX.
  word group 0: Sort routines are composed of four
  word group 1: p
  word group 2: : input list , output list , sort criterion and sort algorithm
  word group 3: .
XkX.
  word group 0: Sort routines
  word group 1: k
  word group 2: four elements : input list , output list , sort criterion and sort
algorithm
  word group 3: .
XkX.
  word group 0: Sort routines are
  word group 1: k
  word group 2: four elements : input list , output list , sort criterion and sort
algorithm
  word group 3: .
XpX,X.
  word group 0: Sort routines are composed of four
  word group 1: p
  word group 2: : input list
  word group 3: ,
  word group 4: output list , sort criterion and sort algorithm
  word group 5: .
XpX,X.
  word group 0: Sort routines are composed of four
  word group 1: p
  word group 2: : input list , output list
  word group 3: ,
  word group 4: sort criterion and sort algorithm
  word group 5: .
X=X,X,X.
  word group 0: Sort routines
  word group 1: =
  word group 2: composed of four elements : input list
  word group 3: ,
  word group 4: output list
  word group 5: ,
  word group 6: sort criterion and sort algorithm
  word group 7: .
X=X,X,X.
  word group 0: Sort routines
  word group 1: =
  word group 2: composed of four elements : input list
  word group 3: ,
  word group 4: output list
  word group 5: ,
  word group 6: sort criterion and sort algorithm
  word group 7: .
Xk$p:X,X,X+X.
  word group 0: Sort routines
  word group 1: k
  word group 2: $
  word group 3: p
  word group 4: :
  word group 5: input list
  word group 6: ,
  word group 7: output list
  word group 8: ,
  word group 9: sort criterion
  word group 10: +
  word group 11: sort algorithm
  word group 12: .
Xk$p:X,X,X+X.
  word group 0: Sort routines are
  word group 1: k
  word group 2: $
  word group 3: p
  word group 4: :
  word group 5: input list
```

```
  word group 6: ,
  word group 7: output list
  word group 8: ,
  word group 9: sort criterion
  word group 10: +
  word group 11: sort algorithm
  word group 12: .
 Tokenisation attempt complete.
Sentence had some tokenisations.
Tokenisation XpX. matched template 0pC.
This is template match no. 0 for this tokenisation.
Tokenisation XkX. matched template Ck0.
This is template match no. 0 for this tokenisation.
Tokenisation XkX. matched template Ck0.
This is template match no. 0 for this tokenisation.
Tokenisation XpX,X. matched template 0pC,X.
This is template match no. 0 for this tokenisation.
Tokenisation XpX,X. matched template 0pC,X.
This is template match no. 0 for this tokenisation.
Tokenisation X=X,X,X. matched template C=0,1,2.
This is template match no. 0 for this tokenisation.
Tokenisation X=X,X,X. matched template C=0,1,2.
This is template match no. 0 for this tokenisation.
Tokenisation Xk$p:X,X,X+X. matched template Ck$p:0,1,2+3.
This is template match no. 0 for this tokenisation.
Tokenisation Xk$p:X,X,X+X. matched template Ck$p:0,1,2+3.
This is template match no. 0 for this tokenisation.
CANDIDATE EXTRACTION No. 1
Concept being partitioned: ': input list , output list , sort criterion and sort
algorithm'
Part given     : Sort routines are composed of four
Unable to validate the concept in any way.
  CONCEPT HAS INVALID SYNTAX - CANDIDATE REJECTED
CANDIDATE EXTRACTION No. 2
Concept being partitioned: 'Sort routines'
Part given     : four elements : input list , output list , sort criterion and sort
algorithm
Concept is technical term, and so is valid.
  CONCEPT HAS VALID SYNTAX
Rejected elucidation 'four elements : input list , output list , sort criterion and sort
algorithm'
 because it was probably a different relation.
  ELUCIDATION HAS INVALID SYNTAX - CANDIDATE REJECTED
CANDIDATE EXTRACTION No. 3
Concept being partitioned: 'Sort routines are'
Part given     : four elements : input list , output list , sort criterion and sort
algorithm
Unable to validate the concept in any way.
  CONCEPT HAS INVALID SYNTAX - CANDIDATE REJECTED
CANDIDATE EXTRACTION No. 4
Concept being partitioned: ': input list'
Part given     : Sort routines are composed of four
Unable to validate the concept in any way.
  CONCEPT HAS INVALID SYNTAX - CANDIDATE REJECTED
CANDIDATE EXTRACTION No. 5
Concept being partitioned: ': input list , output list'
Part given     : Sort routines are composed of four
Unable to validate the concept in any way.
  CONCEPT HAS INVALID SYNTAX - CANDIDATE REJECTED
CANDIDATE EXTRACTION No. 6
Concept being partitioned: 'Sort routines'
Part given     : composed of four elements : input list
Part given     : output list
Part given     : sort criterion and sort algorithm
Concept is technical term, and so is valid.
  CONCEPT HAS VALID SYNTAX
Rejected elucidation 'composed of four elements : input list'
 because it was probably a different relation.
  ELUCIDATION HAS INVALID SYNTAX - CANDIDATE REJECTED
CANDIDATE EXTRACTION No. 7
Concept being partitioned: 'Sort routines'
Part given     : composed of four elements : input list
Part given     : output list
Part given     : sort criterion and sort algorithm
Concept is technical term, and so is valid.
  CONCEPT HAS VALID SYNTAX
Rejected elucidation 'composed of four elements : input list'
 because it was probably a different relation.
  ELUCIDATION HAS INVALID SYNTAX - CANDIDATE REJECTED
CANDIDATE EXTRACTION No. 8
Concept being partitioned: 'Sort routines'
```

```
Part given      : input list
Part given      : output list
Part given      : sort criterion
Part given      : sort algorithm
Concept is technical term, and so is valid.
  CONCEPT HAS VALID SYNTAX
  ELUCIDATION HAS VALID SYNTAX
  ELUCIDATION HAS VALID SYNTAX
  ELUCIDATION HAS VALID SYNTAX
  ELUCIDATION HAS VALID SYNTAX
CANDIDATE EXTRACTION No. 9
Concept being partitioned: 'Sort routines are'
Part given      : input list
Part given      : output list
Part given      : sort criterion
Part given      : sort algorithm
Unable to validate the concept in any way.
  CONCEPT HAS INVALID SYNTAX - CANDIDATE REJECTED
AMALGAMATION CANDIDATES ARE AS FOLLOWS:
Amalgamation candidate 1:
  Concept: 'sort routine'
  Elucidation: 'input list' etc
Amalgamation code: 1 candidate only - so returning it.
RELATION AMALGAMATION DONE.
(1 candidate(s) for this amalgamation.)
AMALGAMATED EXTRACTION:
Concept being partitioned: sort routine
Part given      : input list
Part given      : output list
Part given      : sort criterion
Part given      : sort algorithm

SENTENCE 4
No partition-triggers found, so looking for apposition triggers...
No apposition-type triggers detected either.

