A Natural Language Processing Framework for Automated Assessment

Nicholas Mark Allott

PhD Thesis

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

The Nottingham Trent University
Department of Computing

May 2000

Nicholas Mark Allott

A Natural Language Processing Framework for Automated Assessment

PhD Thesis

Abstract

A novel knowledge representation schema for application in natural language processing is presented along with the algorithms for its automatic population from text corpora. The natural language understanding task under consideration is the task of the automated assessment of students' single sentence responses to technical questions. This knowledge representation schema shares much in common with a localist connectionist network and consequently possesses both representational and computational properties. The task of automated assessment is used as an empirical framework within which the performance of the knowledge representation schema can be evaluated. An initial experiment tested the ability of hand crafted knowledge structures to effectively encapsulate a correctness decision procedure. A correlation of 85% with the performance of an independent human marker was achieved. A second experiment tested the knowledge schema's ability to generalise to new unseen data; this correlated 65% with human performance. To address the issue of knowledge base creation two algorithms are presented which produced networks consisting of composite and clustering node-types. These activation-passing networks provide a perceptual function generating higher order descriptions of the input data. As a control, a system using Latent Semantic Analysis as an automated decision procedure was input with raw student sentences to mark. A correlation of 37% with human marking was achieved. By feeding the same LSA procedure with perceptually augmented input, derived from algorithmically produced activation passing networks, correlations of 55% were achieved. For this application the perceptual enhancement provided by these networks produces a 48% improvement in correlation scores, demonstrating the network's empirical utility in the automated assessment domain.

Acknowledgements

I must first thank my wife and children, not only for their patience in putting up with me whilst this research was underway, but for their love and support, which is cherished by me above all else. Also I must thank my parents for their encouragement and belief in me, as well as the constant help they give to us as a family. And finally thanks must go to my supervisors, Pete Halstead, Lindsay Evett and Pauline Fazackerley, without whose guidance and assistance this work would not have been possible.

Table of Contents

APPENDICES

Table of Figures

1. Introduction

1.1. The Nature of Natural Language

Natural language is complex, surprisingly so. Many people, computer technicians and lay people alike have underestimated the difficulties involved in producing computer based systems capable of understanding natural language, and have been amazed how slow the technologies are in coming. There have been many techniques explored, some meeting with limited success, others not even achieving that. There has as yet been no comprehensive solution proposed or implemented.

This thesis is not, nor does it pretend to be, that solution. What this thesis does do is explore some of the problems of natural language from the perspective of a particular application; that of automated assessment. This application, it is proposed, offers an ideal empirical framework in which to study language phenomena, particularly those to do with meaning. The core technology employed, in the solution implemented, is a connectionist representation schema, which contrasts well with the logic and frame based representational schemas that are more common within the natural language understanding domain. This schema brings the advantages of simplicity and inherent processing capacity and also opens up avenues for exploring novel learning algorithms.

The problem of meaning is a difficult problem to solve, or even define. The problem is this: we wish to extract the meaning from text, but what form should this meaning take? Clearly the meaning cannot be the text itself, for not only is it possible to mean the same thing using two different pieces of text, but it is also possible to mean two different things with the same fragment of text, given different contexts. We can hypothesise that meaning itself is some complex, dynamic, context sensitive informational structure lying above the raw text level. But at this moment we have neither access to, nor knowledge of this structure. It seems that language itself is the only way we can measure, define and relate this concept of meaning.

If this is true there are two practical ways we can formally explore the nature of this meaning:

1. To examine or attempt to generate the logical consequences of a particular piece of text. These consequents are, or must constitute, a large component of the meaning of this text. These consequents must be expressed (or generated) in an analysable textual form.

2. To examine many different pieces of text which supposedly mean the same thing. The invariant aspects of those sentences that do have identical meaning must shed light on the nature or structure of the meaning of these sentences.

An extensive search of the literature reveals no application that attempts to generate the full set of consequents that are possible from a piece of text. However, in a limited sense, message understanding and information retrieval tasks are, indeed, performing this type of task. In such applications a set of consequent or consequent types is predetermined. The retrieval process searches through this text looking for the matches or antecedents which would imply the consequent. Meaning is therefore being investigated by examining implication.

The task of automated assessment, on the other hand, more closely resembles the second of these techniques. All correct answers generated do indeed have a similar meaning. Further, this synonymity, or equivalence of meaning, can be rigorously defined by examining the mark awarded to it. The construction of a knowledge schema which models marker performance is essentially a process of reverse engineering. That is, a reconstruction of the decision making process by emulating observed behaviour. There is a caveat; there is no absolute guarantee that the reverse engineered knowledge structure will at all correspond to *true* meaning, in other words how it is mapped within the human brain. However this is a fundamental problem that is true for all artificial cognitive models. As with all such cognitive models, by virtue of the fact that the artificial construct exhibits (or will hopefully exhibit) reasonable performance, it gives an indication of the absolute complexity of the problem, regardless of the true human implementation.

1.2. Motivations for and Applications of Natural Language Processing

Computers are pervasive in modern society. Most jobs entail the use of computers in one form or another. The interface to computers (basically keyboard and monitor) is not only bulky and expensive but requires specific skills to use. Natural Language interfaces come in two forms: those that understand the spoken word and those that understand the written word. A written word interface cuts down on the specific skills required to use a computer; a spoken word interface does this, but also makes both keyboard and monitor redundant. The pressure to develop systems that would allow a more natural interface to computers, one that does not require specific training, is therefore very great.

Although speech and text recognition require completely different technologies to capture their input (sound processing techniques vs image processing techniques) they share

many features of higher level processing. Orthodox linguistic theory would portray the processing strategy as shown in Figure 1-1.

Both spoken and written raw input is inherently non discrete and noisy. There is an isomorphism between these data streams that can lead us to hypothesise an idealised string which represents the abstracted properties of language. The phrase "the cat sat on the mat" can be spoken or written down (as it is here) but no two instances will be precisely the same.

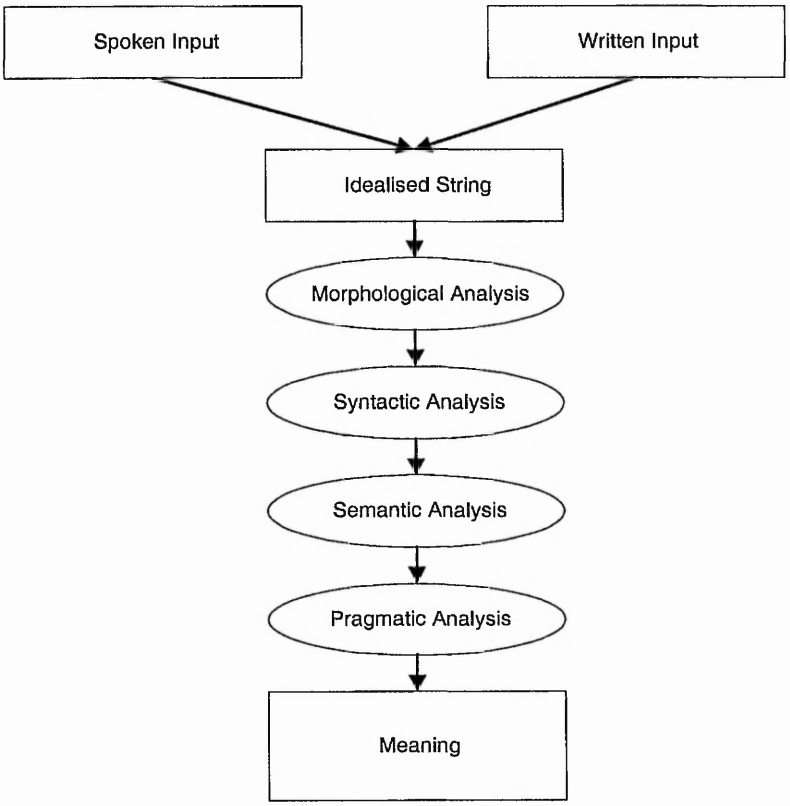


Figure 1-1 Typical Processing Strategy

The language theories represented above are all theories that describe the composition of these idealised strings at different granularities and with their own symbol set. In other words they constitute a set of observed regularities within a particular symbolic framework. Syntax, for example, describes the composition of a sentence in terms of syntactic categories; componential semantics describes a sentence in terms of its primitive semantic categories.

In performing recognition we are attempting to identify the correct idealised string to which the raw instance of spoken or written input maps. Language models can help in this task. Typically a selection of possible strings are generated by the recognition engine and language theories are applied to discount or affect the probability estimates of the various possibilities. This is one application of NLP technologies: text and speech recognition.

Another application of the technology is the divination of meaning. This is a task which, if the above diagram is to be believed, implies the understanding and application of all the subsumed language theories. It is a task, therefore, which although entirely distinct, may share many of the computational dependencies with recognition.

To emphasise the distinction between language understanding and language recognition consider the use of a word processor. Probably one of the most productive initial applications of NLP technology, and hence most researched within corporate institutions, would be the generation of text documents from dictation. This task of speech recognition is the task of mapping noisy indistinct acoustic data to idealised strings. Contrast this with the ability to say, *"Computer: please save my document"*, and have the computer perform the intended action. This implies understanding by the computer. It would require for example a knowledge of the various potential meanings of the word save and a mechanism for its resolution. It would also require an ability for the computer to identify the intended referent of a word from its internal set of observed (known about) entities and be able to relate these entities to the set of actions of which the computer is capable. (*My document* is the document being dictated into, *save* is an action of which the computer is capable).

There are two primary applications of natural language understanding technologies. Firstly, the translation of a natural string to an appropriate action by the computer, as above. Secondly, the ability to search documents intelligently. Microsoft, for example, is now providing help interfaces in the form of answer wizards, where a question is phrased in natural English and the documents are searched for the most relevant articles (Heckerman & Horvitz 1998). Although the technology employed in this case is relatively simple, to provide an accurate measure of relevancy through such an interface would require in-depth understanding.

Language understanding is a task distinct from language recognition but does share many of the computational dependencies.

1.3. A Classification of Natural Language Understanding Problems

In order to set the problem of automated assessment into context we will consider four different types of natural language understanding problem.

1. What does "string1" mean?
2. Does "string1" match "template1" "template2" or "template3"....?
3. Out of a set of "strings" which match "template1"?
4. Does "string1" mean the same as "string2"

The first is the most complex. As an open-ended problem it raises the philosophical issues of what it is to mean. How do we determine meaning and how do we demonstrate meaning? For computational systems it is difficult to see how we can approach this issue directly. The closest we can get is to take an entirely pragmatic, functional, Turing (1964) type approach. One way of tackling this problem it to attempt to demonstrate understanding by generating an appropriate response to an utterance. Conversational NLP models such as ELIZA (Weizenbaum 1966) and PARRY (Schank & Colby 1973) are the classic examples of such systems, and are indeed attempts to satisfy the Turing test for limited domains (that of a Rogerian psychoanalyst and a paranoid schizophrenic, respectively).

In the second type of problem the domain is more limited and the number of potential actions is fewer. The task of interpreting *"Computer: please save my document"* is within this class of problem. Here potential actions can be represented as templates and matching an utterance to a template is the same as saying is "*string1*" most similar too "*template1*" or "*template2*" or "*template3*"? etc. (Here a template is taken to be some arbitrary but matchable representation of meaning). This implies of course that the number of potential actions is predetermined and the various expectations can be generated. The in depth search application is also in this class of problem. If "string1" is the search string, each article consists of a series of strings, then the aim of the search process is to return the article that contains the string or set of strings with the closest match.

Interestingly, despite first appearances, the actual implementation of conversational models such as ELIZA and PARRY, are closer to the second class of problem than the first. The systems appear to generate sensible output in response to input and therefore tackle the open ended question of meaning. However in practice such behaviour is often illusory as the systems are implemented with a sophisticated set of pattern matches. The

question of whether pattern matching over a critical level of sophistication approximates to true meaning is legitimate, but shall not be tackled here.

The third problem is: out of a given set of strings, which strings mean the same thing where this meaning is embodied within a particular template. It is a problem that has largely been overlooked within the literature, as at first it is difficult to see applications that require this form of processing. However, it is this task that will be analysed within this thesis as it is this that most closely matches the problem of automated assessment. It is perhaps a more simple task than those defined above, but this simplicity is seen as its great advantage. Given the fundamental complexity of the natural language understanding problem any task which simplifies the problem but still addresses the critical linguistic issues (and in addition can exhibit graceful degradation of performance, see later) must surely be useful.

The final problem is a more sophisticated and complex version of the third, but in the context of automated assessment, a more ideal solution. It asks almost the same thing: out of a given set of strings which strings mean the same thing where this meaning is embodied within a particular string. The difference being of course that the meaning is embodied, or entered as a string, as opposed to some arbitrary template. In real terms this means that the person defining the criteria for success need only specify a single correct sentence rather than a template in some arbitrary syntax. It is of course far more difficult to implement. To illustrate the difference in complexity imagine making a cat/dog identification machine. The third problem is analogous to saying - "I'm going to build this machine hand crafting whatever sensory apparatus and decision criteria I wish to use." The fourth problem is analogous to saying - "I'm going to create a generalised decision making machine, this machine will then be able to tell me the difference between a cat and a dog after I have shown it just a single example of a dog."

It is the intention in this thesis to prove that a solution to the third of these problems is tenable. This does not necessarily mean a complete 100% success solution, but to achieve results that are significantly above those that would be expected from random. (Where by random, in a marking schema where an answer is marked either wholly right or wholly wrong, we would expect half of those marked incorrect to be in truth correct and vice versa). Only after this has been achieved will the fourth problem be tackled. It will not be tackled head on.

1.4. Automated Assessment

Specifically, the problem of automated assessment is this: a question is given to many students and many responses are generated. These answers are hand marked so it is possible to identify which answers are correct and which answers are incorrect. An artificial decision procedure must be generated which is capable of making this distinction between correct and incorrect answers. This decision procedure, if successful, will have the property that correct answers will be marked correct, whilst incorrect answer will be marked incorrect. Obviously there is no superficial characteristic of correct sentences upon which marking criteria can be defined. It is the contention of this thesis that a good performance can only be achieved if the knowledge base is capable of identifying the *deeper* semantic commonalties between correct sentences

As an evaluative mechanism, automated assessment is particularly useful due to the nature of the data produced. Many sentences are produced in answer to a particular question, and as these can be hand marked, we have the information of which is right and which is wrong. If a knowledge model is produced which does not work well, it is probable that only a few correct answers are identified and perhaps a few incorrect answers have been mis-identified as correct. A few minor adjustments could result in a marginally higher correlation rate. This means that automated assessment is an evaluation process that exhibits a graceful degradation of performance, in line with the quality of the knowledge model.

The task of automated assessment is to assess whether a response to a question is correct or not and, if the mark to be allocated is scaleable, to assess what mark should be given to it. If the tutor is able to generate an expectation of what the correct response should be then the task may be paraphrased thus: the task of automated assessment is to assess which responses out of a defined set most closely match tutor defined criteria.

1.4.1. Motivations for Automated Assessment

There are several motivations for developing an automated assessment system. Of the task itself: firstly, speed. In today's climate of reducing staff/student ratios, automated marking frees up staff time for more all important student contact time. Also, marking turnaround is greatly reduced, giving the students feedback more quickly. Secondly, objectivity: by marking all scripts with a single machine, a standard is introduced where formerly marker variability (both between different markers and the same marker at different times) was notoriously high. Thirdly, focus: it enforces a stricter definition of the syllabus which must be advantageous to staff and student alike. Students are more aware

of what they are expected to learn, and tutors aware of the knowledge they are supposed to impart.

Further, in line with an argument that shall be developed later within this thesis, automated assessment is a peculiar task which is ideal for empirically evaluating performance of natural language processing systems dealing with semantic extraction.