Sentence 4 not triggered.
Sentence was:
   4 An example of a sort criterion is alphabetical order .

SENTENCE 5
No partition-triggers found, so looking for apposition triggers...
Just found first part of apposition pattern at position 58
No apposition-type triggers detected either.

Sentence 5 not triggered.
Sentence was:
   5 A sort routine is a type of data rearrangement algorithm , or DRA .

SENTENCE 6 possible partition at sent 6, char 16.
Sentence was:
   6 In these , data elements are not themselves altered , but their order of pr
     esentation is changed to assist the calling application .
No technical term exists in sentence '6 '

PARTITION TRIGGER STATISTICS
The following positive triggers passed the negative triggering stage:
TRIGGER: '(*)'                          COUNT: 1
TRIGGER: ' includ'                      COUNT: 1
TRIGGER: ' compos'                      COUNT: 1
TRIGGER: ' element'                     COUNT: 1

4 partition instance(s) passed triggering stages.
3 partition instance(s) DID NOT pass triggering stages.
(3 positive trigger(s) and 0 negative trigger(s) (not including appositions))

THESIS:
Apposition trigs=1,
s(all)=6
s(passed)=3
s(blocked)=3
actual pass rate=50 percent

2 of those had one or more template matches.
And of those 1 had at least one validated extraction.

 ***** 1 PARTITION(S) EXTRACTED *****


***** 1 HYPERNYM(S), 2 EXEMPLIFICATION(S), 1 DEFINITION(S), 1 PARTITION(S) *****
```

```
THESIS: Amalgamated PRESENTATIONAL counts for all 4 relation types:

There were 13 highlighted sentences. (h)
There were 0 presentational sentences detected (p(KEP)).
There were 13 non-presentational sentences detected.

Condensing concepts from separate linked lists...
Opening hypernym list...
Concept 'sort routine' is NOT already in condensed LL. Creating new element...
All hypernyms condensed.
Opening definition list...
Concept 'sort routine' is already in condensed LL. Merging...
All definitions condensed.
Opening exemplification list...
Concept 'sort routine' is already in condensed LL. Merging...
Concept 'sort criterion' is NOT already in condensed LL. Creating new element...
All exemplifications condensed.
Opening partition list...
Concept 'sort routine' is already in condensed LL. Merging...
All partitions condensed.

CONDENSED CONCEPT LIST:-


Concept: sort routine
Definition: a function which orders a list of items according to some criterion
Hypernym: data rearrangement algorithm , or DRA
Example: bubble sort
Example: quick sort
Part: input list
Part: output list
Part: sort criterion
Part: sort algorithm

Concept: sort criterion
Example: alphabetical order

Making glossary file...


   Processing acronyms...


   Processing remaining TTs...


   Processing elucidations...

About to write glossary to file...