1.4.2. Problems with Automated Assessment

The notion of truth and its contextual variability raises some issues for an automated assessment system. Two questions may be set, and correct answers may be given for both questions. Both correct answers are 'true' in an abstract sense, but the correct answer for question two is not 'true' in response to question one, or at least a mark could not be given for it. To counter this the knowledge base in which information is stored could be marked up for its relevance towards particular questions. Note, this does not prevent sharing of information between questions, an obvious design criteria for any knowledge base system, but simply serves to identify the highest level 'facts' with their appropriate question. This is basically another way of saying: automated assessment is not the same as attempting to evaluate the truth of a specific sentence rather it is the task of determining whether a specific sentence or phrase correctly answers a specific question.

The issue must be raised of how variable marks could be allocated to question responses. Initially at least, this problem is best avoided by simply giving an all or nothing response to question responses, it is either all right or all wrong. At a later date this problem could be approached by either building composite answers out of parts (for example points to be raised in an essay) or employing fuzzy techniques.

Both the above problems refer to the logistical issues of the problem domain itself. However, by far the most complex problem is how to address the fundamental issue of assessing the semantic correlation of tutor defined criteria and student response. In brief, the approach to be used is the development of a representational schema which has processing as well as information storage capability. The development of this schema is based heavily upon careful investigation of a data set, which closely reflects the requirements of the problem domain in general.

1.4.3. Aims and Scale of Expected Success

It is important to stress that the development of an automated assessment system which is capable of discriminating correct from incorrect answers with 100% success rate is not expected. This would imply the development of a system which could comprehend the entire form and variety of the English language. Success is to be measured by several

measures. Firstly, the coverage a particular knowledge representation schema gives over a data set gives an indication of the system's semantic expressiveness and its applicability to the problem domain. Secondly, its ability to generalise to new unseen data should give a measure of the robustness and semantic coherence of the system. Finally, the extent to which the error reporting scheme correctly models of the deficiencies of the knowledge base gives an indication of the usefulness of using the framework to develop and refine knowledge base systems.

1.5. Outline of Task

In summary, automated assessment is suggested as an empirical framework within which knowledge representation schemas for natural language processing may be evaluated. Such a knowledge representation architecture is presented which is suitable for, not only high level semantic representation, but low level language processing tasks. From a careful analysis of the nature of the language presented by students, macro-structures[1] are developed within the knowledge schema which closely reflect the requirements of the problem domain. The architecture is evaluated within the automated assessment framework and refined.

Breaking down the path to this goal further and correlating with the chapters that are to appear within the thesis:

1.5.1. Investigation

Aside from the obvious areas of natural language processing and knowledge representation to be covered by a literature survey, a careful analysis of the form and variety of language is performed which concentrates upon those issues of language which are particularly problematic for computational systems. Where appropriate, mention is made of the probable information and processing requirements that a good solution to these problems would require. To focus the investigation, the analysis is based upon sample questions and responses, of a form similar to those likely to be used in the final system. This ensures relevance to the problem domain. Some work is also done placing the specific field of automated assessment into the wider context of natural language processing system evaluation.

---

[1] The terms macro-nodes and macro-structures are used within this thesis in a very specific sense. The terms are equivalent and are intended to embody the fact that a discrete subset of nodes and links may be considered as a whole and demonstrate functional and representational properties as an *ensemble*.

1.5.2. Knowledge Architecture and Learning Algorithms

A knowledge schema is developed which specifically addresses the issues raised in the above chapter at an architectural level. The schema draws heavily on the variety of knowledge representation schema that exist, but particularly the connectionist approach. The schema is especially similar to connectionism in that it incorporates an implicit processing component, through the passing of activation through a network. Unlike the more symbolic representation schemas (e.g. frames or logic) which would use a separate processing module to perform the semantic resolution, the mode of resolution is implicit in the network structure.

To address the principal problem of the large human effort required to produce these networks, two integrated algorithms are developed. The first algorithm is developed which generates networks of the type described in the section entitled 'knowledge architecture'. It is a statistical algorithm that operates upon corpora of linear strings of discrete units. From statistical co-occurrence of items it recursively identifies composite units, growing a hierarchical tree in this manner. Once identified new links are instantiated between pre-existing nodes and newly identified nodes that model the causal relationships between the two. The second algorithm also operates on these same network types. Similarly, from the same type of learning data, new clustered nodes are identified and appropriate links are instantiated between pre-existing nodes and newly identified nodes. A clustered node is a node that represents a commonality between two or more other nodes. This commonality is defined in terms of context history. Context histories are therefore recorded for each node, and it is in respect to these that similarities between nodes are identified.

1.5.3. System Architecture and Implementation

Within this chapter the architecture of the final system is defined, and the mode of communication between the distinct modules is outlined. Also, the specifics of the data resources available are reviewed and their method of employment specified. Particular issues of implementation are also discussed in this chapter.

1.5.4. Experiments

Two sets of experiments are performed, the first analysing the knowledge architecture, the second analysing the learning algorithms. First an experimental approach is devised and empirical measures of error to be used are mathematically defined which allow the knowledge framework to be investigated. Specifically two experiments are performed: the first, called the blind test, which determines the adequacy of the defined representational

schema to provide coverage of students; the second called the retrospective test which determines the ability of a configured network to generalise to new unseen data.

The networks that are produced by the learning algorithms are necessarily unsupervised in order to satisfy the constraint that little human effort is required in the network generation. The produced networks are therefore incapable of making a correctness decision in their own right; they simply provide a perceptual function, describing the same input data in a higher order form. The question of the evaluation of the networks is therefore complicated. Latent semantic analysis is introduced as a mechanism for performing this evaluation. The development of this evaluative framework and the description of the experimental results form the body of this chapter.

1.5.5. Conclusions

The final chapter summarises the conclusions that can be drawn from the previous chapters. Also some more abstract comments are made relating the implementation of the algorithms to what is known of human problem solving behaviour. Finally, consideration is given to areas of potential future development.

2. Investigation and Literature Review

2.1. Introduction

The investigation chapter will cover the following areas in order:

1) Analysis of properties of language
2) Review of knowledge representation schemas
3) Review of relevant natural language processing techniques
4) Review of automated assessment in the literature

In a little more detail: a thesis, which is to tackle some of the problems in natural language processing, and automated assessment specifically, must necessarily span a wide range of academic fields. The major fields to be considered are natural language processing and knowledge representation. We will start by analysing various properties of language, with specific reference to the type of language to be found in students' single sentence responses.

An effective schema for knowledge representation, which adequately models the specific requirements of the problem domain, is seen as the key to a successful system. To this end a survey is later made of the major representation alternatives and comment is made on their relative merits and demerits.

A full survey of the field of natural language processing is impossible due to the vastness of the subject domain. Within this review, attention is paid solely to the issues of adapting connectionist representation schemas to the field of natural language processing and some of the problems to be overcome for this adaptation to be successful, such as the representation of time. Secondly, attention is paid to some of the general problems of NLP that are particularly relevant to the target application of automated assessment and solutions to these problems that have been attempted.

Finally, a review is made of the field of automated assessment. Although this is a very young discipline and only lightly represented within the literature the two major applications that have been researched cover some useful ground. This work on automated assessment is placed in the context of more general research that has been undertaken in the field of the empirical evaluation of natural language processing systems.

2.2. Nature of Language

2.2.1. Problems Faced

In order to successfully represent the high order semantic information that is necessary to discriminate between a correct and an incorrect sentence, a clear analysis of the nature of language and the difficulties in representing the semantic information embodied within is vital.

This was performed by analysing in depth the responses given by students to a series of questions requiring simple single sentence answers. These sentences were collected in two sets from a total of approx 150 students over a period of two years (one set from each years intake). Twenty questions were asked of these students, and these students were expected to answer these questions in stressful, test conditions. (A broader range of sample questions and responses may be found in Appendix A.) While there are examples of many of these linguistic properties and difficulties in the literature, examples given here are mainly from the data collected. This is because text- book examples are often contrived and rare, whereas the examples from the data set clearly do occur and are not created solely for the purpose of illustrating a point.

The following lists those issues which were identified as particularly problematic, but which a knowledge representation scheme, if it is to be successful, must adequately model. Comments where appropriate have been made on the likely resources required to overcome these problems and the processing techniques which are most suitable.

2.2.1.1. Non-Reductive

The principle of reductive analysis implies that the whole is the sum of the parts. The corollary of which is: the sum of analysing the parts is equivalent to analysing the whole. Within the field of natural language processing, if this principle were to hold, the meaning of a sentence might be inferred by analysing the meaning of the individual words.

Within natural language this is clearly not the case, and there are at least two distinct reasons for this.

2.2.1.1.1. Ambiguity

Ambiguity is rife in the everyday use of language. A classic example is the word *"bank."* Examine the following two sentences.

"I am going to fish off the bank"
"I am going to the bank to get some money"

In the first example, clearly the speaker is referring to the bank to be found at the side of a river (a river-bank). In the second the speaker refers to a financial institution. A human reader would have no difficulty making the correct interpretation. The process of disambiguation is in most cases entirely unconscious.

However, if we wish to make our automated marker perform the same process of disambiguation we clearly have the problem of providing the system with enough information to perform the task. An entry in a dictionary will indicate that there are indeed two (or more) distinct senses of the word *bank*, but this in itself is insufficient to identify which sense a particular instance of the word *bank* refers to. For the human reader clearly it is the context of the word (the previous sentences, the sentence structure and the other words used) that make the disambiguation possible. The automated system must make use of this same resource.

There is a clear requirement here for a feedback type mechanism or blackboard architecture. This is the means by which a local process (in this case the correct sense identification of a particular lexical entry) is affected by information from a distinct local process (here this could be either the sense identification of other words or a syntactic parse of the entire sentence.)

A feedback type mechanism is obviously the necessary means by which disambiguation is performed, but it will require information resources (other than the sentence itself) to complete the operation. A list of the distinct senses for a particular word must be the primary resource. There are many possible sources for this type of information. Most dictionaries will list the distinct senses it attributes to each lexical entry, but the number and form of these senses can differ significantly in between dictionaries. Roget's thesaurus has a coarser grained list of senses, many of which can map to a single word. WordNet has a similar mapping of many senses to a single word.

Secondly a signature or a set of discriminating criteria must be identified for each of the distinct senses of a particular word. Each of the above resources (dictionary, thesaurus and WordNet) have associated information that may be used for this purpose. For example information may be extracted from a dictionary definition to make the distinction. Alternatively the statistical distribution of a particular sense may be analysed within a sense tagged corpus, and a discriminating signature may be generated from this.

2.2.1.1.2. Idiom

There are many cases in natural language where a set of words if found in a particular order, imply a meaning which can not be inferred from the meanings of the constituent words. For example.

> *"It came straight from the horses mouth"*
> *"It's raining cats and dogs"*

To take an example from the sample student responses, phrases such as *"operating system"* or *"assembly language"* clearly refer to something more specific than a system that operates or a language used to assemble things.

Within a knowledge representation structure it is taken for granted that semantic information can be attached to word level objects. However, if the representational structure is to take account of such idiomatic phrases, there must be a facility for attaching semantic information to units greater than a single word. Furthermore, if such a higher level unit is identified, the meaning that would have been associated with the constituent words must be inhibited.

2.2.1.2. Ambiguity - Combinatorial Explosion - Bootstrapping

The problem of ambiguity, as in resolving the meaning of an individual word with two distinct meanings, has already been mentioned. However the fact is, most words have, by dictionary definition, more than one meaning. This compounds the problem considerably. If a sentence is composed of 10 words and each of these words has 3 meanings, assuming all combinations are not mutually exclusive the sentence could have 3^{10} or 59049 distinct meanings.

In the above example of ambiguity the context of the word was deemed the necessary means of resolution. In reality, often the context itself is ambiguous.

To demonstrate the enormity of the problem let us consider the stereotypically simple sentence:

The cat sat on the mat.

Using the relatively small Concise Oxford English Dictionary as the source for identifying distinct senses of a word, the following table is produced.

Word	THE	CAT	SAT	ON	THE	MAT
Part of Speech	*adjective adverb*	*noun*	*verb*	*preposition adverb*	*adjective adverb*	*noun verb*
Meanings	definite article	small furry quadruped	supported by buttocks	supported by or covering	definite article	coarse fabric or floor covering
		spiteful or malicious women	rested with hind legs bent	close to		small rug
		person(Jaz z fan)	pose for portrait	concerning		piece of material laid on table
		whip (cat of nine tails)	to be an MP for a constituenc y	added to		to bring into a thickly tangled state
		any wild feline animal	remain on nest to hatch eggs	forward		
		abv for caterpillar vehicle	be member of committee	movement of operation being shown or performed		
		abv for catalytic converter	to be in session			
			to cause to be seated			
			remain in the same position			
Number of Meanings	1	7	9	6	1	4

Figure 2-1 Ambiguity in "the cat sat on the mat"

Assuming meaning independence of individual word this produces a possible (1x7x9x6x1x4=) 1512 different possible interpretations of this sentence, and this was using a very small primitive dictionary. If a larger dictionary is used an even greater number of permutations are generated.

If we have no *a priori* reason for preferring one interpretation over another.

> *"The spiteful lady posed for a portrait close to the table covering"*

Is as valid an interpretation as.

> *"The small furry quadruped rested with its hind legs bent supported by the small rug"*

Because four of the six words in the sentence are ambiguous the word *"the"* is the only definite context available in order to perform the sense disambiguation. This is clearly insufficient. In order to solve this problem a method of bootstrapped sense disambiguation will have to be used. This is a method by which multiple processes performed in parallel may affect each others outcome even though none are necessarily complete.

2.2.1.2.1. Samples

In considering the relation of ambiguity to the automated assessment problem domain and this data set in particular, there are two essential questions to answer:

1. How frequently does ambiguity occur within the data set?
2. Does ambiguity pose particular problems for the task considering the constraints involved?

Assessing the level of incidence and importance of ambiguity within a given data set is not easy. There are a number of reasons for this:

1. Firstly, in order to assess the level of ambiguity we need an almost prescriptive definition of the number of senses that are valid for any particular word. As has been mentioned earlier, there can be vast differences between different dictionaries' judgements on the number and form of these senses.

2. Secondly, the distinction between the different possible senses of a word and the metaphorical use of a word can be difficult to make.

If we are to take a dictionary as the authority on the number of senses of a word (in this case the Collins Concise Dictionary), the level of ambiguity within possible sentences is enormous.

Taking a sentence at random

"Allows use of data structures but not interrupt handling "

The dictionary gives us the following possible senses for the component words used within the sentence. Note for simplicity word stems (lexemes) are used circumventing the need for morphological analysis:

Word Stem	Number of Senses
allow	8
use	17
of	10
data	2
structure	6
but	12
not	3
interrupt	4
handle	15

Figure 2-2 Word Stems and Senses

Again, assuming *sense independence* this gives a total of (8x17x10x2x6x12x3x4x15=) 35,251,200 possible interpretations of meaning of the entire sentence. (Of course, the use of syntax could cut down the number of potential meanings considerably.)

This brings us to the second question: does ambiguity pose particular problems for the task considering the constraints involved? There are a number of reasons why ambiguity is not as big a problem as we may at first assume.

Firstly, in the above example we use a dictionary to give us the number of senses possible for a word; in a real system the optional senses are more likely to come from a domain specific knowledge base, which is unlikely to have such a copious number of senses per word. Also, it is worth noting that in general it is the frequent non-domain specific words

that have a greater number of senses. The domain specific words tend to have far less senses per word and it is these that are more relevant to an automated assessment system.

Secondly, the assumption of *sense independence* made above is clearly not a valid assumption. Each sense of a word is not equally likely, further, the choice of sense for a particular word within a sentence often has a clear effect on the choices of possible senses for other words occurring in that sentence.

Let us explore this issue more formally. Consider a notation where a sentence A consists of an ordered set of words, w_1, w_2, ..., w_n., i.e. A={w_1, w_2, ..., w_n}. If a particular word w matches a particular string S then S(w)=1. The distinct senses of S are denoted by the functions $S_1()$, $S_2()$, ..., $S_{ns}()$ where ns is the number of senses of S.

If w matches a particular string S then we impose the ideal where w must match one and only one of the senses of S i.e. $\sum_1^{ns} S_i(w) = 1$ and $\forall y \exists x, S_x = 1 \wedge y \neq x \supset S_y = 0$. The probability of a word w being a particular sense S_1 given that the word has been identified as string S is the Bayesian probability p(S_1(w) | S(w)), and obviously $\sum_1^{ns} p(S_i(w)|S(w)) = 1$.

So give two adjacent words w1 and w2, and two distinct identifiable string types S and T each with their own set of senses. And given that w1 and w2 have been identified as string types S and T respectively (i.e. S(w1) =1 and T(w2) =2), then in this notation the assumption of sense independence[2] is basically stating $p((S_1(w_1)|S(w_1))|T_1(w_2)) = p((S_1(w_1)|S(w_1))|T_2(w_2))$. That is, the probability of a word taking on a particular sense is independent of the sense taken on by adjacent words.

What does this mean in real terms? In a phrase such as *"empty file"* where *empty* has 14 and *file* has 12 possible meanings, if the assumption of sense independence holds true, it would mean the probability of *file* taking on the sense of *data* is completely unaffected by the sense taken on by *empty*. This clearly does not tie in with observed behaviour. On a brief analysis of the data, from 19 instances of the word pair *"empty file"* it is clear that in the context of *empty*, *file* consistently takes on the sense of *data*. In other words

---

[2] Note, to amplify the assumption that sense of a particular word is independent of the string type of the adjacent word can be stated as $p((S_1(w_1) \mid S(w_1)) \mid T(w_2)) = p((S_1(w_1) \mid S(w_1)) \mid U(w_2))$, where U is another string type.

$p((FILE_{DATA}(w_2)|FILE(w_2))|EMPTY(w_1)) = 1$ and even more specifically $p((FILE_{DATA}(w_2)|FILE(w_2))|EMPTY_{CONTAINING\_NOTHING}(w_1)) = 1$.

Why should this be so? The only sensible conclusion is that word sense distribution is not independent. Also, to explain the surprising Bayesian probabilities of 1 found above, it is reasonable to assume that word sense distribution is heavily dependant upon the nature of the corpus.

This concept contrasts interestingly with Church's (1988) notion of "lexical probabilities" where this refers to the likelihood that a word will behave in a particular syntactic manner. It is also in effect a subtle variation of the notion of collocations and mutual information (Rose 1993). The difference being, instead of looking at the effect the presence of one word string has on the probability of another word string being found adjacent, we are considering the effect that one word sense has upon the sense taken on by adjacent words.

In real terms, this means that the complexities that ambiguity can introduce are not as bad as it may first appear. The high domain specificity of the problem domain means that the sense distributions are highly idiosyncratic and focused. Also, the high domain specificity means that not only can the lexicon be of minimal size, but the senses that have to be considered for each lexical entry are small.

2.2.1.3. Anaphora

Anaphoric reference in language is a form of data compression. It is the means by which a sentence may refer to a noun already mentioned earlier in the text. It is a shortcut and a pronoun is often used. However multiple possible referents can cause ambiguity making the process of resolution non-trivial. This is a classic natural language processing problem.

The method of resolution must again balance multiple constraints against one another. These constraints may be of several forms: syntactic, semantic or constraints of recency. Syntactic constraints obviously relate to word classification, however the distinction between semantic and syntactic constraints can be somewhat nebulous. A semantic constraint is any element of a word's meaning which precludes it from matching the wider semantic net supplied by the pronoun. Examine the following examples:

> *My wife has a daughter.*
> *She likes to go to the cinema.*

My wife has a son.
She likes to go to the cinema.

The first example, because gender does not serve to discriminate the alternatives, is somewhat ambiguous. The second example exhibits no such ambiguity: the first possible referent is female, the second male, as the pronoun is female there is only one possible solution. This demonstrates anaphora resolution where the semantic information necessary to disambiguate is tightly tied to the word meaning, i.e. the sex. In this case the information is so tightly bound the necessary information could probably be derived from a dictionary definition. The following demonstrates a subtler example:

Peter has a dog called Rover.
He has strong sharp teeth.

Peter has a dog called Rover.
He takes him to the vet every year.

In the first example the pronoun *he* probably refers to Rover. The context of sharp teeth matches well with the notion of dog. In the second example *he* clearly refers to Peter as only a human will sensibly take something to the vet every year. There is nothing within the first sentence to disambiguate the two as the first sentences are identical. The contextual information is in the second sentence after the pronoun. But further the semantic contextual information necessary to disambiguate could not easily be inferred from a dictionary. This represents another example where what can only be referred to *real-world knowledge* is necessary.

Balanced against these constraints of syntax and semantics is the obvious constraint of recency. This simply means a pronoun is far more likely to refer to the noun in the last clause of the previous sentence than to a noun three sentences previous. From a psychological perspective this probably has much to do with the interaction of long term and short term memory.

Finally, a pronoun does not necessarily refer to a single lexical item. Frequently the referent is a whole clause or noun phrase.

My friend came to see me last night. It was an enjoyable experience.

In the above example *it* does not refer to my *friend,* nor does it refer to *last night*, it refers to the fact that *my friend came to see me last night.*

If an automated system were to attempt to tackle some of these problems the following types of resources would be required.

- A resource that would identify the constraints implied from a particular syntactic category.
- An efficient and accurate syntactic parser, which could identify syntactic class given a word.
- A resource that would identify the constraints implied from a particular word sense.
- An accurate disambiguation model which given a word and context would identify appropriate word sense.
- A logical scoping context which given a possible list of referents and an anaphoric reference would compare constraints against each other and come up with a best match.
- A process to adjust for recency, which given a previous set of sentences would identify a list of possible referents.

2.2.1.3.1.1. Samples

Anaphoric reference is not particularly common within the sample data[3]. To give some idea of the few contexts in which it arises the following examples are given.

> *"read a file or record or data before it has to be processed"*
> *"to be processed before it trys and reads it"*
> *"code sections which can be accessed then resume where it left off"*
> *"It relies on the weather conditions."*
> *"High Level as it needs to be compiled."*
> *"COBOL is a high level language because it uses english-like statements"*
> *"ones that have no following or proceding items. The item is done and nothing relating to it follows, and it then goes onto the next step of the program if one is there or exists."*

Figure 2-3 Sample Student Data

---

[3] There are in this section many examples of student data. Although there may indeed be genuine spelling mistakes within this thesis (although I hope not) those to be found within quoted student text are intentional verbatim copies of the text that the student in question used.

Note that from the above samples, and indeed from the entire data set "it" appears to be the most common form of anaphoric reference, with an occasional example of the use of "one" or "ones". This is to be expected as we are dealing with academic text that is concerned primarily with sexless, conceptual nouns.

Secondly, it should be noted that the referent is frequently found in the question. This means that answers cannot be considered as a self contained unit and the question and answer must be considered together in order to resolve the anaphoric references.

2.2.1.4. Synonymity

A single meaning can be represented by many words or groups of words. Or put the other way round, many words can mean the same thing. A good model of this synonymity would be the corner stone of a successful automated assessment system. If the aim is to produce a system that identifies the semantic correlation between two sentences, clearly a process that identifies the semantic correlation between component words is a valuable asset.

A thesaurus or WordNet are the two most likely resources for this type of information. However, an obstacle to the easy integration of such resources is the problem of ambiguity mentioned earlier. A typical lookup for a word entry in either of these resources can easily return upward of five distinct word senses.

2.2.1.4.1. Samples

For the purposes of automated assessment there is an important distinction to be made between theoretical, de-contextualised synonymity and two words that mean the same thing within a particular context. There are a number of reasons why this is the case.

1. A point (made more comprehensively in the metaphor section below) is that the meaning of a word is heavily affected or constrained by the context in which it occurs.
2. *Meaning* is a complex problem. The question of what does a particular sentence *mean,* as has been shown above, is extremely difficult to answer. The more accessible question is posed in automated assessment: Does SentenceA correctly answer the QuestionA?

It is the latter type of synonymity, therefore, that is of interest here: finding two words that *mean* the same thing within a particular context.

The following two examples were produced in answer to the question:

> *"Define the term 'low-level language' by completing the sentence 'A low level language is ....?'*

Answers:

> *"nearer to machine code"*
>
> *"close to machine code"*

In the above contexts *"nearer"* and *"close"* are synonymous in that using a simple replacement criteria they both adequately answer the question. However it is unlikely that anyone would claim that *"nearer"* and *"close"* are at all synonymous in the more general de-contextualised sense.

Again a similar situation is shown below.

> *"dependant upon conditions being met"*
> *"dependant upon conditions being satisfied"*
> *"dependant upon conditions being true"*

These are fragments from answers to a question concerning the definition of *"condition lists"*. Again in context *"met"*, *"satisfied"* and *"true"* have similar meanings, but in a wider sense *met"*, *"satisfied"* and *"true"* could not be considered synonymous.

As a final demonstration consider the following phrases:

> *"follow one another"*
> *"items following each other"*
> *"follow each other"*
> *"processed one after the other"*
> *"lead on singularly one after the other"*
> *"carried out one after another in an order"*
> *"processed one at a time"*

Each phrase is used to answer a question concerning the definition of sequence. This time the contexts are not identical, but are subtle variations of one another. The variability

captures well the difficulty of identifying the invariant aspects of language that correct answers share.

2.2.1.5. Metaphor

Metaphor according to a dictionary (Collins Concise) definition is:

> *"a figure of speech in which a word or phrase is applied to an object or action that it does not literally denote in order to imply a resemblance from, for example he is a 'lion in battle' "*

This poses a major problem for natural language understanding systems in general. However, its relevance to the specific automated assessment problem should be assessed first. Within novels and entertainment literature metaphor is undoubtedly pervasive. The sentence

> *George swam through the crowd.*

in the context of a novel is easily interpreted, and clearly the meaning here is not literal. But how frequent is the use of metaphor in the academic type prose under investigation here? A careful study of the sample responses produced by students uncovered some interesting points.

Certainly the overt use of metaphor is considerably less frequent than in novels etc. In fact out of 1280 sentences (64 student's responses to 20 questions) no overt use of metaphor could be found. The term overt metaphor here suggests that there are direct contradictions between a dictionary definition of a word and the use to which the word is put. However, a more subtle use of metaphor was found to be very prevalent. A good example can be found below.

> *"code closest to the machines architecture"*

This was produced in answer to the question *"What is a low level language?"* and is in fact a fairly good answer. But although it may not at first seem so, this answer uses metaphor. 'Close' according to a literal definition implies physical locality. A language, which is an abstract thing, can not be close to a machine architecture, which is also an abstract thing.

Such examples may appear pedantic, but it must be borne in mind that computers and the artificial reasoning processes implemented upon them are fragile and pedantic by

nature. Such examples also highlight the limitations of the above dictionary definition of metaphor. According to the above definition a word can be used metaphorically if it is an object or an action, but in this case it is neither. The term being used metaphorically here is *closest* which is an adjective, or an object descriptor.

By again making use of the terms context and constraint, as used frequently above, and accepting that contextual constraints can mutate or pervert the meaning of a particular word, a better working definition of metaphor might be:

> *"an instance of speech whereby the meaning of the word within the context only takes on some of the characteristics of the de-contextualised meaning due to the constraints implied by other word meanings."*

This would account for the above "lion in battle" example whereby the constraints of other words prohibit the individual taking on the qualities of yellow and furry but would not prevent him from taking on the qualities of strong and proud. Also, it accounts for the second example where the abstract nature of the terms being used prohibit physical locality from being a valid interpretation but would allow the more relaxed interpretation of conceptual locality.

In terms of the resources required to address problems of metaphor a mapping from individual words to contextual constraints is needed. Also an essentially fuzzy processing mechanism is required, whereby these contextual constraints may be checked against each other and loose or inconsistent implications pruned off.

2.2.1.5.1. Samples

Evaluation of the incidence of metaphor within the sample data is somewhat problematic. This is because in order to tell whether a particular word or phrase is being used metaphorically we need to do the following:

1. Identify the meaning of the word or phrase, as it is being used from its context.
2. Look up the literal meaning of that word or phrase from some prescriptive source, e.g. dictionary.
3. Identify whether there is significant difference between the "meaning in context" and the literal meaning.

Stages (1) and (3) are obviously subjective and possibly error prone.

With this caveat in mind the following examples demonstrate how meanings of words within the test data can differ from their defined literal meanings if we were to apply the strictest computer-type mentality to the sense matching:

> *"items which appear at 'the end' of a structure"*

"appear" according to *Collins Concise Dictionary Plus (1989)* has seven literal meanings none of which strictly mean "being at", which is the meaning implied within the above context.

> *"Make sure that before a program tries to process some data that there is actually some data to process"*

Again, this may appear pedantic, but a strict definition of "try" must include some notion of volition, of which a program, being inanimate, must be incapable. There must be some level of metaphorical usage here.

> *"A selection in JSP is the choice of two or more separate items only one of which will be chosen in the program"*

Similarly, it is only by stretching the literal definition that an inanimate program can be imbued with the power to choose.

It is perhaps also worth noting the comments of Garagliano (1996) who in connection with analysis of the data used for the LOLITA project stated:

> *"Our data has shown how, against popular belief, metaphors are as common in dry texts (eg financial reports) as they are in literature"*

2.2.1.6. Inference

Making the distinction between what a sentence means in its own right and what is it reasonable to infer from a sentence is far more difficult than you might first assume. In other words what knowledge is it reasonable or necessary for a listener to possess before he can understand the meaning of a sentence?

When dealing with natural language processing the knowledge that the system possesses prior to being supplied with the sentence must either be explicitly supplied or learnt by

the system. The most obvious source for word meanings is a dictionary, but is this sufficient to disentangle the sentence's meaning?

Consider the above lion example:

> *"he is a lion in battle"*

A dictionary definition of lion provides us with a definition:

> *"a large gregarious predatory feline mammal of open country in parts of Africa and India having a tawny yellow coat and, in the male a shaggy mane."*

This is obviously insufficient for a computer to be able to make the appropriate interpretation. It seems that a certain amount of what we call *'real-world knowledge'* is necessary to make valid interpretations of sentence meaning.

2.2.1.6.1. Samples

In order to demonstrate this same problem of inference in a more practical way consider the following example from the automated assessment samples discussed above. In answer to the question

> *"Define NON-RESIDENT' in COBOL terms by completing the sentence 'NON-RESIDENT' segments of COBOL code are...'?"*

The following answers were given.

> *"segments that are pulled in to memory only when they are needed."*

> *"swapped out to disk when they are not being used."*

Both these answers are correct, but are in a sense the *double negative* of each other. However the information necessary to ascertain that these statements mean similar things will certainly not be available from a dictionary. In order to make any sense of the above two sentences information along the following lines needs to be recorded somewhere.

1. MEMORY and DISK can store the same sort of things (INFORMATION)
2. SWAPPED in the context of DISK implies swapped from MEMORY

3. PULLED in the context of MEMORY implies pulled from DISK
4. If INFORMATION is not being USED then it is not NEEDED

A reasoning process would then be required which could apply the above knowledge to one of the above sentences to obtain a logical identity with the other.

On balance, however, the incidence of such answer variants within the data set is extremely low. Representing the two logical variants explicitly and distinctly as two distinct patterns is a pragmatic and achievable alternative. (Specifically examples of these types of answer variants were found in questions 1 and 13 only and the ratios between the two logical variant of the answer were approximately 80:20 for the former and 70:30 for the latter.)

2.2.1.7. Role of Syntax

The role syntax should take within a natural language processing system is a contentious issue. The conventional understanding is that syntax is lower in the processing hierarchy than semantic analysis. Typically NLP systems will parse a sentence first then ascribe meanings to the individual parsed components and finally assemble these.