GLOSSARY MADE SUCCESSFULLY

TERM SUMMARIES MADE SUCCESSFULLY

END OF KNOWLEDGE EXTRACT
```

Appendix E - KEP-made Glossary for Chapters 1 to 4 of This Thesis

This Appendix contains the complete glossary output file made by KEP when passed the first four chapters of the submitted version of this thesis as input. Refer to section 5.4 for details.

KEP VERSION 103

********** GLOSSARY OUTPUT FOR USER-ENTERED IDENTIFIER 'chaps1to4' **********

ACRONYM	TERM	EXPLANATION
	3-char identifier	
	3G language	Definition: technical term within the text. Examples: missed, PASCAL, PASCAL, PASCA, PASCAL and PASCA. SEE ALSO technical term, text, term
	acquisition function	
	acquisition method	
	acquisition system	
	acquisition	
	acronym acquisition	
	acronym extraction	
	acronym extractor	
	acronyms expansion	
APN	activation passing network	
AVP	adverb particles	
	AI practitioner	SEE ALSO artificial intelligence
	AI system	SEE ALSO artificial intelligence
	algorithm	Definition: described as follows. Parts: looking for strings using the following rules. SEE ALSO rule, string
	alphabetical order	
	alshawis MRD analyser	SEE ALSO machine readable dictionary
	alshawis MRD	SEE ALSO machine readable dictionary
	alternative fragmentation	
	alternative way	
	amalgamation process	Definition: important area for future improvement since the correct extraction is usually present within the amalgamation candidate set when the set is not actually

		empty. SEE ALSO correct extraction, extraction, set, candidate
	american rodent	
	amount of domain	
	amount of memory	
	amount of processing	
	amount of text	
	amount of WK	SEE ALSO world knowledge
	amount	Definition: transferred etc.
	analyser	
	analysis rule	
	analysis	
	anaphoric link	
	annotation scheme	
	anomalous example	
	apparent failure	
	application	
	apposition syntax	
	appositive phrase	Definition: , a hypernym , a description of the components of the concept , a statement of the material it is made from , and so on. SEE ALSO description, concept
	approach	Definition: one which uses term acquisition techniques (see also section 4.6.4) Definition: fully detailed in Bowden , Halstead and Rose (1996c) , but the major points will be outlined below. SEE ALSO term acquisition, technique, section, term, acquisition
	aquatic mammal	Parts: walruses.
	area of research	
	array	
AI	artificial intelligence	Definition: study of how to make computers do things which , at the moment , people do better. SEE ALSO computer
	ASCII text	
	associated software sale	Definition: forecast at 645 million pounds sterling in 1993. SEE ALSO million pound, pounds sterling, million pounds sterling
	associated software	
	atomic nucleus	
	ATRANS system	SEE ALSO n is a noun
ATN	augmented transition networks	

	authors addition	
	automatic construction	
	automatic creation	
	automatic glossary creation	Definition: not a trivial task. SEE ALSO trivial task, task
	automatic glossary maker	Examples: , than for an automatic encyclopaedia constructor.
	automatic glossary	
	automatic index creation	
	automatic index	
	automatic marking	
	automatic system	
	automatic technique	Parts: that of Crowe (1996) in the CONTESS system. SEE ALSO system
BNF	backus-Naur form	
	bare template	Definition: a pattern of three heads , such as MAN GIVE THING. Type of: pattern of three heads. SEE ALSO pattern
	basic unit	
	basis	
	black dog	
	BNC file name	SEE ALSO british national corpus
	BNC file	SEE ALSO british national corpus
	BNC text	SEE ALSO british national corpus
	BNC	SEE ALSO british national corpus
	body of text	
	border plant	
	boundary exception phrase	
	boundary exception	
	branch of AI	SEE ALSO artificial intelligence
BBC	British broadcasting corporation	
BNC	british national corpus	Definition: fully part -of -speech tagged , an extremely useful property which will be referred to in some detail later (see Chapter 4 for details of the tagger , CLAWS4). SEE ALSO part, tagger, tag
BSI	british standards institute	
	brown corpus	
	brown dog	

bubble sort	
c BNC file	SEE ALSO british national corpus
c BNC	SEE ALSO british national corpus
c token	
c vertical-format LOB	SEE ALSO lancaster oslo/Bergen corpus
CAL system	SEE ALSO computer aided learning
candidate expansion	
candidate extraction	
candidate string	
candidate	
carnegie group	
case of terms	
case template	
categorisation scheme	
categorisation	
central finding	
certainty rating	
chain of events	
chain reaction	
chapter summary	
chart parser	
chomsky hierarchy	
chosen relation	
class inclusion	
class of items	
class of objects	
class word	
class	Examples: regarded as useful facts. SEE ALSO useful fact, fact
CLAWS tagger	SEE ALSO constituent likelihood automatic word tagging system
CLAWS-tagged input text	SEE ALSO constituent likelihood automatic word tagging system
CLAWS-tagged input	SEE ALSO constituent likelihood automatic word tagging system
cleft sentence	
clinical information system	
clinical information	

	closed class word	
	closed class	
	coherence relation	
	coherent text	
	cohesive link	
	collection of definitions	
CPMC	columbia-Presbyterian medical center	
	COMMIX system	
	common sense rule	
	common sense	
	common tag	
	communication channel	Definition: restricted to newswire input (JASPER does not read newspaper articles Definition: not restricted (to e.g. telex messages) , because full NL text , as found in newspaper reports etc is processed. SEE ALSO telex message, NL text, newspaper report, text, input, report, process, message
	communication of information	
	compact disc market	Definition: worth 345 million pounds sterling. SEE ALSO million pound, pounds sterling, million pounds sterling
	compact disc	
	comparison operation	
	complete set	
	complex construct	
	component part	
	comprehensive list	
	computational linguist	
	computational linguistics	
CAL	computer aided learning	
	computer game	
	computer printer	
	computer program	
	computer programming language	
	computer programming	
CS	computer science	
	computer scientist	
	computer system	
	computer	
	concept formation	
	concept fragment	

concept node

concept part

concept sort

concept structure

concept tag pattern

concept tag

concept — Definition: the hypernym (parent class) and the elaboration is a member of that class ; this relation is signalled by phrases such as include e.g. mammals include humans Definition: 3G language and the example of it is PASCAL. Examples: corporate takeover concept. SEE ALSO parent class, 3G language, language, relation, class, phrase, example

concepts class

conceptual dependency

conceptual relation

concession relation

conclaws pre-processing

concluding remark

CLAWS constituent likelihood automatic word tagging system

construction

content

CFG context free grammar — Definition: perhaps the grammar type which has most often been used to underpin parsers. SEE ALSO grammar type, parser, type

context problem

context — Definition: important when assessing the effectiveness of exemplification in instructional texts. SEE ALSO instructional text, text

contiguous group

continuous updating

corpus linguistics

corpus study

corpus

correct extraction

correct order

correct tag

creation

criterion example

curly moustache

	current context	
	current sentence	
	curriculum graph	
	CYC project	
	data element	Definition: not themselves altered , but their order of presentation is changed to assist the calling application. Parts: not themselves altered , but their order of presentation is changed to assist the calling application. SEE ALSO application, order
	data item	
DRA	data rearrangement algorithm	
	data rearrangement	
	data structure	
	daughter pattern	