However, for real world systems, of the type under consideration here, this processing paradigm cannot be applied. There are several reasons for this. First, a typical syntactic parser using a generative grammar, implemented as a bottom-up process, will never consistently provide a perfect parse of naturally occurring language (although coverage rates are constantly improving). The reasons for the difficulty in producing valid parses are twofold:

1. As discussed above words are inherently ambiguous. Where the distinct senses of a word have a different syntactic class a syntactic parse can no longer be implemented as a bottom up deterministic process (an example of this is the word *love*, *"I love you"* uses love as a verb, *"this thing called love"* uses love as a noun.). Higher level semantic information derived from context is sometime a necessity in order to arrive at a valid parse
2. A generative grammar produces valid grammar category strings from the recursive application of a finite set of rules. When a parse takes place a valid grammar category string is re-engineered to the rules that could generate it. There do exist a set of valid grammar category strings for which several distinct sets of grammar rules are acceptable parses. (Note this point contrasts interestingly with Sampson's (1987) argument against the grammatical/ungrammatical distinction).

The implication of a low parse rate is that sentences that cannot be parsed cannot have their content analysed.

Secondly, but possibly most importantly, the student responses are produced under test conditions; a naturally stressful environment. They are frequently ill formed, ungrammatical and contain multiple spelling errors. This considerably affects the performance of any syntactic parser.

Finally consider the following four sentences:

Romeo loved Juliet
Romeo felt great love for Juliet
Romeo felt lovingly towards Juliet
Romeo had a loving feeling for Juliet

All four sentence mean approximately the same thing. Certainly in answer to the question *"How did Romeo feel towards Juliet?"* they would all be marked correct. However syntactically they are diverse. The concept *love* is embodied as a verb, noun, adverb and adjective. Careful consideration must be given to the role syntactic analysis is to play in the automated assessment system.

2.2.1.7.1. Samples

The following sample phrases, taken from the data set, show how the concept of *dependency* is discussed in its various syntactic forms:

"*structure dependant on the data*"
"*depending on the input*"
"*depend on each other*"
"*depends on the selection*"

Similarly to demonstrate various syntactic forms of the concept *process*:

"*available to input during processing*"
"*operations that apply to a process*"

and for *test*

"*involve a lot of unnecessary tests*"

"being true thus not having to be tested"
"helps in the testing of the program"

Finally, for *access*.

"the main program able to be accessed by code"
"they are accessible by the rest of the program"
"accesses are allowed by"

Note that to adopt a processing strategy that does not presume a syntactic parse is not entirely unconventional. For example see Weischedel et al (1991) and Lou & Foxley (1993, 1994a, 1995), which will be discussed later in this section.

2.2.1.8. Noise and Redundancy

Natural Language is almost by definition noisy. Within the automated assessment task this noise can be categorised into:

1. spelling errors
2. ungrammaticality
3. sense confusion

The following sentences have been taken from the generated responses to demonstrate the depth of the problem.

"peices of code that is not in memry"
"segments of code that is called"
"i that is har to understand but is faster as the comp can trslate it easier"
"one that uses mneumonics that the programmer and the machine understand"

Such sentences pose a serious challenge to a system which is intended to match the semantic content against some prescribed criteria. Let us consider each of the problems to be tackled in turn.

Superficially spelling correction is a well defined problem: an ill formed word is easily identified by comparing it against a list of acceptable well formed words. Non set membership implies an error. On deeper inspection this solution is somewhat problematic. Due to the morphological irregularity of English words, a comprehensive list is difficult to derive. Further the inflectional morphology of English leads to an ever

expanding word list (Pinker 1994). Once identified the error must be corrected. Most approaches (Du & Chang 1994, Marzal & Vildal 1995, Zobel & Dart 1995) employ a technique which computes the normalised edit distances between two words. The edit distances are computed between the ill-formed word and all appropriate valid words in the lexicon. The valid word, which gives the smallest edit distance, is deemed the correct word. Although variations on this technique do exist (Kucick 92) a process with the same end results would have to be implemented. Note, though, that these techniques do not address the problem of a misspelled word coincidentally matching another well-formed word, a classic example being "*there*" and "*their*".

The problems with syntax have been discussed above. The introduction of noise compounds these problems considerably. The problems of spelling mistakes heighten the ambiguity surrounding the grammatical classification of words. Also, it is impossible to parse a non-grammatical sentence with a generative grammar because these grammars only produce grammatical sentences.

Natural language contains a high degree of redundancy. Shannon (1951) placed this figure at 50%. This figure was arrived at by investigating the level of content that could be eliminated from a sentence before it became unintelligible. Redundant in this context does not imply devoid of semantic content, but implies a duplication of semantic content. A strong argument could be made for redundancy being nature's way of resisting noise and damage to data. Nevertheless, redundancy is the only mechanism available to tackle the problems of noise within spelling and grammar.

2.3. Knowledge Representation Alternatives

The representation of knowledge is the key to the success of the outlined task. The representation module is the permanent store for the expert's knowledge. There are several alternative forms of knowledge representation commonly used: logic, frames, semantic nets, cognitive graphs and connectionism. These are now individually discussed.

2.3.1. Logic

Logic has a long history, its roots go back to before Aristotle. Logic can be seen as an attempt to formalise the reasoning or thinking process. Indeed one of the first major works on logic (Boole 1854) was called "The Laws of Thought." Logic took on its modern form with Frege (1879) who introduced the first complete theory of first order logic. However, the symbols now commonly associated with logic were first used comprehensively within Principia Mathematica (Russell & Whitehead 1910). Russell and Whitehead were also responsible for reducing mathematics to logic.

2.3.1.1. Propositional Logic

The most simple form of logic is propositional logic. Logic consists of statements which are considered either TRUE or FALSE depending upon their *interpretation* under a domain. These statements may be combined with well defined logical connectives, which are modelling or constraining the possible truth values of objects in the domain.

Although originally conceived as a tool to aid thought, the principal logical operators are somewhat divorced from their common sense counterparts. This is for reasons of internal integrity. The form of implication used in logic is known as "material implication" or the "Philonean conditional". It can be defined within a truth table as follows:

p	q	p⊃q
F	F	T
F	T	T
T	F	F
T	T	T

Figure 2-4 Truth table for the Philonean conditional

Lewis (1912) published a paper which objected to the ∨ and ⊃ operators being interpreted as their common sense equivalents of *or* and *if-then*. Specifically this is a question of implication. Common sense implication implies a causal connection between two events or states. Within logic this is not so and the truth value of the statement, with the if-then clause, is entirely dependant upon the truth value of its parts and not the constituents meaning.

Take the statement *"penguins can fly"* which is untrue. According to the above interpretation of implication this would render both the statements below correct.

IF *penguins can fly* THEN *penguins are birds.*
IF *penguins can fly* THEN *penguins are not birds.*

Both statements are, of course, in the real world, incorrect: there is no such simple rule for membership of the bird class. However because of the logical definition of implication and because the antecedent of both clauses is false, both of these apparently contradictory clauses are correct.

Implication is one manner in which logical reasoning differs from common sense reasoning; extensionality is another. To explain, the distinction between intension and

extension must be made. The extension of a logical token is the class of entities (in the real world or world model) to which the token correctly applies. The intension of a token refers to the logical properties of that token only. The extension of UNICORN is therefore NULL or the empty set, while the intension is rich. Since the truth values of symbolic logic are determined by the extension of the referent, a clause such as

$$\forall x \text{ UNICORN}(x) \Rightarrow \text{COW}(x)$$

or every unicorn is a cow, which is obviously FALSE by intension, is TRUE because there is no extension of unicorn.

2.3.1.2. First Order Predicate Logic (FOPL)

Propositional logic's expressive power is severely constrained by the lack of variables. First order predicate logic introduces these as well as some quantification operators. Instead of simple propositions, predicates are used which take one or more variables as parameters. These predicates are also either TRUE or FALSE depending upon their *interpretation* under a domain. The predicate taking a variable is therefore defining a set of objects for which a certain property is TRUE.

The variable under the predicate e.g.

$$P(x)$$

can be scoped with one of the operators.

The *universal quantifier* or:

$$\forall x \, P(x)$$

States that for all x the predicate P is true, or all x have the property P.

The *existential quantifier* or

$$\exists x \, P(x)$$

Asserts that ther exists at least one x that satisfies the predicate P (or has the property P).

2.3.1.2.1. Advantages of FOPL

Drawing in part from the analysis made by Pavelin (1988) the advantages of FOPL can be listed thus:

1) Precision

First order predicate logic is an extremely well-defined formalism, with well understood semantics. There is no ambiguity within the system; in fact the rules of formation prohibit this.[4]

2) Expressiveness

The use of variables, and the notions of quantification and negation serve to make FOPL an expressive formalism, within which, with some work, most statements are encodeable.

3) Proof Theory

Once the state of the represented model is encoded within FOPL, the rules of logical deduction can derive the truth status of anything that is implied by the state. That is the completeness theorem: anything that is true in all models of the world can be proved.

2.3.1.2.2. Disadvantages of FOPL

1) Deductive Reasoning

The reasoning, that logic models so well, is deductive reasoning. That is the process of reasoning by which a specific conclusion necessarily follows from a set of general premises. As both McDermot (1987) and Russell (1945) have argued most reasoning in the real world is inductive not deductive, where inductive reasoning is the process by which general principles are drawn from a set of principles, based mainly on experience or experimental evidence. As such inductive reasoning is far harder to formalise than deductive reasoning, for it is frequently an uncertain process, producing probabilities rather than absolutes.

2) Implications

The limitations of deductive reasoning tie in closely to the distinction made between common sense implication and Philonean conditional. As mentioned above, logic is

---

[4] Although well defined it is undecidable if used to support arithemetic. There is also ambiguity because although the symbols are precise the reasoning is free of semantics and can be performed purely syntactically. Any symbol stands for whatever you choods it to stand for providing it satisfied the axioms and statements set down for it. The underlying logic for set theory and Boolean logic are equivalent and form the same category, so every symbol in FOPL stands for a category of objects and as such is refered to as a functor or a function symbol which stands for any function of atom that fits.

commonly interpreted as being an extensional theory. It follows, therefore, that conditional statements that include objects having no extension have complex epistemological status. In other words we cannot describe the intension of a concept.

3) Non-monotonicity

A monotonic representational system is one where the set of all statements that are provable increases monotonically with the axioms added. A new axiom can never cause something that was previously provable, to become improvable, thus leading to a reduction in valid theorems.

However within real-world reasoning we frequently wish to express rules then stipulate exceptions to these rules. For example after saying all birds fly and there exists a bird called an ostrich:

$$\forall x \; bird(x) \Rightarrow flies(x)$$
$$bird(ostrich)$$

we may wish to refine this by saying ostriches do not fly:

$$\neg flies(ostrich)$$

Within monotonic logics this introduces an inconsistency after which everything becomes provable rendering the logic useless (Sowa 1991).

4) Boolean Truth Values

Within classic logics the interpretation of the model ascribes to a statement a value that is either TRUE or FALSE. However, from the common-sense reasoning angle, this is not flexible enough to capture the many vagaries of the world

5) Representational Recursion

Self-reference is a complex issue within formal systems such as logic (Hofstadter 1979).

But representational recursion, or the power to refer to propositions or other predicate names within other propositions, although prohibited, would be a powerful technique. One example of its usefulness is the power to represent equality of propositions. But a second and a more common requirement is the ability to represent information about propositions. For example "it is uncertain that..." or it is an "absolute necessity that..." or even "it is interesting that..." All such statements would be impossible to represent in first order predicate logic.

There are three major variations on logic each of which address some of the problems discussed above.

2.3.1.3. Modal Logics

Modal logics were introduced by Lewis (1912) to tackle the problem of the Philonean conditional, which he replaced with the strict implication.

$$P \Rightarrow Q$$

P implies Q, now can be read as: it is not possible that p should be true without q also being true.

It also addresses the last of the disadvantages listed above: the representation of knowledge about propositions. Hughes and Cresswell (1968) and Moore (1985) both give introductions to this theme. Specifically they deal with the problem of modelling an intelligent agent. Where a standard logic models the state of events within the world, if we wish to model an intelligent agent within this world we must be able to describe the agent's *beliefs* about the state of events within the world. Lewis, in his original paper, also introduced the notions of 'necessity', 'contingency, 'impossibility' and 'possibility'.

The principle by which these notions are introduced is by augmenting the semantics of the logic model. Instead of a single proposition being true or false under a particular interpretation upon a domain, there exist a set of domains upon which the proposition may be interpreted. One of these said domains is distinguished and is the actual world.

2.3.1.4. Non Monotonic Logics

Non-monotonic logics deal with the problem of adding axioms to a system. Specifically, axioms that lead to contradictions of the previous theorems of the system. Non-monotonic logics can do this without invalidating the entire deductive scheme. As explained above, this is of primary use in the representation of rules which have exceptions. Various formulations of non-monotonic logics can be found and although their theoretical bases may differ, there is a fair amount of common ground. McDermott & Doyle (1980), McCarthy (1980) and Reiter(1985).

2.3.1.5. Fuzzy Logics

In fuzzy set theory (Zadeh 1974) an interpretation of a proposition within a domain returns, instead of a Boolean variable (true or false) a real number between the value of 0 and 1. It is an attempt to capture the vagaries of the world, which is particularly

pertinent to the scope of this thesis: natural language processing. As Zadeh (1982) himself states: "Almost everything that relates to natural language is a matter of degree." Core to fuzzy logic is the redefinition of the principle logical operators AND, OR and NOT, to viable numerical equivalents, which preserve the logical integrity of the system.

When applying this formalism to the application of natural language processing careful consideration must be given to the epistemological status of the real value, between zero and one. The consensus within the literature is that fuzzy logic is intended to model "uncertainty" and many researchers go to great lengths to make the distinction between the interpretation of fuzzy logic under "uncertainty" and under "probability" (Kosko (1990))[5]. But within natural language there is a further temptation, and that is to interpret the fuzziness as conceptual locality, or semantic distance. There are two dangers inherent in this line of thinking. Firstly, is it valid to model conceptual locality along a single dimension? Secondly, is the integrity of the logical operators preserved under this interpretation?

2.3.2. Frames

The main premise of frame type representations is that knowledge should be clustered. This underlying notion of organising perception into chunks can be traced back as far as 1781 with the publication of Kant's Critique of Pure Reason (Kant 1787), but was revisited near the start of the century by Bartlett (1932).

From the AI perspective it was Minsky (1975) who stressed the importance of perceptual/representational chunking within the computational fields, and his represents the first attempt to formalise the process. He believed that most theoretical work in AI, as Ringland (1988) paraphrased: "was too fine-grained, local and unstructured to account for effective common sense thought." Schank and Abelson (1977) emphasised similar points from the psychological perspective with their theory of Scripts.

In essence a frame is simply a cluster of facts and objects and a set of inference strategies for reasoning about this cluster. The frame theory has three major architectural

---

[5] Some claim Kosko's version of fuzziness to be somewhat naïve. Using an underlying counting model a whole set of semantics can be derived. It is also possible to calculate the probability of a fuzzy event where fuzziness can be determined as a restriction of a set of probability families. The two probabilities are different as they operate on different support sets or universes.

components: frames themselves, slots and fillers. A knowledge base consists of a series of frames. Each of these frames comprises a series of slots. These slots define the frame in terms of its composition and properties. Each slot is instantiated with a particular value; this is the filler.

The values that populate the fillers are determined by the nature of the individual knowledge base. If the knowledge base is epistemologically circular and so self contained (i.e. like a dictionary), each filler will be another frame. However in most cases, although some fillers will refer to other frames, most will be defined as primitive data types by the application process.

One slot type of particular distinction is the IS-A slot. The epistemological status of the IS-A slot is a cause of some concern in the AI literature (Brachman 1983). Here it shall be used to simply define the type hierarchy and therefore the vehicle for defining inheritance routes. For this is one manner in which frames are distinct from logics: inheritance is defined within the architecture. This simply means if one frame inherits from another, by default the inherit-er is assumed to have (at least) all the same slots as the inherit-ee and the filler values are that of the inherit-ee by default, but can be overridden.

Take the following simple example:

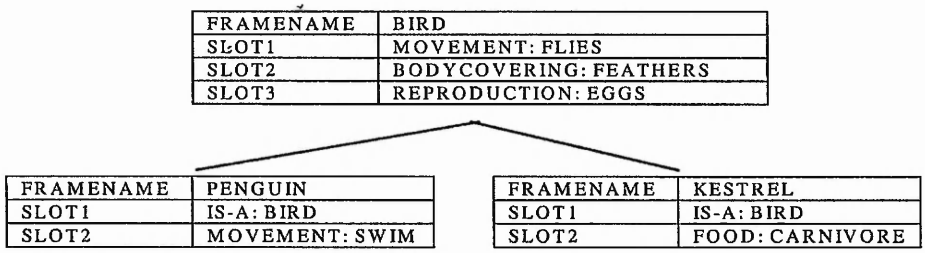


Figure 2-5 Frame inheritance example

This defines a fragment of a bird type hierarchy. A bird's primary form of motion is defined at the abstract level to be flying. Kestrel is defined as an instance of a bird by use of the IS-A slot. As such it inherits the flying property and an application process reasoning upon this data base would assume as such. The penguin also inherits from bird but its motion is overridden to be swimming.

In 1979 Hayes published a fierce attack on frame theory as it was presented in Minsky's original paper (Minsky 1975). Her primary criticisms were that the theory itself added little new to what is expressible in first order predicate logic and that there was a severe lack of analysis in the formulation of the theory. To quote him directly: "Minsky introduced the terminology of frames to unify and denote a loose collection of related ideas on knowledge representation: a collection which, since the publication of his paper, has become even looser. It is not clear now what frames are, or were ever intended to be." Brachman (1985) has also put his reservations about frames into print. He acknowledges that the frame theory attempts to address the "common-sense" reasoning problem but points out that the "epistemological ambiguities" within the architecture itself make a formulation of a reasoning knowledge base within the frame formalism a highly error prone process for the unwary.

However, to balance these comments it must be noted that the frame formalism still remains one of the most popular knowledge representation schemes for both theoretical work and practical working systems; particularly in the field of natural language understanding (NLU).

2.3.3. Semantic Nets

Quillian (1966) proposed the first major computer system to use semantic nets. Since this many people have taken up the idea in a variety of working applications. One of particular relevance to this thesis is the spreading activation form outlined by Collins and Loftus (1975). However, there is a great diversity to be found in the implementation and the specifics of the architecture. Johnson Laird and his associates (Johnson-Laird, Herrmann & Chaffin 1984) published an analysis of the variety of semantic networks available that identified only three common properties:

1. All nets are constructed from nodes and links, and there exists a set of interpretative processes to operate upon the inter-linked network.
2. Networks map out the relations between a set of concepts and within the network formalism nothing can be expressed about the relation between the concepts and the objects themselves (in other words extension can not be represented) . This is another way of saying that semantic networks are constructed on the assumption that intensional relations can be considered independently from extensional relations.
3. There is a general commitment to efficient representation.

The semantic net formalism has two architectural components: nodes and links. Unfortunately, much like the frame formalism, in its earliest incarnations the

epistemological status of its primary architectural components was deemed ill defined (Randal 1988).

Roughly, a network consists of a series of nodes interconnected by *different kinds* of associative links. Each node is defined in terms of its links to other nodes. For application to his desired field, i.e. natural language translation/understanding, Quillian (1968) found it necessary to introduce *different kinds* of associative links in order to deal with the "complexity of English definitions."

This touches on a problem that will surface frequently throughout this thesis, that is: over-imbuing the architectural components with semantic information can obscure reasoning processes operating on the knowledge base unless the epistemological distinction is clear cut and the reasoning process is aware of and can access this information. This reason has been put forward to explain the poor performance of Quillian's (1996) early systems, by Randal amongst others: "One of the reasons for the poor performance of Quillian's Teachable Language Comprehender, was that it did not take the semantic meanings of the links into account."

Two subsequent systems, that of Carbonell (1970) and Winston (1975) adopted a different strategy for link meanings. Carbonell allowed links to be labelled and thus richness or word inter-relationships could be stored within the link itself. Winston kept the links more homogenous, but in order to compensate for the reduced expressiveness introduced a new type of node, a relationship node, with which complex relationships may be modelled. However both systems are flawed through inconsistent application which is due largely to epistemological ambiguity within the architectural distinctions being made.

This same criticism may be levied against most semantic network models before 1975. However at this time Woods (1975) published a paper which evaluated the epistemological foundations of semantic networks. In brief his conclusions may be summarised as follows:

1. Identified the need for logical adequacy for semantic networks (this was later established by Schubert (1976)).
2. Make a firm distinction between structural and assertional links where:
    a) assertional links establish a relationship between two existing nodes and are represented with relational nodes.
    b) structural links defines the meaning of that node.

3. Deal with the intension/extension problem by regarding all implied statements as intensional but using an explicit existence predicate where necessary to state the extensional state of affairs.

To clarify the distinction between assertional links and structural links, it is worth noting that this is in fact a similar but more well grounded distinction to that made by Winston (1975) and even earlier by Shapiro (1971). However their solution was to represent the assertional links as nodes within the network.

Another interesting and distinguishing feature of semantic networks is that they make a type/token distinction at the architectural level. This is the distinction, say, between COW the type and ERMINTRUDE, a token of the COW type. The need for type/token distinction is a pervasive problem within AI and has obvious implications for the implementation for deductive and inductive reasoning schemes.

2.3.4. Cognitive Graphs

Although cognitive graphs have been criticised as offering "little new" to established knowledge representation schemes (Pavelin 1988), there can be no doubt that they do bring together what is best in previous schemes resulting in a flexible, extensive and precisely defined formalism. They were originally conceived by IBM employee John Sowa and were developed in his publication: Conceptual Structures: Information Processing in Mind and Machine (Sowa 1984). There have been later revisions to the initial work, e.g. Sowa & Way (1986), Sowa & Foo (1987) and a collection of conference papers Sowa, Minuau & Moulin (1993). Conceptual graphs are based on Pierce logic which is equivalent to FOPL but differs in that it is based on the $\exists$ operator rather than the $\forall$ operator.

Cognitive graphs are predominately logic based absorbing most of the formality of first order predicate logic inference. However, as the intended application is natural language processing they support many of the more common-sense reasoning features such as inheritance and modality.

An individual cognitive graph is like a small semantic network, which represents information at the same level as an individual sentence. There are three principal architectural constructs: *concept-type*, *concept* and *conceptual relation*. A concept type specifies a class of objects whilst a concept is an instance of this class; this is identical to the type-token distinction made within semantic nets. Conceptual relation simply identifies the relationship between concepts or concept types.

It is assumed that a partial ordering is defined on all concept types which serves the same function as IS-A links in semantic links or frames, i.e. defines the inheritance hierarchy.

2.3.5. Connectionism

Connectionism, parallel distributed processing and neural networks are all terms which identify a biologically inspired processing paradigm which has curious representational properties. Especially with regard to the representational status of hidden nodes within a neural network.

The primitive components within this architecture are: nodes and links. The links connect the nodes into an interconnected network. Each node possesses an activation which it passes to other nodes through the links. Each link can be weighted which modifies the strength of the signal, to either excite or inhibit. Each node has a threshold or threshold function which determines the node's activation on the basis of the incoming accumulated signal.

Formally, a particular node n_i is connected to j nodes: n_0 to n_j. A set of weighted links (w_{i0} to w_{ij}) make this connection. (This specifies the most general case of full network inter-connectivity where any node can be connected to any other - including itself). An activation a_0 is determined by the threshold function $t()$, defined on n_i. The activation of n_i is calculated as:

$$a(n_i) = t(\sum_{x=0}^{x=j} w_{ix} \times a(n_x))$$

A subset of nodes within the network are connected to the outside world and for these nodes their activation is taken as an *a priori* value.

One of the earliest connectionist systems is that of McCulloch & Pitts (1943). Their simple system consisted of two layers of nodes, between which full inter-connectivity existed. A linear threshold activation function was used. They showed that within such a system the logical functions *and*, *or* and *not* could be modelled. Logic theory shows that any logical expression may be modelled using these primitives.

Hebb (1949) was the first to suggest a biologically plausible learning model which could adjust the weight values such that *en masse* a network may act as a simple pattern

associator. In essence the rule was extremely simple: if two nodes are simultaneously excited then the weight linkage between them should be strengthened.

Rosenblatt (1962) continued this work with his perceptrons. This was a working model that has a precise mathematical formation and could further solve a wide variety of problems. Minsky & Papert (1969) pointed out two flaws in this work:

1. Due to combinatorial explosion the processing time necessary for the system to solve any but a few trivial problems makes it unusable (on a serial processor).
2. There exists an entire class of problems which the two-layer perceptron is entirely unable to solve (the most famous of which is the XOR problem) and there did not exist a training algorithm that could operate on multi layer perceptrons.

Due to the possibility of implementing such an algorithm on a parallel processor, and/or the huge increase in computing power which makes serial implementation for even quite large systems feasible, the first of these criticisms is no longer relevant. The second, however, dealt a serious blow to the connectionist community and research into this field essentially stopped for close to 20 years.

The breakthrough came with the generalised back propagation algorithm of Rumelhart & McClelland (1986)[6], which was a well founded learning algorithm which could operate on multi-layer networks. (Multi-layer networks are able to solve the XOR problem and others like it). They state that the only requirement is the use of an activation function for which the derivative is known.

In brief, the network has a set of input nodes and a set of output nodes. The network is trained with input patterns and output patterns. An error value is computable for each of the output nodes (usually some function of the difference between actual activation and desired activation). If we know the activation levels of the node leading into this node and the weights of the links upon which this activation is carried, we can assess the contribution each node makes to the error. If we also know the derivative of the activation function we can assess how much each weight has to be changed in order to minimise the error value. Errors may be *back propagated* throughout the network by summing the error contribution each node makes to the output nodes, and so forth[7].

---

[6] Although to some extent, the basic elements of the theory can be traced back to the book of Bryson and Ho (1969).

[7] Back propagation is essentially a negative feedback mechanism and as such is amenable to analysis techniques derived from control theory.

This computational paradigm has proved hugely successful on a whole set of problems with which the formal computational model had previously had much trouble. Specifically four subclasses of problems have been identified for which the connectionist approach has proved particularly successful.

1. Integrating diverse sources of information (McClelland & Rumelhart 1981) (McClelland 1981) (Hinton 1984)
2. Extracting prototypes from examples (McClelland & Rumelhart 1986)(Elman 1990)
3. Representing rules sub-rules and exceptions (Sejnowski & Rosenburg 1987)
4. Generalising from seen to new data (McClelland & Rumelhart 1986)

Interestingly, the connectionist paradigm circumvents many of the epistemological issues from which many of the formalisms above suffer. The input and output nodes have well defined meaning, determined by the causal, sensory process that activates the nodes on presentation with the input pattern. The epistemological status of the hidden nodes is somewhat nebulous and largely irrelevant, particularly for networks using distributed input/output representation and to which a learning algorithm is applied. Hidden nodes need not represent anything. Their presence is justified by the performance of the network as whole. This is taking an entirely functional, pragmatic view of the representation; a representation or token is justified if the existence of that token leads to a better performance of the system as a whole. Interestingly however, in several systems it has been found that hidden nodes evolve whose behaviour correlates highly with features to be found in the input data, and an internal representation can be seen to have evolved, the work of Rumelhart, Hinton & Williams (1986) for example.

2.3.5.1. Temporal Processing in Connectionist Networks

Processing data, which has a time dependent component, poses particular problems for a connectionist system. The dimensionality of the input pattern is generally limited to avoid combinatorial explosions between the hidden nodes. Therefore it is usually infeasible to encode many time frames within a single input pattern. Conversely, there is no direct analogy to long-term memory so the information from one time frame can be difficult to preserve in a distinct processing epoch. It can therefore be difficult for the network to make useful associations, or perform any general processing, between the information resident in different time frames. There are a number of solutions that have been proposed to this problem.

2.3.5.1.1. Windowed Input Patterns

The simplest solution to the time representation problem is the time window approach. Here data is encoded within a fixed time 'window'. This is the approach that was taken in the NETTALK system (Sejnowski & Rosenberg 1987). Under this system the input vector used is a concatenation of inputs from a series of time frames. The number of time frames defines the size of the window. This is limiting in that the size of the input vector is dependent upon the number of time frames the process has to take into account.

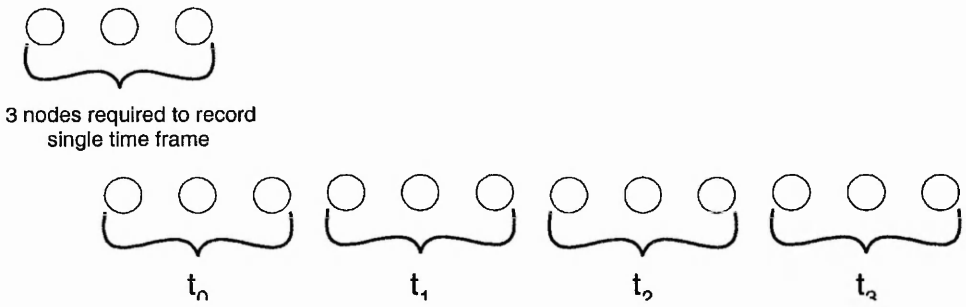


Figure 2-6 Windowed input pattern encoding 4 time frames

2.3.5.1.2. Simple Recurrent Networks

An alternative is the use of a simple recurrent network (SRN) originally devised by Elman (1990) as a modification of a related structure developed by Jordan (1986). With this technique instead of the input pattern from previous time frames constituting part of the primary input vector, the hidden layer is fed back upon itself using time delay nodes. As the hidden layer is deterministically computed from the input layer it forms a valid if different encoding of the state of a particular time frame. Further, as the hidden layer is frequently smaller than the input layer it can be interpreted as a compressed version of the same. As such, the network necessary to compute the time dependent information is frequently smaller. Such networks can be used for prediction tasks (Servan-Schrieber et al 1989) amongst others and trained with the simple back propagation algorithm (Rumelhart et al 1986).

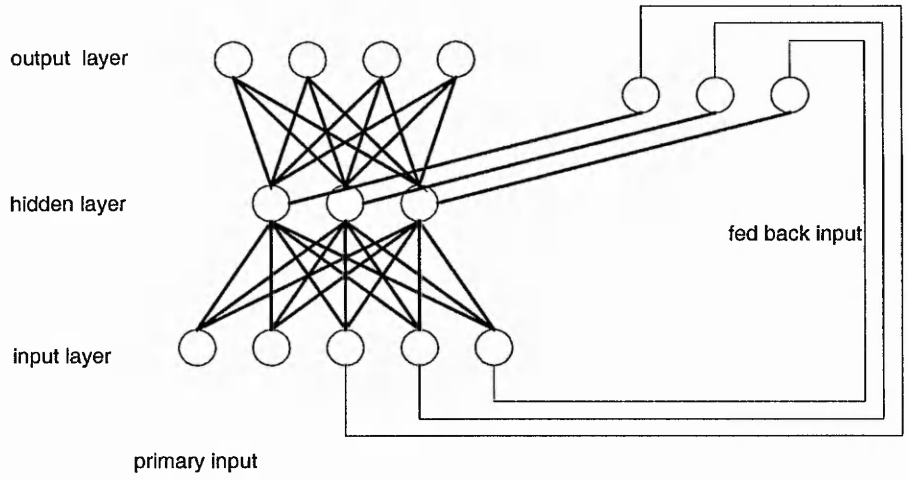


Figure 2-7 A simple recurrent network

This technique is not as computationally expensive as the above 'windowed' approach, even if we wish to take into account many time frames into the past. This is due to the nature of the activation propagation. By simply adding a single set of delay nodes to re-circulate the hidden layer nodes' activations, automatically the input vectors from many previous time frames are taken into account in the computation of the output nodes. This is because the activations from the very first time frame never entirely disappear. They are simply diluted with new input. Each previous time frame simply becomes less important in the final calculation, in fact the rate of decay is proportional to the ratio of the weights between input layer and hidden later, and the delay layer and hidden layer.