	deep approach	
	deep method	Definition: difficult and time consuming to develop , and so it would seem that KE must also be a difficult goal. SEE ALSO KE
	deep NLP system	SEE ALSO natural language processing
	deep NLP	SEE ALSO natural language processing
	deep processing approach	Definition: considered with respect to the problems they entail. SEE ALSO problem
	deep processing DS	SEE ALSO domain specific
	deep processing	Parts: use of the full range of techniques and resources available to the traditional natural language processing (NLP) researcher. SEE ALSO natural language, language processing, natural language processing, language, processing, technique, NLP, researcher, process
	deep system	
	deep technique	Definition: the traditional methods from NLP and computational linguistics , and are aimed at language understanding. Examples: that of automatic parsing. SEE ALSO computational linguistics, computational linguist, language, linguistics, NLP, parsing, method, understanding
	default value	Definition: provided wherever possible and KEP echoes responses back to the user. SEE ALSO KE, response
DCG	definite clause grammar	

	definition relation	
	definition	Definition: given to tell the reader what is meant by a concept , examples seem to be given to aid in reaching an understanding of complex or subtle concepts , hypernyms place a concept into a tree -like categorisation scheme and hence allow the description of a new concept based upon differences from existing concepts , and partitions describe concepts as aggregates of components (which might already be familiar to the reader) Definition: spread over two sentences in a similar manner : The smallest readily accessible unit of memory storage is the byte. Parts: This problem is linked to the issue of whether historical facts. SEE ALSO historical fact, categorisation scheme, memory storage, part, categorisation, sentence, reader, issue, fact, description, understanding, problem, tree, example, link, concept, scheme, storage, unit
	degree of success	
	deliberate non-use	
	description	
	detailed description	
	detection	
	device within text	
	disc market	
	discipline of AI	SEE ALSO artificial intelligence
	discipline of linguistics	
	discourse analysis	
	discourse structure	
	discourse tree	
	dog	
	domain dependent	
	domain knowledge	
	domain pattern recognition	
	domain pattern	
	domain specific knowledge	
	domain specific NLP	SEE ALSO natural language processing
	domain specific system	
DS	domain specific	
	domain specificity	

	domain-specific knowledge	
	DS knowledge	SEE ALSO domain specific
	DS system	SEE ALSO domain specific
	early computer	
	element	
	ellipted material	
	elucidation fragment	
	elucidation part	
	elucidation tag pattern	
	elucidation tag	
	elucidation text fragment	
	elucidation text	
	embedded subordinate phrase	
	end se	
	end ss	
	end y	
	english language	
	enlarged heart	
	entire text	
	entire thought	
	episodic knowledge	
	error message	
	error rate	
	error triplet	
EC	European community	Parts: (1d) Germany.
	evaluation result	
	event expectation	
	event structure	
	examination marking	
	example categorisation	
	example glossary output	
	example glossary	
	example KEN output	SEE ALSO knowledge extraction, knowledge extraction network
	example KEN	SEE ALSO knowledge extraction, knowledge extraction network
	example of concepts	
	example of KE	SEE ALSO knowledge extraction
	example of SRs	SEE ALSO sort routine
	example of text	
	example pattern	

example term summary	
example term	
example	Definition: part -whole descriptions and class inclusion statements are clearly useful pieces of knowledge which help a reader to understand a new concept Definition: a valuable tool in instructional text and their roles in pedagogical applications have been much studied. Parts: recent year , old friend , serious error. SEE ALSO pedagogical application, class inclusion, instructional text, text, knowledge, class, part, application, reader, description, concept
exception list	
exception phrase	
exception word	
exemplification relation	Definition: similar to the instance relation. SEE ALSO instance relation, relation
exemplification template	
exemplification token	
existing LTM	SEE ALSO long-term memory
expansion	
expectation	
expert system	
explanatory text	Definition: on the other hand precisely those texts designed to convey knowledge to the reader. SEE ALSO text, knowledge, reader
expository text	
extended timespan	
external file	
extra letter	
extracted fact	
extracted part	
extraction candidate	
extraction method	
extraction of facts	
extraction performance	
extraction process	
extraction program	
extraction result	
extraction stage	
extraction system	

	extraction task	
	extraction	
	fact density	
	fact extraction system	
	fact extraction	
	fact	Definition: simply as a true statement about the universe or its contents. SEE ALSO true statement, content
	fact-Poor text	
	factor for domains	
	factor	
	factual knowledge	
	FASTUS system	SEE ALSO United states
	fat jolly man	
	feature	
	fictional text	
	field of NLP	SEE ALSO natural language processing
	file name	
	file	Definition: usually very large (more than twice the size of the input text). SEE ALSO input text, text, input, size
	filter phrase	
	final output	
	financial institution	
FSA	finite state automaton	
	finite state	
	fleeting event	
	form	
	formal definition	
	format	
	fragment of sentences	
	fragment of text	
	fragment validation	
	fragment	Definition: validated to confirm that the pattern match was a useful one. SEE ALSO pattern match, pattern, match
	free text	
	full description	
	full evaluation	
	full list	
	full parse	

	full parsing	
	full stop	
	full syntactic/semantic parsing	
	full text understanding	
	full text	
	function	
	functional part	
	future enhancement	
	future sentence	
	game	
	games industry	
	general knowledge	
	general text	Definition: whose target readership is not the closed class of a specialist group. SEE ALSO closed class, class, reader, group
	generative grammar	
	generic fact	Definition: more knowledge -like than facts about specific single objects. SEE ALSO knowledge, fact, object
	geographical information system	
GIS	geographical information systems	Definition: gIS).
	geographical information	
	given relation	
	glossary creation	
	glossary entry	
	glossary maker	
	glossary output	Definition: also produced from the spinal LL but in a reformatted form that orders glossary entries alphabetically on the first column present (acronym or term). SEE ALSO glossary, term, form, order
	glossary	Parts: lists of terms present in the text , together with explanations of those terms. SEE ALSO text, term, list
	good example	
	good excuse	
	good extraction	
	good tokenisation	
	grammar book	
	grammar type	
	grammatical construction	
	grammatical feature	

	grammatical function	
	grammatical phrase	
	grammatical sentence	
	graphical output	
	group of words	
	group	
	handwriting recognition	
	hardware associated software	
	hardware sale	
	head noun	
HMM	hidden markov model	
	high success rate	
	high success	
	high-energy radiation	
	high-level language	
	highest level	
	historical fact	
	historical narrative	
	historical report	Definition: not valid input texts. SEE ALSO input text, text, input
	historical text	Definition: which are descriptions of chains of events Definition: not generally of interest to KE systems. SEE ALSO KE system, system, KE, description
	human behaviour	
	human intelligence	
	human intervention	
	human knowledge	
	human language	
	human reader	Definition: able to perform KE almost effortlessly , but the term KE is used in this thesis to refer to KE by computer program. SEE ALSO computer program, program, term, KE, computer