2.3.5.1.3. Time Delay Networks

A third option is the use of time delay nodes within a network to model the temporal latency between component nodes of the input vector. Such an approach is used by (Lucas & Dampner 1992) in their syntactic neural networks.

2.3.5.1.4. Short Term Memory

As analysed within Gasser and Lee (Gasser and Lee 1992) both the above techniques are essentially contributing a short term memory to the connectionist process. This avenue is explored further in (Allott et al 1995) and indeed the fruits of which contribute largely to the second half of this thesis.

2.4. Natural Language Processing

A complete survey of the natural language processing field is obviously not possible within the space available here, nor is it required. A general introduction to some of the problems and standard approaches to NLP can be found in (amongst others) Allen (1995), Winograd (1972), Carbonell & Hayes (1987), Collier (1994), and Dahl (1995). Here we consider the work of a few individuals that is directly pertinent to either the target application (automated assessment) and its inherent problems, or the relationship between the chosen knowledge schema (connectionism) and natural language processing issues.

2.4.1.1. Connectionism and Computational Linguistics

Harris (1992) presented four aspects of natural language that pose particular problems for conventional symbolic natural language processing methods, but for which, he concludes, connectionism is better suited. As a vehicle to introduce some of the major problems that face natural language understanding and in order to review some of the linguistic approaches to these problems, these four issues will be quickly summarised.

1. Schematicity: this alludes to the fact that natural language can be described at varying levels of specificity: syntactic and semantic levels being the two most obvious. Conventionally linguists have restricted themselves to the analysis of only one of these levels. Harris claims that the 'entire schematicity continuum' must be used if all the regularities of a natural language are to be captured. Further, idiosyncratic utterances (which are far more frequent than may be first expected) have their own schema which must be represented in order to correctly extract the information from them (Filmore et al 1987).
2. Non-autonomy of Syntax: this is simply a corollary of the above which purports that syntax (or syntactic analysis) as an isolated sub task is not solvable. This is largely due to the ambiguities that are so prevalent within natural language. Further, there is work that suggests that certain syntactic sub-tasks such as noun-phrase extraction are influenced by semantic factors e.g. (Kuno 1987) and (Kluender 1989).
3. Non-compositionality of Semantics: the compositional theory of semantics as first proposed by Katz and Fodor (1963) is a theory of lexical semantics which equates an individual lexical item with a set of semantic features. The aggregation of the semantic features therefore defines the real-world referent to which the lexical item alludes. An example would be to equate the term *bachelor* with the semantic feature set [HUMAN, MALE, UNMARRIED]. There is a certain amount of psychological research which supports this view (Miller & Johnson-Laird 1976),

and a fair few natural language understanding systems have performed well while operating under the basic tenants of compositional semantics (Schank 1975). However since its inception problems have been noted. Bolinger (1965) noted that the compositional theory works fine for concrete nouns but considerably less well for verbs and abstract nouns. Miller (1978) among others (Miller & Johnson-Laird 1976) noted the theory's inability to model polysemy, or the change in sense of a word due to context. However this charge may be levied at any localised theory of lexical semantics which by definition does not take context into account. Jackendoff (1983) gives a fuller exposition of these criticisms, but further suggests an epistemological shift, or reinterpretation of the basic theory which counters some of these criticisms. That is to interpret the composite parts as conditions of use or preference rules rather than the more fixed components.

4. Constraint Satisfaction: this issue is a reiteration of the point made in issue (3) above, that is: a localist theory of lexical semantics is insufficient to deal with the constraints imposed by context. Brugman (1981) and Brugman and Lakoff (1988) demonstrate clearly the profound effect of context on meaning by a detailed analysis of the variety of uses and implied meaning of the word *"over"*.

2.4.1.2. The Difficulties with NLP

2.4.1.2.1. Lexical Disambiguation

Lexical ambiguity is a pervasive problem within natural language processing. The polysemous nature of *most* words whereby a single lexical item has several meanings or interpretations, means that a method of semantic resolution must be present if interpretative systems are to be developed. Miller et al (1991) propose a corpus based algorithm for lexical disambiguation. A parse is assumed such that the grammatical status of each word can be identified. When an ambiguous lexical item is encountered the alternative senses are listed and the context of the item is noted.

Using an external resource such as WordNet (Miller et al 1985) or Roget's Thesaurus (Gwei 1987) a *synset* can be generated for each sense alternative. A corpus is searched for similar contexts and the co-ordinating item (an identified discriminatory criteria) for each of these contexts identified. The co-ordinating items for each of the contexts are matched against the members of each of the synsets. The ambiguous sense which has the synset with the greatest number of hits is deemed the correct item.

2.4.1.2.2. Sense Matching

Many NLP systems rely on a process whereby text input is matched against a predetermined template. The ATIS (Air Travel Information System) project focuses on NLP problems within the travel domain. A team from SRI International lead by Jackson et al (1991) have produced a template matcher for robust natural language interpretation. Working from the principal that, within the travel domain, although the volume of information requests is high and the variety of actual utterances is high, the variety of the logical form of the request is very low. The task tackled was to reduce the large number of possible utterances to small number of logical requests (which were eventually translated into database requests). This was done by means of a robust template matcher.

A small number of templates are generated for each of the request types; four in the initial case. The template is rendered using a frame type of knowledge representation. Each frame has a set of slots, and each of the slots has a set of associated keywords. A scoring mechanism is devised using the words in the utterance with the number of slots and the keywords per slot. The template which scores highest under this mechanism becomes active. Once active the template fills its slots with filler values derived from the utterance.

The strength of this work are that an explicit parse is unnecessary, and so "the failure to find a complete parse can no longer be used as a hard constraint to reduce perplexity of the speech recogniser." However the lack of a complete parse is also its downfall in that, due to the reduced amount of information, the filling of the slots is a more error prone process. This is addressed by combining a parsing element onto the system (Moore & Dowding 1991).

2.4.1.2.3. The Syntax Semantics Relation

Approaching the NLP problem as a modular sequential process is doomed to failure. The ambiguities of natural language mean that semantic analysis is impossible without syntactic structure and syntactic analysis is impossible without semantic information. The work of Weischedel et al (1991) represents an approach which recognises the impossibility of each of these tasks in isolation. Concentrating on NLP tasks as exemplified in the message understanding conferences (MUC-3) Weischedel attempted to match natural language utterances to pre-defined classes or templates. A standard NLP system architecture is as follows (see Figure 2-8).

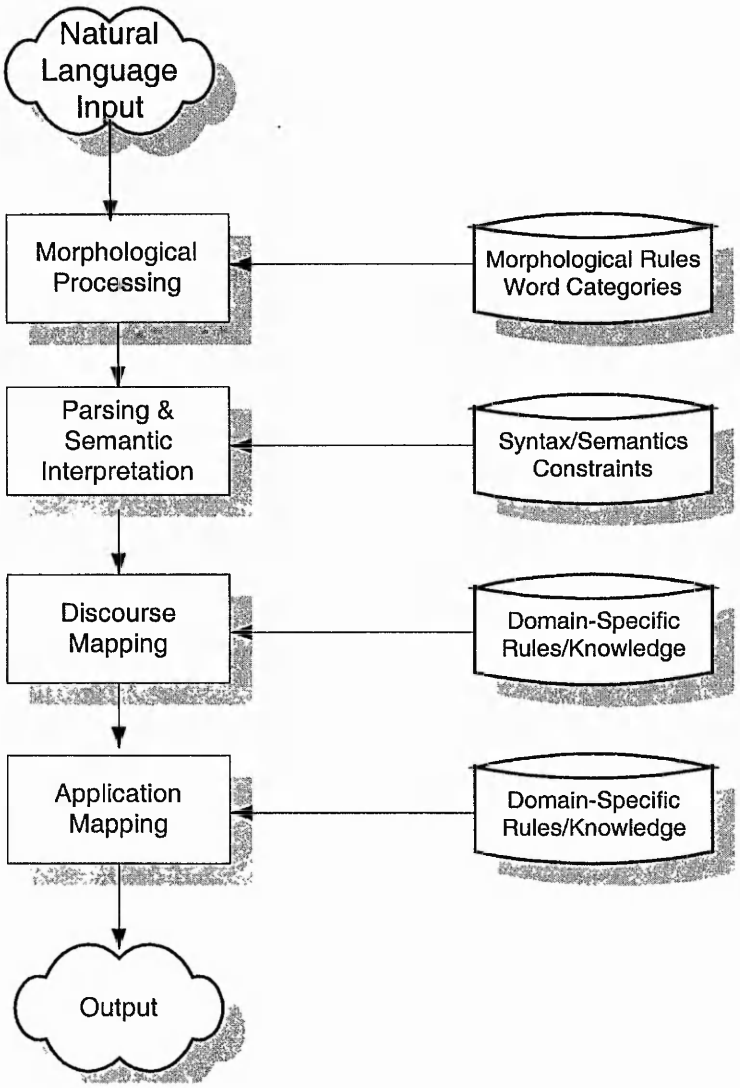


Figure 2-8 Standard NLP Architecture

As with most message understanding systems the process relies on being able to match a natural language utterance to a predefined class or template. In order to do this the main constituents or referents of the sentence need to be identified. This is typically performed through a syntactic parse. Syntactic parses when applied are rarely completely successful. It is more common to produce a series of partial parses. To use their own analogy; instead of a single parse tree being produced with a single root, a forest of parse trees is produced with independent roots. The approach that Weischedel[8] et al propose to tackle this problem is as follows.

---

[8] Earley (1970) has also published work along similar lines, documenting an efficient "chart parser" which is capable of producing partial parses.

1. A syntactic parse is produced which will typically produce many distinct partial parses. A semantic interpreter assigns semantic interpretations to the constituents of the partial parses.
2. Using the greater information provided by the semantic assignment, a probabilistic algorithm combines the fragments acknowledging both semantic and syntactic information.

The above represents an algorithm which recognises the difficulty in implementing a natural language system as a modular sequential process, and proposes a tenable procedure for combining the information from the two sub-tasks in an interleaved manner.

2.5. Automated Assessment

Finally, we consider the application of automated assessment itself. This is done in two phases; firstly a survey of the specific techniques and solutions to this task available in the literature is given. Secondly, we define the scope of the investigation to be undertaken. This is done by placing the application of automated assessment in the context of more general work concerning the evaluation of natural language processing systems.

2.5.1. Automated Assessment Literature

Within the literature identified, only two[9] instances of automated assessment research have been identified. The first, Marshall's intelligent marking assistant (Marshall 1986), documents an assisted process of essay marking and evaluation. The second, the STAMS, system researches a fuzzy logic based model for marking single sentence answers to questions. Finally the features available within QuestionMark, a commercially available system for marking simple text answers, are evaluated.

2.5.2. Marshall's Intelligent Marking Assistant

Marshall's Intelligent Marking Assistant (REPORT) tackles the problem of automated assessment of student scripts (these are essays not multiple question/answer scripts).

---

[9] There is actually a third (very recent) automated assessment technique to be found in the literature, that of Latent Semantic Analysis. This will not be introduced till chapter 5 for the necessary background theory is more naturally presented there.

Marshall classifies his system as an expert system in that it possesses the following elements:

1. A Knowledge Base to contain the expertise.
2. A Reasoning Engine to make inferences; applying (1) to the specific case.
3. A User Interface to allow the user to communicate with the system

However, although its function is to assess the validity (semantic content) of student scripts, it is not in any real sense a natural language understanding system. To justify this assertion the system must be explained in greater depth.

The knowledge base itself consists of a series of rules of classic expert system form:

```
IF p AND q THEN z
```

The antecedent conditions p and q are criteria by which aspects of the script are evaluated. The consequent z can be of one of two things.

Either it is an action - in this context this could be a modification to the running mark for this script or possibly a prompt for the user to make a subjective evaluation of an essay fragment; or the assertion of a new *state* which could lead to the satisfaction of a different rule thus allowing rules to be chained.

For example a typical rule may be:

```
IF (Z% of the marks for any report X are available for some
section Y) AND (Z% is greater than 0%) THEN (X should
include Y)
```

The problem solver component triggers these rules as determined by the evidence within the script. In line with many implementations of expert systems it attempts to produce an explanation of the evaluation arrived at. Explanations are implemented by tagging individual explanations to each of the triggered rules and so a trace of rule activation can be followed.

As far as can be discerned the conditions are of two types.

1. Automatically assessed conditions that are implemented as small functions. These will be such things as: word/sentence count, paragraph length and simple

syntactic evaluations such as punctuation/capitalisation validation and subject verb agreement.

2. Subjectively assessed conditions: which by definition can only be satisfied by the user's intervention. These will be rules such as (Is the main Body of text coherent?). In the working system when rules with such conditions are triggered, the system prompts for the user's evaluation.

The final output of the system is a mark and an explanation. As noted above the explanation is generated as a series of 'canned texts', but like the ELIZA system (Weizenbaum 1976) there is a random element to the canned texts to introduce variety.

The REPORT system does not attempt to evaluate or classify the semantic content of texts. All semantic evaluation that is necessary is performed by the user, albeit in a structured and computer driven way. The system could be seen as an NLP system by virtue of the syntactic evaluations that are automated, however for the reasons explained above it should not be seen as an NLU system.

The work however is highly relevant to the task in hand. Firstly, it addresses the same problem: how to objectively and efficiently evaluate student work, in the context of ever increasing staff/student ratios. Secondly it is a paradigm of structured evaluation; in Marshall's own words: "it can apply the evaluation criteria, make objective judgements and supply explanations consistently."

2.5.3. A Simple Text Automatic Marking System (STAMS)

The STAMS system (Lou & Foxley 1993,1994a,1995) again tackles the problem of automated assessment but as distinct from above, concentrates on: "marking the semantic content of single sentence responses in a marking context." It forms part of the wider Ceilidh project (Abdullah & Foxley 1991, Benford, Burke & Foxley 1992, Lou & Foxley 1994b) which deals with computer managed course assessment in a broader perspective.

The aim is to match sentences of similar meaning, for example a tutor specified target meaning could be

The big hedgehog eats the small caterpillar.

It is desired that the automated marking system could identify all the following as having similar meaning.

The huge hedgehog kills the small worm.
The great hedgehog eats the little catepillar.
The small larva is killed by the large hedgehog.

At the core of the system is a fuzzy logic (Zadeh 1965) inference engine which derives its fuzzy parameters from Roget' Thesaurus through a series of independent tools (Gwei & Foxley1987,1989, Gwei 1987).

The 990 or so headwords within Roget' s Thesaurus are taken as the primitive concept types. The tools developed by Gwei and Foxley, when given a word (present in the tutor criteria or student sample) to look up, will return a list of all possible relevant primitive concepts. The following table illustrates a few of the most relevant headwords when the word 'file' is looked up.

Headword No	Relevance	Confidence	Primitive Concept
71	100%	100%	Continuity
548	52%	100%	Record
136	40%	100%	Lateness
332	40%	100%	Powderiness
194	37%	100%	Receptacle
228	37%	100%	Smoothness

Figure 2-9 Sample thesaurus output

According to Lou and Foxley (1994a): "the relevance value for a headword determines its ordering amongst the paragraphs considered, while the confidence value is a measure of the certainty of its meaning relative to the word given." He also explains: "These measures depend on the number of occurrences, the degree of suffix/prefix manipulation used in deriving the term for each occurrence and whether the occurrence is a phrase containing a derivation or the derivation by itself." For further explanation see Gwei and Foxley (1987).

The fuzzy membership of a particular word in a concept is derived from the above data. In cases where confidence is rated 100% the fuzzy set membership is equal to the relevance factor.