	hypelab/Hypertutor system	
	hypernym relation	Definition: the other facet of the hyponym relation. SEE ALSO hyponym relation, relation
	hyponym relation	Definition: also called class inclusion. SEE ALSO class inclusion, class
	hyponymic relation	
	identical concept	

	important factor	
	important topic	
	incident merging	
	incident monitoring	
	indefinite article	
	index creation	
	indirect anaphora	
	individual fact	
	individual sentence	
	individual word	
	industrial complex	
	information content	
	information extraction task	
IE	information extraction	
	information from text	
IR	information retrieval	
	information system	
	information	Definition: distinguished from knowledge in that it is intended to be used within a short time after its reception Definition: conveyed for a specific purpose. SEE ALSO specific purpose, knowledge, purpose
	initial indefinite article	Definition: always stripped off. Parts: always stripped off.
	ink cartridge	
	input definition	Type of: taken and attempts are made to match patterns with it. SEE ALSO pattern, match
	input for KE	SEE ALSO knowledge extraction
	input list	
	input output	
	input sentence	
	input text	Definition: also assumed to be free from spelling mistakes and grammatically correct.
	input	Definition: is taken and attempts are made to match patterns with it Definition: short machine readable report derived from dictated comments which includes some fixed fields and some free NL fields Definition: top -level goal such as describe list. SEE ALSO pattern, list, report, match
	instance of apposition	
	instance relation	

instruction manual

instructional text

intelligence — Definition: , but it is relatively easy to identify a system which is apparently intelligent within its application domain. Type of: property possessed by humans. SEE ALSO system, application

intelligent recognition system

IRSG intelligent recognition systems group

intelligent recognition

intended application

intended meaning

intended readership

interbank money transfer

interbank money

interested reader

internal storage

IUCN international union for conservation of nature

introduction to knowledge

introductory example — Definition: aimed at novices who want to learn about the concept. SEE ALSO concept

IR system — SEE ALSO information retrieval

issue

item

jolly man

JASPER journalists assistant for preparing earnings reports Definition: DS system aimed at automating a specific task for a commercial organisation Definition: potentially NDS Definition: a good example of a very successful , fast , shallow IE/KE system , based on pattern matching. Examples: very successful , fast , shallow IE/KE system , based on pattern matching. Parts:) The frame with its slots , slot patterns and slot processing methods make up the DS. SEE ALSO pattern matching, DS system, pattern match, good example, processing, task, system, pattern, method, process, matching, NDS, match, example

just name

KE application — SEE ALSO knowledge extraction

KE approach — SEE ALSO knowledge extraction

KE input — SEE ALSO knowledge extraction

KE program — SEE ALSO knowledge extraction

	KE system	SEE ALSO knowledge extraction
	KE task	SEE ALSO knowledge extraction
	KE	SEE ALSO knowledge extraction
	KE-Relevant grammatical feature	SEE ALSO knowledge extraction
	KEN output	SEE ALSO knowledge extraction, knowledge extraction network
	KEP glossary	SEE ALSO knowledge extraction, knowledge extraction program
	KEP output	SEE ALSO knowledge extraction, knowledge extraction program
	KEP preprocessor program	SEE ALSO knowledge extraction, knowledge extraction program
	KEP preprocessor	SEE ALSO knowledge extraction, knowledge extraction program
	KEP program	SEE ALSO knowledge extraction, knowledge extraction program
	KEP system	SEE ALSO knowledge extraction, knowledge extraction program
	KEP tokenisation	SEE ALSO knowledge extraction, knowledge extraction program
	KEP user query	SEE ALSO knowledge extraction, knowledge extraction program
	KEP user	SEE ALSO knowledge extraction, knowledge extraction program
	KEPs pattern	SEE ALSO knowledge extraction, knowledge extraction program
	key phrase	
	key word	
	knowledge acquisition system	
	knowledge acquisition	Type of: This phase is possible only after the language learning phase has reached a certain maturity , but it is difficult and possibly erroneous to separate these two. SEE ALSO language
KB	knowledge base	
	knowledge basis	
	knowledge categorisation	
	knowledge engineering	
KEN	knowledge extraction network	
KEP	knowledge extraction program	Definition: working program which demonstrates the usefulness of shallow , NDS methods , and which has opened

		up the possibilities of several new research directions , including automatic index creation , student assignment marking , and information retrieval from the Internet for the automatic construction of semantic -net knowledge bases Definition: not designed to extract procedural knowledge Definition: designed only to consider the first of these levels Definition: designed to process texts from various sources Definition: started with the minimum of keystrokes Definition: unconcerned with such matters Definition: currently not capable of finding mixed -case shortenings (such as DfE for Department for the Environment) Definition: more easily able to remain domain dependent Definition: designed to process explanatory texts. Parts: novel function developed specially to do this (the sing ( ) function). SEE ALSO explanatory text, research direction, automatic index, index creation, student assignment, information retrieval, automatic construction, automatic index creation, knowledge base, procedural knowledge, domain dependent, novel function, text, program, knowledge, creation, construction, method, information, process, function, level, marking, NDS
KE	knowledge extraction	Definition: the automated extraction of facts from machine -readable text Definition: branch of Natural Language Processing (NLP) Definition: the process of obtaining knowledge from text Definition: exciting and challenging new discipline. SEE ALSO extraction of facts, extraction, text, knowledge, NLP, fact, process
	knowledge representation	
	knowledge type	
	knowledge	Definition: regarded as a collection of true -for -all -time facts Definition: for specific -object facts tend to look more like information and generic facts more like knowledge. Type of: This thesis is not about such. SEE ALSO generic fact, fact, information
LOB	lancaster oslo/Bergen corpus	
	language acquisition	Parts: language variation (geographical) and semantics. SEE ALSO language
	language processing	
	language	Type of: The central idea in

	Chomskys work has been that of innateness , the ability of humans to learn certain.
large amount	
large brown dog	
large curly moustache	
large number	
laser printer	
LCS function	SEE ALSO computer science, longest common substring
leftmost ruleid	
letter	
level of sugar	
level	
lexical ambiguity	
lexical disambiguation	
lexical form	
lexical item	
lexical pattern	
lexico-syntactic pattern	
light	Definition: although of relatively low energy by comparison.
likely tag	
line of input	
line of text	
line structure	
linear progression	
linguistic complexity	
linguistic issue	Examples: problems caused by anaphoric links between sentences. SEE ALSO anaphoric link, sentence, problem, link
linguistic knowledge	
linguistics	
link between sentences	
link type	
link	
linked list	
list of acronyms	
list of exemplification	
list of items	
list of tag	
list of tags	
list of templates	