The multiple fuzzy memberships of a single word can therefore be expressed in a form (taking just the 3 most relevant terms):

$$\mu_{word} = \frac{1.0}{\mu 71} + \frac{0.52}{\mu 548} + \frac{0.40}{\mu 136}$$

Where, μ0 to μ990 (the denominator) represents an orthogonal vector space upon which an individual word's meaning is plotted and the variable on the top (the numerator) of the fractions indicate the extent to which a word shares meaning with the abstract thesaurus headword (relevance). In a sense μ embodies a words/phrases meaning and the μ0 to μ990 is the orthogonal vector space upon which aspects of this meaning are plotted. The above syntax should be read as indicating set membership and not numerical division.

And so the multiple fuzzy memberships of two words (a phrase) are expressed in a multi dimensional equation:

$$\mu k^2_{i \times j} = \left(\frac{1.0}{\mu 71} + \frac{0.52}{\mu 548} + \frac{0.40}{\mu 136} \right) \times \left(\frac{0.60}{\mu 21} + \frac{0.77}{\mu 14} + \frac{0.36}{\mu 232} \right)$$

By defining an operator which assesses the overlap between two such phrases of arbitrary dimensionality, it is proposed that a fuzzy measure of the semantic similarity of two textual phrases can be assessed. The operator suggested for identifying such an overlap is the STAMS distance formula. Defined for example on a 3x4 and 2x4 phrase as:

$$D\mu E \times \mu K = 1 - \max(\min(\mu E^2_{2\times4} \times \mu K^2_{3\times4}))$$

The architecture of the system can be pictured diagramatically as follows:

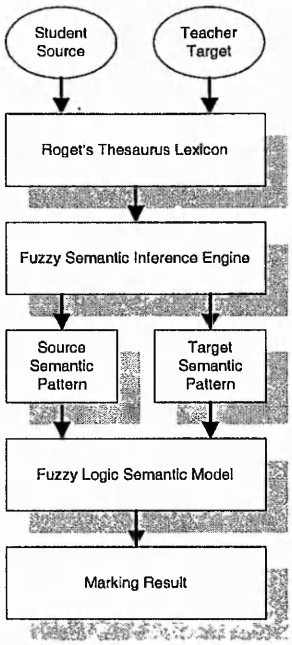


Figure 2-10 STAMS process architecture

The above discusses the aggregation of semantic components from the word level to the phrase level and the subsequent comparison of semantic aggregates against each other. As yet nothing has been said of the semantic information that resides above the word level; principally the case role's themes that are indicated by the syntactic structure.

The approach that has been used here is simplistic, but arguably necessarily so, for it is imperative that the process is as robust as possible. Essentially, case roles can be inferred from consistent word order, except for passive phrases where the assumed word order must be reversed. Passive phrases are identified by the independent storage of the passive verb in the lexicon.

However this approach is flawed. To discriminate between the active: "a tiger eats a cow" and the passive: "a cow is eaten by a tiger"; eat and eaten are stored independently in the lexicon. However this can lead to confusion if a sentence is phrased: "a tiger had eaten a cow".

2.5.4. QuestionMark

Question Mark is a commercial system for automating the dissemination of tests and the automatic collection and marking of the results. It deals in the main with the simple question forms i.e. multiple choice and single word response. However, a module does

exist for marking free text answers. This is done by means of complex Boolean search phrases. Although the core technology is quite simple, it is possible to phrase relatively complex marking criteria within this formalism. To generate Boolean search criteria for answers, over even a very low level of complexity, proves extremely difficult to do. Especially so as there are no in built tools to help in this process.

2.5.5. Evaluation of NLP systems with Automated Assessment

Finally, it is the purpose of this section to give an overview of the task of automated assessment. After an analysis of the general problems and possible approaches to automated assessment, some specific examples are. Most importantly, the use of automated assessment as an evaluative mechanism for knowledge representation schemas operating within a natural language processing context is presented and justified.

2.5.5.1. Question Type Alternatives

2.5.5.1.1. Multiple Choice

Multiple-choice questions are a well-established question type and automated methods of marking such scripts are actually used in many national exams. Such systems work by making students fill in pre-formatted papers with a pencil: drawing a heavy line between blocks for example. Machines then sense the graphite marks left upon the page and compare these against the stencil of the correct marks.

The method of assessment is well established and therefore poses no challenge. There is however substantial literature on the question types that are appropriate for multiple choice questions, the range of alternative choices that should be supplied and the distribution of mark we should expect a body of students to produce if it is to be considered a fair test. Some of these aspects may be relevant to the investigation.

2.5.5.1.2. Single Word

The automated assessment of questions requiring a single word response poses somewhat more of a challenge than that of multiple choice questions. However, again real world systems (such as QuestionMark) already have modules which can quite competently perform this function.

Over and above multiple choice, single word evaluation is problematic in that the input can be fuzzy, variable and/or damaged. Some form of error correction is necessary in order to capture all, not just the exact matches to be found within the scripts. Error correction on this type of data is little more than spelling correction, which is a relatively mature (although still perhaps imperfect) science. Although the responses are single words and therefore have no context with which to augment the constraint satisfaction procedures used for error correction, the high level of domain specificity and the strong expectations that can be generated of responses can be used in their place.

Systems such as QuestionMark do not enter this level of complexity when evaluating damaged data, and in real terms the only way of coping with such input on real systems is for the marker to hard-code the expected variations.

2.5.5.1.3. Essays

At the other end of the spectrum is the marking of essays. For essays, however, formalising a marking scheme can be very difficult. Interestingly the marking criteria within academic fields differ, not simply between problem domains, but by subject area and *style* of teaching and evaluation used therein. Not surprisingly the most marked differentiation is between the arts and science subjects.

Going a step further than Marshall's (1986) attempts, if it is our aim to evaluate the semantic content of an essay by an automated assessment procedure we must have some criteria by which to evaluate the script. The most obvious criteria to use for such a task is some sort of *model answer*. For arts based subjects, such as English literature, a definitive model answer which captures the sense and variety of what was an acceptable analysis would be virtually impossible to create. In this thesis we shall concentrate on science based subjects, such as physics, computer sciences or geography. For these subjects the creation of a model answer is frequently an integral part of the examination process, although just how frequently they are in reality used as a marking template is debatable.

An analysis was made of the questions, scripts and model answers produced for the 1989 set of exams set by the Computing Department at The Nottingham Trent University.

The questions themselves were a mix of single sentence questions that required a full essay type response and series of shorter questions each expecting a paragraph or more in response, the sum of which could be seen as a consistent script. For both question types, however, full model answers were prepared. Both model answers were similar in

form: a point by point analysis of the elements of the answer expected and a mark allocation for each point.

On the surface these may seem like viable criteria against which the scripts may be evaluated. In principle each defined criterion could be *fuzzily matched* against each sentence within the script and a mark awarded in proportion to the number of good matches found. However, on closer analysis, comparing model answers to actual marked scripts, this was found not to be feasible. Although the marking scheme held in general, frequently marks were given for points raised outside of the marking scheme. Furthermore, there were many occasions where there existed no single section of the script where a particular point was raised, instead it was implicit throughout the response, or the point spread over several sentences. This would be extremely problematic for an automated assessment system, if implemented using a reductive sentence by sentence analysis.

On a more general note, since there is a tendency for students to use a far more expressive form of language in essay based answers, they are consequently far more difficult to mark by formal, artificial means than a more constrained question type. Also there are certain extremely problematic linguistic mechanisms, such as anaphoric reference, that are far more prevalent within essay based answers. Therefore in this work it was deemed more appropriate to perfect a more simple constrained question type before any such essay type questions were attempted.

2.5.5.1.4. Single Sentence

As a consequence the mid range problem of single sentence semantic evaluation was chosen as the target application. Essentially this is the same problem identified and attempted by Lou and Foxley (1994a) however there are a few essential differences in the approach taken here:

1) Primarily the difference is the core technology employed. A connectionist based system is being used as both the evaluative mechanism and the representation schema (as opposed to fuzzy logic and a fuzzy logic reasoning engine).
2) A heavy emphasis is put upon the application of the system to real data, and the performance of the system is to be evaluated with this data.
3) The application itself (single sentence automated assessment) is presented as a general method for evaluating the performance of any combined NLP/KR system.
4) It is intended to implement learning algorithms which will simplify the knowledge creation phase.

As mentioned earlier such tasks may be interpreted as semantic correlation. Given a tutor-defined *target*, the aim is to find instances of student sentences that correlate well with this prescribed meaning. By drawing on a range of educational evaluation texts (Nelson (1970), Grondlund (1981), Chase (1978)), questions types were constrained to three possible types. Each of the question types was intended to constrain the syntactic and semantic variety of the expected response.

The three question forms were:

1) Complete the sentence...?
2) Define ...?
3) Iterate...?

A quick example of each question type is given below.

Complete the sentence: *"'One to One correspondence is .....' : complete this sentence."*

Define: *"Define the term SEQUENTIAL in Jackson Structured Programming (JSP) by completing the sentence "Sequential items are...."."*

Iterate: *"List the extra 3 schematic titles resulting from a JSP selection item ALPHA that has three options BETA, GAMMA and DELTA. The schematic logic item ALPHA-SEL would be followed by these three matching 'ALPHA-xxxxx'.*

2.5.5.2. Comments

2.5.5.2.1. Scoring

For simplicity of analysis as well as enforcing a strict discrimination upon the tutor a Boolean marking scheme was employed, such that answers were marked as being either completely correct or completely incorrect. The decision process of the automated assessment was similarly constrained. This facilitates an easy comparison of results between methods.

2.5.5.2.2. Marker Variability

Whether a particular answer to a question is in any absolute sense correct is philosophically a highly problematic question. This problem is circumvented in the outlined scheme by supposing the independent conclusion of the tutor is a good approximation to the absolute truth value of the answer. However, it must be recognised that this is an approximation only, and in recognising this, obvious variability that can

occur between markers must be admitted. To counter this, great care was taken over the generation of the question, such that the form of the expected responses was not only maximally constrained but that the semantic content could be objectively evaluated. Should marker variability still be considered an issue the *truth value estimator* (the tutor's decision) could be augmented by using more evaluators and taking the consensus.

2.5.6. Difficulties in Evaluating Natural Language Processing

As noted by King (1996) the evaluation of natural language processing systems is surprisingly sparsely represented within the literature (certainly as of five years ago). However, it has recently been identified by many bodies as a core interface technology as regards the future and the use of computers, and efforts have been made to formalise the evaluative mechanisms to encourage sharing of information and techniques. Most notably:

- DARPA/ARPA conferences (DARPA proceedings 1993 etc.): the DARPA/ARPA series of evaluations covers a variety of speech and natural language processing tasks and sub tasks each of which is formally evaluated.
- MUC (Message Understanding) conferences(e.g. MUC-3 1991): deal with information extraction from message streams of a variety of forms.
- ATIS project: (Boisen & Bates 1992): concentrates mainly on spoken language tasks but within the common application domain of air transport.
- TREC series of conferences (Harman 1995): deals with the domain of text retrieval.
- TSNLP initiative (Lehmann 1993)(Balkan et al 1994) (Test Suites for Natural Language Processing) is an academic initiative to develop widespread pre-standard diagnostic and evaluations tools for both developers and users of NLP applications (but with an apparent bias towards machine translation.)
- EAGLES initiative (draft interim report 1994): (Expert Advisory Group for Language Engineering Standards) is a European Commission instigated initiative which builds on the ISO 9126 standard for quality characteristics to be used in the evaluation of software and applies it to NLP.

King (1996) in her extensive study of the field of natural language processing evaluation identifies two primary criteria under which evaluations should be made. Firstly, the issue of functionality and usability. Does the system do what is intended and does this have a useful real world application? Can the system easily be made to perform this function, in other words is it useable? This issue also extends to extensibility. Many NLP processes are typically highly domain specific. Can this domain be switched easily?

Secondly, there is the issue of validity and reliability. Can the system perform the function intended for it and can it do so consistently and correctly? The problem of assessing the correctness and consistency of the system will be addressed next.

Flanagan (1994) made a thorough analysis of error types within the domain of machine translation. Great importance is placed on the fine-grained analysis of error type for two reasons. Firstly, it comprises fine grained, detailed feedback on the performance of the systems. Secondly careful analysis of error types and distribution can help identify problem errors and how best to correct them.

Given that an analysis of error types is clearly desirable, the question of how to generate this error data must be addressed. The most common criticism levied against NLP evaluative systems is that the human-assisted nature of the evaluations, inevitably creates a subjective element and "raises the issue of the extent to which the human is being evaluated as much as the systems performance" - a point made by King (1996). This issue is also raised by White et al (1994) as well as some more general concerns of the validity and reliability of the measures.

Interestingly, one of the first formal evaluations of a functioning NLP system (ALPAC 1966), despite being criticised for being politically motivated and deliberately biased, circumvents the issue of human assistance and subjectivity, and provides a rigorous analysis of the performance of the system. Again the target domain was the heavily researched discipline of NLP: machine translation. As is the case with most NLP an element of subjectivity is inevitable, for the human determination of meaning is the only reliable process we have for "getting at the truth." However, the critical distinction of this process is that human evaluation is used as the *control* against which the automated system is compared; the critical comparison between the two systems is as procedural and objective as possible. In other words, a task is given to system and human in parallel, and the performance correlated with human performance the target. This is as opposed to an automated system performing a particular function and a human evaluator assessing the output of the system.

Finally, a thorough analysis of the whole field of NLP evaluation has been made by Sparck-Jones & Galliers (1993), a summary of which exists in Sparck-Jones (1994). In this analysis an important distinction is made. That is between glass box and black box evaluations, where a glass box evaluation relates to metrics which are implementation specific and black box evaluation deals with application performance only and is therefore implementation independent. This ties in very closely with the distinction made between application task and application sub-task. An important point is made that as NLP

applications are often quite diverse, comparison between systems is difficult due to the different application specific metrics employed. However, where application sub tasks correlate strongly with distinct linguistic phenomena (e.g. parsing, anaphoric reference, disambiguation etc.) cross NLP application evaluation and comparison is easier.

2.5.6.1. Automated Assessment as NLP Evaluation

With these NLP evaluation issues in mind the task of automated assessment is proposed as an application within which the performance of a knowledge representation schema operating with a natural language processing domain can be evaluated. There are several points which support this.

Firstly it is a macro application, the metrics for which can be compared between systems. But it is a macro application, which if it is to perform well, must subsume all other necessary linguistic sub-tasks (from disambiguation to logical implication), and therefore gives a general measure of linguistic competence.

Secondly, there exists an objective evaluation scheme by which performance can be measured. This scheme can be entirely automated. Further, as the evaluation of the component elements (individual sentences) is a Boolean process the performance of the entire system can be reductively and unequivocally compared against the human target performance.

The task lends itself to the generation of large amounts of input data, i.e. many sample sentences from many students all in answer to a particular question. The evaluation measure of the performance of the knowledge base therefore degrades well. The same cannot be said of many template matching systems where a single template is matched against a single stream of data; if there is no match or it is matched incorrectly there is no partial measure of success.

Finally, many tricky epistemological issues to do with the meaning of words and sentences are subtly avoided. A sentence is deemed true simply if its truth-value correlates with the assessment of the tutor. The large-scale statistical nature of the analysis and the diversity of the responses generated, which is due largely to the stressful exam type setting of the answer collection process, make systematic bias within the evaluation highly unlikely. Should it prove necessary the subjective nature of the tutor's evaluation may be further diluted by taking the consensus of a number of human markers.

The specific measures that are used to empirically investigate the application are developed and defined in Chapter 5.

2.6. Conclusions

Natural language is complex. All the features discussed in the "Nature of Language" section are present to varying degrees within the test data. The aim of the automated assessment task is to assess the similarity between sentences in order to divide the set into two groups: those that are correct and those that are incorrect. With this in mind, a good model of synonymity is essential to the success of the task. However, a number of factors mean that a model of synonymity is both insufficient in itself and non-trivial to achieve:

- Non reductive aspects of language such as idioms require distinct modelling.
- Ambiguity can make the assignment of a sense to a string very difficult.
- Noisy data, that is produced under exam type conditions, can mean that that some strings are not readily identifiable as there is no exact match with any of the contents of the lexicon.
- Metaphor and the general contextual effect of other words, mean that words can take on senses that are not necessarily one of the *prescribed* senses.
- In order to ascertain whether a particular question is correct sometimes it is necessary to compute the logical implications of a sentence rather than performing a bottom up word sense match.