	list of token/phrase	
	list of tokenisations	
	list part	
	list	Examples: By concept they mean thing. SEE ALSO concept
	LL data structure	
	LL data	
	LOB file	SEE ALSO lancaster oslo/Bergen corpus
	local context	
	long leg	
	long output file	
	long output	Definition: rarely printed. Parts: same extraction data plus all processing comments and error messages (if any) as well as line and sentence structures. SEE ALSO sentence structure, error message, extraction, processing, sentence, structure, process, message
LTM	long-term memory	Definition: as a semantic net comprising only hyponym/hypernym information. SEE ALSO semantic net, information
LCS	longest common substring	
LDOCE	longman dictionary of contemporary english	
MRD	machine readable dictionary	
MT	machine translation	
	machine-readable text	
	main purpose	
	manual	
	mark template	
	market	
	marking system	
	marking	
	mass noun	
	match	
	matching approach	
	matching technique	
	matching	
	material relation	
	meaning	
MED	medical entities dictionary	
MEDLEE	MEDical language extraction and encoding	Definition: DS system. SEE ALSO DS system, system, medical entities dictionary

	medical practitioner	
	MEDLEE system	SEE ALSO MEDical language extraction and encoding, medical entities dictionary
	memory requirement	
	memory storage	
	merging function	
	message clarifier	
	message content	
MU	message understanding	
	message	Definition: also displayed as to the current stage of processing attained. SEE ALSO stage of processing, processing, process, stage
	method	
	middle column	Parts: 3rd -column text mentions GIS , or if GIS. SEE ALSO text
	million dollar	
	million pound	
	million pounds sterling	
	modern linguistics	
	money transfer	
	morphological form	
	motivating factor	
	mr. smith	Definition: in apposition to fat jolly man with a large curly moustache. SEE ALSO jolly man, curly moustache, fat jolly man, large curly moustache
	MRD analyser	SEE ALSO machine readable dictionary
	multi-word noun phrase	
	multi-word noun	
	multi-word phrase	
	multiple pattern	
AN	n is a noun	
	name source	
	name	
NLG	natural language generation	
NLI	natural language interfaces to computers	
NLP	natural language processing	
	natural language	
	natural unit	
	nature of knowledge	

	nature of language	
	NDS approach	SEE ALSO non domain specific
	NDS KE system	SEE ALSO knowledge extraction, non domain specific
	NDS KE	SEE ALSO knowledge extraction, non domain specific
	NDS system	SEE ALSO non domain specific
	NDS	SEE ALSO non domain specific
	need for WK	SEE ALSO world knowledge
	negative example	Definition: counter -examples and as such are deliberate non -instances of the concept being described ; they play a contrastive role. SEE ALSO concept
	negative trigger phrase	
	negative trigger	Parts: But note that if a positive trigger is found in a sentence , and this positive trigger is subsequently found to be. SEE ALSO positive trigger, sentence, trigger
	net KB	SEE ALSO knowledge base
	news story	
	newspaper report	
	newswire story	
	NL text	
	NLP application	SEE ALSO natural language processing
	NLP community	SEE ALSO natural language processing
	NLP practitioner	SEE ALSO natural language processing
	NLP program	SEE ALSO natural language processing
	NLP researcher	SEE ALSO natural language processing
	NLP system	SEE ALSO natural language processing
	NLP task	SEE ALSO natural language processing
	NLP	SEE ALSO natural language processing
NDS	non domain specific	
	non-terminal symbol	
	nonrelevant phrase	
	nottingham trent university	
	nottingham trent	
	noun group	
	noun phrase	

NP	noun phrases	
	noun	
	novel acronym extractor	
	novel acronym	
	novel function	
	novel KE	SEE ALSO knowledge extraction
	number of rules	
	number of tokenisations	
	number	
	object	
OO	object-oriented	
	open class word	
	open class	
	order of processing	
	order	
	original BNC file	SEE ALSO british national corpus
	original BNC	SEE ALSO british national corpus
	original text	
	orthographic word	
	output file	
	output format	
	output list	
	output phrase	
	output	Definition: into a structure designed to hold clinically salient information , based on information formats of the Linguistic String Project (Sager , Friedman and Lyman (1987)) Definition: plan of the tutorial required. SEE ALSO structure, form, information, format
	page layout	
	paper cardie	
	paragraph of text	
	parent class	
	park border	
	parse tree	
	parser	Definition: designed to handle fragments of sentences , and not just whole grammatical sentences Definition: interactive in that it will prompt the user for the syntactical usage of the verb when it comes across an unknown verb. SEE ALSO

	grammatical sentence, sentence, fragment
parsing	
part of speech	
part	Definition: , however , homeomerous (the Everglades are Florida , as is Florida). Parts: , however, homeomerous (the Everglades are Florida and as is Florida).
part-of-speech tag	
part-of-speech tagger	
part-of-speech tagging	
part-whole description	
partial parser	
partial parsing	
partition relation	Definition: signalled by keyphrases such as is made up of three parts , comprises , has the following components etc. SEE ALSO part
passive voice	
patch corpus	
patch rule template	Definition: built -in.
patch rule	
pattern file	
pattern from justeson	
pattern match	
pattern matcher	Definition: sophisticated enough to recognise different morphological forms for nouns and verbs and so removes the need for the person specifying the patterns to list all possible forms. SEE ALSO morphological form, pattern, list, form, noun
pattern matching approach	Definition: used (fifth point) which relies on part -of -speech tagging (sixth point). SEE ALSO part, tag
pattern matching technique	
pattern matching	Definition: then performed on each of these sentences. SEE ALSO sentence
pattern recognition stage	Definition: left to make the decision. Parts: left to make the decision.
pattern recognition	Definition: far more accurate and efficient than parsing. SEE ALSO parsing
pattern variable	
pattern	
pattern-matching approach	

pattern-matching technique	
pedagogical application	
performance of KEP	SEE ALSO knowledge extraction, knowledge extraction program
personal computer	
philosophy department	
phrasal analysis	
phrase analysis	
phrase part	
phrase recognition	
phrase within sentences	
phrase	Definition: recognised using syntactic information and domain knowledge , and patterns of those phrases are then looked for Definition: was matched to the token = , the phrase PASCAL was matched to the token X , and the phrase. Parts: punctuation marks are and allowed. SEE ALSO punctuation mark, domain knowledge, syntactic information, knowledge, pattern, token, information, match
phrases example	
physical object	
piece of information	
piece of knowledge	
piece of text	
plain text	Definition: first tagged and then passed to KEP. SEE ALSO KE, tag
plural form	
plural noun	
point size	
positive integer	
positive trigger phrase	
positive trigger	
possible meaning	
possible tag	
potential term	
potential TTs	SEE ALSO technical terms
pounds sterling	
practical benefit	
practical consideration	Definition: not the sole factors here. SEE ALSO factor, fact
practical ground	

practical problem	
practical reason	
practitioner	
pragmatic knowledge	
pragmatic processing	
pre-processor program	
pre-word tag	
preceding word	
precision figure	
preference semantics	
preprocessor program	
presentational sentence	Definition: then rejected.
prime aim	
printed document	
printer	
problem size	
problem	Definition: that the KB needs to be huge.
procedural KE	SEE ALSO knowledge extraction
procedural knowledge	Definition: less likely to be present within a single sentence. SEE ALSO single sentence, sentence
procedural text	
process	Definition: chains of events.
processing approach	
processing category	
processing DS system	SEE ALSO domain specific
processing DS	SEE ALSO domain specific
processing NDS system	SEE ALSO non domain specific
processing NDS	SEE ALSO non domain specific
processing stage	
processing step	
processing system	
processing time	
processing	Parts: form suitable for input to the next stage, parser, also domain specific (the domain being that of medical examinations), message interpreter and to find all acronyms in the input text , and if possible , what they stand for (called by the author the acronyms expansion). SEE ALSO input text, acronyms expansion, text, parser, input, form, stage, message, expansion,

domain specific

production rule

program name source

program name

program run

program — Definition: described here which really lies in the content analysis or text summarisation fields
Definition: domain specific
Definition: still running and that all is well. SEE ALSO text summarisation, text, content, analysis, run, domain specific

programming language

proper noun

punctuation character

punctuation mark

purpose wheres

purpose

quick sort

RIMNET radiation incident monitoring NETwork

radiation with wavelengths

radiation

rate

readable dictionary

reader

real term

rearrangement algorithm

recognised phrase — Definition: passed onto the pattern recognition stage , which processes them in the order they occur. SEE ALSO pattern recognition, recognition stage, pattern recognition stage, pattern, process, recognition, order, stage

recognition stage

recognition system

recognition systems group

recognition

regular expression

related term

relation definition

relation detection

relation extraction

relation of interest

	relation present	
	relation reference	
	relation type	
	relation	Definition: used as the functional building blocks. Type of: elaboration , evidence , justification , summary , volitional cause , and background Type of: The KEP program represents an attempt to rectify this omission , albeit by looking for specific. SEE ALSO KEP program, volitional cause, program, summary, KE, function
	relative clause	
	report	
	required response purpose	
	required response	
	research area	
	research direction	
	research field	
	researcher	
	reserve of WK	SEE ALSO world knowledge
	response purpose wheres	
	response purpose	
	response	
	restrictive relative clause	
	result	
	reuters story	
	review paper cardie	
	review paper	
	rhetorical relation	
RST	rhetorical structure theory	
	rhetorical structure	
	river bank	
	role filler	
	role-filler expectation	
	routine definition	
	routine	
	RST relation definition	SEE ALSO rhetorical structure theory
	RST relation	SEE ALSO rhetorical structure theory
	RST structure	SEE ALSO rhetorical structure theory
	rule template	

rule	Definition: also quite useful since it can find abbreviations such as RIMNET if their expansions are like Radiation Incident Monitoring NETwork. Type of: e. Examples: n -10. SEE ALSO expansion
run of KEP	SEE ALSO knowledge extraction, knowledge extraction program
run on slot	
run	
running text	
s definition	Definition: which are conceptually very similar. SEE ALSO concept
sale	
sample input	
sample list	
scheme	
SCISOR system	
script version	
search engine	
section heading	
section of text	
section	
selected story	Definition: matched against a frame of slots , the slots defining both the fact to be searched for and the method of processing to achieve this. Parts: matched against a frame of slots , the slots defining both the fact to be searched for and the method of processing to achieve this. SEE ALSO processing, method, fact, process, match
selection stage	
semantic content	Definition: in contradiction to the pragmatic knowledge that (unless you have painted them etc) no bananas are this colour. SEE ALSO pragmatic knowledge, knowledge
semantic head	Definition: cardiomegaly.
semantic knowledge	
semantic level	
semantic net KB	SEE ALSO knowledge base
semantic net	
semantic parser	
semantic part	
semantic processing	

semantic relatedness	
semantic relation	
sense rule	
sentence array	Definition: chosen which makes the processing easier or faster. SEE ALSO processing, process
sentence boundary exception	
sentence boundary	
sentence by sentence	
sentence fragment	Definition: used in the relation extraction , as described in the following subsection. SEE ALSO relation extraction, extraction, relation
sentence number	Definition: used widely in screen and file output. SEE ALSO output, file
sentence structure	
sentence tokenisation	
sentence	Definition: still ambiguous at the semantic level Definition: hyponymy Definition: the basic unit from which texts are built Definition: spread across any number of lines (including the case where more than one sentence can be present on a single line of input) and so KEP attempts to split the input into sentences to fill the sentence storage array Definition: probably presentational Definition: talking about an exemplification given in previous text rather than in the current sentence Definition: reduced to a string of single characters by the replacement of words , groups of words , and punctuation by token characters. SEE ALSO semantic level, basic unit, line of input, current sentence, text, input, token, KE, word, number, level, group, storage, array, unit, string
sentence-by-sentence basis	
sentence-end detection	
sentence-end detector	
separate extraction	
separate function	
separate manual	
separate sentence	
set of patch	
set of production	

set of questions	
set of relations	
set of tokenisations	
set	Definition: not the same thing as a collection ; sets have names which reflect their membership Definition: essentially a collection of individual items , not a whole thing , such as a forest. SEE ALSO item, name
shallow approach	
shallow method	
shallow NDS system	SEE ALSO non domain specific
shallow NDS	SEE ALSO non domain specific
shallow pattern-matching approach	
shallow processing approach	Definition: the only practical route given the timescales involved in this research.
shallow processing category	
shallow processing	
shallow system	
shallow technique	Definition: as methods which achieve NLP goals without recourse to attempts to understand fully the input text Definition: used wherever feasible. SEE ALSO input text, text, NLP, input, method
shallow way	
short output	
simple text	
simpsons arm	
single phrase	
single sentence	
single word term	
single word	
singleword term	
singular form	
singular noun	
size	
skuce et	
small amount	
small set	
small south american	
small south	
social connection	

	software sale	
	sort algorithm	
	sort criterion example	
	sort criterion	Examples: alphabetical order and alphabetical order. SEE ALSO alphabetical order, order
	sort routine definition	
SR	sort routine	Definition: a function which orders a list of items according to some criterion Definition: composed of four elements : input list , output list , sort criterion and sort algorithm. Type of: the bubble sort and the quick sort Type of: data rearrangement algorithm , or DRA Type of: the bubble sort and the quick sort. Parts: input list, output list, sort criterion and sort algorithm. SEE ALSO list of items, bubble sort, quick sort, input list, output list, sort criterion, sort algorithm, data rearrangement, rearrangement algorithm, data rearrangement algorithm, input, output, list, item, function, order, element, sort, algorithm
	sort	
	source of knowledge	
	source text	
	south american rodent	
	south american	
	sparck jones	
	specialist term	
	specific application	
	specific instance	
	specific item	
	specific KE	SEE ALSO knowledge extraction
	specific knowledge	
	specific NLP	SEE ALSO natural language processing
	specific phrase	
	specific purpose	
	specific section	
	specific set	
	specific syntax	
	specific system	
	specific technique	
	specific type	
	speech code	Parts: part of speech tagger

, usually referred to simply as a tagger , is a program which accepts a text and returns that text with each word tagged with a. SEE ALSO part of speech, speech tagger, speech tag, text, program, part, word, tagger, tag

speech information

speech tag

speech tagger

spinal ll data

spinal ll

stage of processing

stage

standard phrase

starting point

state automaton

stepwise refinement

stochastic tagger

stop in reference

stop in title

storage capacity

storage

story

string

structure array

structure for text

structure of text

structure theory

structure

student assignment

student response

subcategorisation information

subject domain

subject matter

subordinate phrase

subsequent processing

success rate

successful extraction

successful KE — SEE ALSO knowledge extraction

summaries output

summary

surface structure

	symbol	
	syntactic information	
	syntactic knowledge	
	syntactic parser	
	syntactic processing	
	syntactic/semantic parsing	
	syntactical information	
	syntax	
	system	
	systems group	
	tag error triplet	
	tag error	
	tag pattern	
	tag	
	tagger	Definition: not a parser Definition: trained by scanning a large correctly tagged corpus. SEE ALSO parser, tag, corpus
	tagging format	
	tagging process	
	tagging scheme	
	target concept	
	target fact	
	task	Definition: to check that the presented sentences are indeed legal. SEE ALSO sentence
TEFL	teaching english as a foreign language	
	technical report	
	technical term acquisition	
	technical term	Definition: etc in the input text , and so KEP needs to be able to identify them Definition: usually domain dependent Definition: almost always multi -word noun phrases , which consist of adjectives and nouns and sometimes prepositions , but very rarely verbs , adverbs or conjunctions. Parts: Such texts also often make use of abbreviations , which play the. SEE ALSO input text, noun phrase, domain dependent, text, phrase, input, KE, noun, noun phrases
TT	technical terms	Definition: present in legal documents , medical texts , technical reports , scientific papers , trade journals , professional newspapers etc. SEE ALSO technical report, text, report

	technical text	
	technique	
	television set	
	telex message	
	template file	
	template matching	
	template pattern	
	template token	
	template	Definition: similar in form to the tokenisations described above except that (1) they always end with a sentence -terminating punctuation mark , and (2) instead of containing X -tokens they contain the token C and the tokens 0 , 1 , 2. SEE ALSO punctuation mark, sentence, token, tokenisation, form
	term acquisition	
	term basis	
	term pattern	
	term summaries output	
	term summary	Definition: not held in the spinal LL data structure and are constructed after TTs and acronyms have been collected. SEE ALSO data structure, LL data, LL data structure, structure
	term	Definition: actually used in two different ways as follows Definition: partly has the intended meaning is partly made of. SEE ALSO intended meaning, part, way, meaning
	terminal symbol	
	terminating punctuation mark	
	terminating punctuation	
	terminology extraction	
	test sentence	
	test text	Definition: given as Figure 11.
	text analyser	
	text block	
TCS	text categorization shell	
	text fragment	Definition: , partition , hypernym). SEE ALSO part
	text in figure	
	text into sentences	
	text processing	
	text required response	

	text stream	
	text structure	
	text summarisation	
	text to text	
	text understanding	
	text	Definition: originally fragmented prior to matching against the 3 -head bare templates using an extensive list of key words to indicate fragmentation points Definition: scanned for apparent events (e.g. rumour of a takeover) , role -fillers obtained where obvious , and event expectations set up whenever possible Definition: not just a set of standalone sentences (see Halliday and Hasan (1976)) Definition: not arbitrary collections of unrelated sentences ; they are coherent Definition: chunked into sentences for the purposes of this scanning. Examples: verbless phrases , due to the production process and actor or recipient). Parts: fact -bearin, Many sentences exist to smooth the flow of reading or point the reader to other, roles played by and wit system of Reimer (1989) required a small amount of domain knowledge to focus its attention on relevant. SEE ALSO key word, bare template, event expectation, domain knowledge, wit system, small amount, amount of domain, system, knowledge, set, phrase, sentence, list, template, reader, purpose, fact, word, amount, process, fragment, matching, expectation, match
	textual element	
	textual form	
	theoretical interest	
TLA	three-letter acronym	Definition: feature of modern technological life. Type of: three -letter acronym. SEE ALSO feature
	TIE routine	SEE ALSO information extraction
	time context problem	
	time context	
	time saving	
	token file	
	token phrase	
	token	Definition: single -characters which are used to stand in for the phrase. SEE ALSO phrase

token/phrase pair	Examples: given in Table 8.
tokenisation method	
tokenisation process	
tokenisation string	
tokenisation	
tokenisation-template match	
too-long sentence	
toy market	
training corpus	
transition network	
tree	Parts: absent , because they were never produced.
trent university	
trigger list	
trigger phrase	
trigger	
trivial task	
trivial tokenisation	
true statement	
true tag	
true-for-all-time fact	
TRUMP output	
txt definition	
txt exemplification	
txt hyponym	
txt partition	
type of data	
type of fact	
type of information	
type of knowledge	
type of lemur	
type of mammal	
type of partition	
type of relation	
type of sentence	
type of text	
type	
ultra-violet radiation	
unconfirmed term	
understanding process	Parts: both semantic and pragmatic aspects.

	understanding	
	unit	
UK	United kingdom	
USA	United states of america	Examples: rarely expanded in text , because probably all adult English -speaking readers know what it stands for). SEE ALSO text, reader
US	United states	
	universal widget	
	unknown word	
	unseen text	
	untagged sentence	
	useful fact	
	useful feature	
	useful information	
	user query	
	UV light	
	valid concept	Type of: If it is , then it is likely that it.
	valid filename	
	validated candidate	
	vander linden	
	verb group	
	vertical-format LOB file	SEE ALSO lancaster oslo/Bergen corpus
	vertical-format LOB	SEE ALSO lancaster oslo/Bergen corpus
	video game	
	video games industry	Definition: growing fast and will dominate the toy market and become an established part of home entertainment. SEE ALSO toy market, part, market
	visible light	
	volitional cause	
	way	
	well-formed sentence	
	whole object	
	whole sentence	
	whole text	
	wit parser	
	wit system	Definition: one example of a KE as opposed to an IE system. SEE ALSO system, KE, example
	word co-occurrence	
WP	word processor	

	word string	
	word term	
	word	
	word-initial letter	
	work of fiction	
	working system	
WK	world knowledge	Definition: deemed to be essential for good NLP programs. SEE ALSO NLP program, program, NLP
WWW	world wide web	
	WP feature	SEE ALSO word processor
	WP package	SEE ALSO word processor

[1090 GLOSSARY ENTRIES]

END OF GLOSSARY **********