To balance this, the high domain specificity of the specific questions and the limited lexicon involved mean that many of the language features described above are less important than in the analysis of true unconstrained natural language. This reduces the complexity of the problems but does not eradicate them.

Automated assessment is a difficult problem. Potentially, it touches on many academic disciplines each of which is as deep as it is broad, for example: knowledge representation, natural language processing, computational linguistics and AI processing techniques. A thorough review of all the potentially relevant disciplines and sciences would be impossible. As the success of the finished system is going to be heavily dependent upon the efficiency and accuracy with which the tutor can specify the criteria, which correct answers have in common, a large part of the above review is dedicated to comparing alternative knowledge representation systems. A cursory examination has also been made of some computational linguistic techniques that have the most relevance to the anticipated NLP problems that automated assessment will throw up. Finally, time is

dedicated to considering connectionist processing techniques, specifically those that deal with temporal data. This will be more relevant to the latter part of this thesis, which will consider learning algorithms for the knowledge base.

In the last section the specific field of automated assessment was considered. First, the instances of automated assessment to be found in the literature were considered. Next, a case was developed for the use of automated assessment for the evaluation of knowledge schemas operating within the natural language processing domain. Specifically the task of the automated assessment of free text single sentence responses to simple constrained questions was suggested. It is believed this offers a fair compromise by introducing some of the complexity of natural language but falling short of the full scale complexity of unconstrained prose, such as that found in essays.

The evaluation of natural language systems is a very active area of research. Some of this work is summarised above and an attempt is made to identify the qualities a good evaluative system should possess. This establishes the relevant background from which a full battery of tests may be formally defined in order to give a good measure of the knowledge schema's various aspects of performance.

3. Knowledge Architecture and Learning Algorithms

In this Chapter novel data structures and algorithms are developed for an intended application to the problem of automated assessment of student text. This will be done in two phases: firstly, the development of the knowledge architecture itself. It is within this architecture that the data must be embodied which is capable of making the sentence judgement decisions. Secondly, to address the problem of knowledge base generation, algorithms shall be outlined which are capable of at least partly automating the generation of instances of these data structures from statistical analysis of student answers.

3.1. Knowledge Architecture

Many of the alternative knowledge representational schemas to be found in the literature have been discussed within the literature review, Chapter 2. To solve this very pragmatic problem of automated assessment an essentially connectionist schema has been developed. There were two overriding reasons for making this decision as outlined below. (However a fuller justification will only be possible once the main architectural elements of this knowledge schema have been discussed and these will be found toward the end of this section.)

1. Architectural simplicity: the connectionist schema has a simple architecture; there are no major epistemological distinctions made at the architectural level. It therefore does not fall prey to many of the representational ambiguities that can affect logic, frames and the like. It is the intention that the required representational and computational complexity can be modelled from these more primitive units. Specific macro-structures will be refined which model the particular requirements of the problem domain. Further, the tutor can interact with the knowledge base more easily due to this simplicity.
2. Processing: the connectionist schema has an implicit processing component, through the passing of activation through the network. The distributed form of processing this facilitates is particularly appropriate for the type of constraint satisfaction problems that characterises natural language processing as it neatly circumvents the inherent bootstrapping problems.
3. Learning: this is possibly the most important. It is intention to implement learning algorithms to operate on the developed knowledge structure. It appears from a cursory overview of the literature that connectionist systems are more

amenable to and have better success with the implementation of general purpose learning algorithms; both supervised algorithms and unsupervised algorithms. Again, these issues shall be discussed in greater detail towards the end of this section.

3.1.1. Architecture

The knowledge architecture developed in this work tackles the complexities of representing the information within natural language by combining the best features of the alternative representation schemes discussed above. An essentially connectionist scheme is proposed where the reasoning engine is implicit in the passing of activation from node to node. But unlike most connectionist schemes each node (even hidden nodes) is symbolic, in that it represents a distinct, identifiable feature within the problem domain. (This approach is similar in many ways to that of Rumelhart and Mclelland (1981)).

This novel knowledge architecture is similar in many ways to a semantic net, but has borrowed heavily from many of the mainstream KR schemes.

As the user has to interact directly with the knowledge base, an emphasis has been put on simplicity. The architecture is essentially a hierarchically structured activation-passing network (APN). Such an architecture was chosen because it seems to model well the aspects of natural language that seem most important.

No distinction has been made between the KR and NLP levels. Also the architecture departs from the symbolic paradigm where a distinction is conventionally made between data representation and an engine used to process it. The architecture corresponds more closely to the connectionist paradigm where the reasoning capacity is embodied within the activation passing aspect of the network itself.

3.1.1.1. Node

The architecture proposed consists of a set of nodes. Each of these nodes has a unique identifier. Nodes can be further subdivided into nodes that receive input from the external world (evidence nodes) and those that receive input only from each other (hidden nodes) and output nodes, thus following the conventional neural network paradigm.

3.1.1.2. Links

Any pair of nodes may be bound together with a link. This link, at least in the knowledge schemes initial incarnation, is Boolean in nature. That is, there either exists a link for

two nodes or there does not. This link is the means by which nodes pass activation between one another and is unidirectional in nature. Therefore with respect to a link there is a child node (the node from which activation is passed) and a parent node (the node that receives activation.) A particular node can therefore be both a child and a parent node, but defined by two distinct links. Conceivably it is possible for a link to have a negative weighting that will mean that the activation of a child will have a dampening (as opposed to an excitory) effect on the parent node.

3.1.1.3. Activation

Every node has an activation value, this represents the extent to which a feature exists within the input pattern. Again, in the initial incarnation of the knowledge scheme this will be a Boolean value, either 0 or 1. Evidence nodes assume an activation of 1 if their feature is identified within the input pattern. Hidden and output nodes assume an activation of 1 if the sum of the activation passing through the links exceeds the threshold value (see below).

If a node does become active, activation is propagated through its output links to the parent nodes.

3.1.1.4. Threshold

Each node also possesses a threshold value. This is the value that the sum of the activation, which is attributable to the input links, must exceed if a node is to be activated.

3.1.1.5. Operational Modes of Nodes

Depending upon the configuration of the network, in other words the topology of the links and nodes, and the threshold value of a particular node, a node can operate in one of three modes.

3.1.1.5.1. Evidence

As mentioned above nodes are primarily divided into evidence nodes and hidden nodes. An evidence node receives input from somewhere other than the network itself. The mechanism by which a particular evidence node is activated is largely unimportant. It could be an external filtering mechanism which activates the nodes manually or the perceptual intelligence may be programmed into the node itself. (By this term "perceptual intelligence" I mean the code/system that is capable of tying raw sensory input to a symbolic item that is deemed to have some meaning within a computational system.)

Within the system to be used here, evidential nodes are used to represent the low-level word knowledge. Each has associated with it a list of words (evidence list) whose presence in the sentence will lead to the activation of the node. This evidence list may contain all the morphological variations of a word, common spelling variations on a word or even abbreviations. In this case the evidential node is analogous to a lexeme.

For example

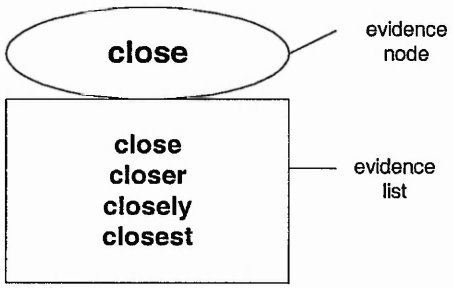


Figure 3-1 Evidence Node Example

3.1.1.5.2. Clustered

A clustered node is of the hidden node sub-type so receives its activation from the links from other nodes. In the situation where a node receives input from multiple other nodes and the threshold is less than the number of links from which it could receive input then the node is said to be an clustered node.

A clustered node can be used to represent the abstract sense of several different nodes. It is then a disjunction of nodes and will become active if ANY of its children become active. In its simplest sense such nodes can be used to represent synonyms. Logically an abstracted node is equivalent to an OR gate (for Boolean activation and linkage models).

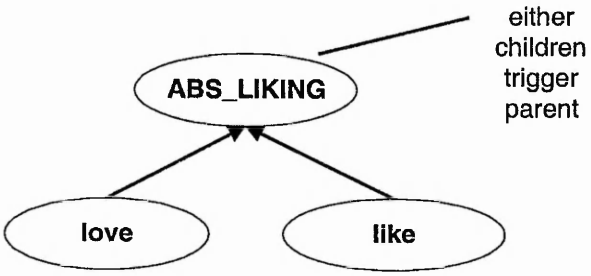


Figure 3-2 Clustered Node Example

As an aside it is worth noting that it is possible to implement evidence lists in terms of abstracted nodes. For this application we have an evidence list for multiple nodes that contains a single entry which is identical to the node name. We then have a single clustering node which represents the superset of all the "evidence nodes".

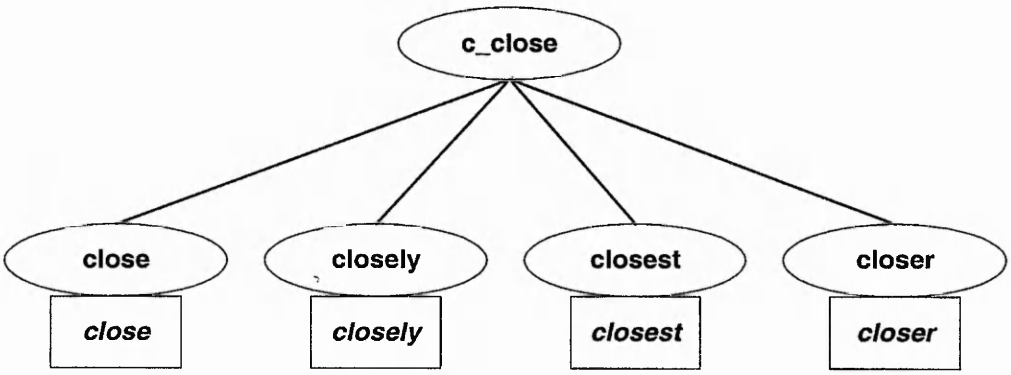


Figure 3-3 Implementing Evidence Lists with Clustered Nodes

For purely practical reasons this approach is not taken. These practical reasons being that:

1. Using the graphical user interface supplied it is marginally more efficient for the tutor to create knowledge bases in this manner.
2. It introduces an architectural distinction within the database that correlates strongly with a functional distinction; this will facilitate an analysis of the knowledge base on pure functional grounds at a later date.

3.1.1.5.3. Composite Node

A composite node is again of the hidden node sub-type. In the situation where a node receives input from multiple other nodes and the threshold is equal to the sum of links from which it could receive input then the nodes is said to be a composite node.

A composite node is used to represent the compound sense of several nodes. It is a conjunction of nodes and becomes active only if ALL of its children become active. A fact can be represented with a composite type as a conjunction of abstract types. Logically a composite node is equivalent to an AND gate.

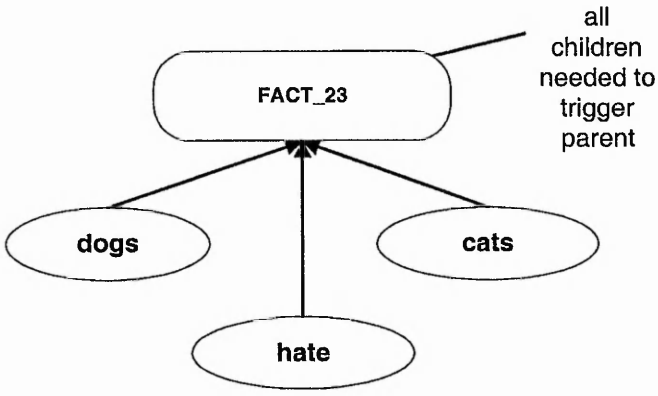


Figure 3-4 Compound Node Example

Note this simple structure on its own takes no note of word order.

3.1.2. Network Qualities

Next we consider the manner in which sets of these nodes and links may act in collaboration to model/solve various natural language processing problems. We shall not concern ourselves here with how these structures will come into being[10], but satisfy ourselves that such structured elements do holistically model the intended problem. Such collaborating nodes and links have been termed "macro-nodes".

3.1.2.1. Abstraction:

Many words in certain contexts have similar meanings (e.g. synonyms). This abstract similarity may be adequately modelled by an interconnected network in which any ONE of a number of child nodes (synonymous words) may trigger the activation of a parent node (synonymous group/ abstract type).

---

[10] Realistically in the short term such structures are to be created manually by the tutor, whilst in the latter part of this thesis we shall consider algorithms for the automatic identification of at least some of these macro-nodes.

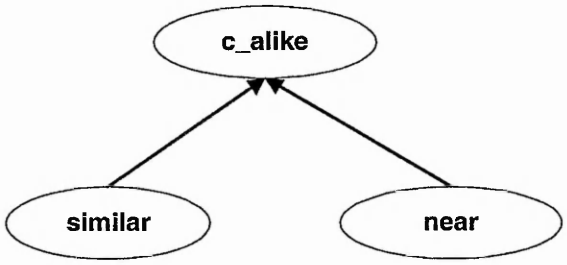


Figure 3-5 Abstraction

This the first level, abstraction correlates closely with the notion of synonymity. If abstraction is recursively applied it produces a deep type hierarchy that can be used to model less and less specific word categories. This is in many ways similar to the taxonomic classification of words to be found in Roget's Thesaurus.

3.1.2.2. Composition

Certain words have a composite sense which is distinct from the conjugate sense of their parts, for example idioms. (Not as much the whole is greater that the sum of its parts but that the whole is different.) This phenomenon may be modelled by a network where a single parent node requires ALL of its children nodes to become active before it itself activates.

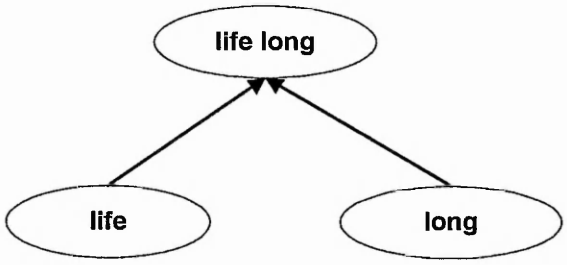


Figure 3-6 Composition

Also as has been seen earlier, this type of network structure adequately models facts, which as distinct from above represent a specific state of affairs as the conjunction of its

component parts. (You may note that in this primitive representation actor and recipient are not distinguished).

3.1.2.3. Decomposition

Some words have a composite structure where their parts contribute different senses to the meaning of the word as a whole. For example, at the morphological level consider the word walk-ed: 'walk' tells us the action, 'ed' tells us that it happened in the past. Similarly, at the semantic level, the word husband, tells us that the entity referenced is [male], [human] and [married]. Both these cases could be represented by a child node that passes its activation to several parent nodes.

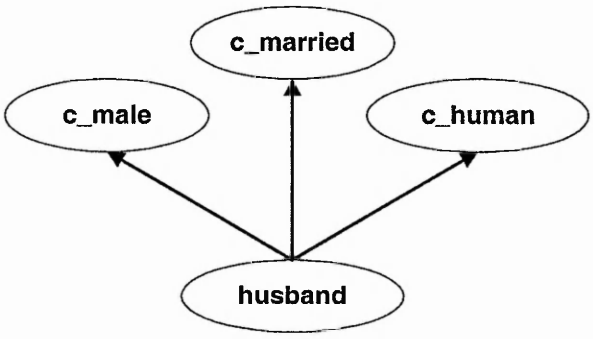


Figure 3-7 Decomposition

3.1.2.4. Parallelism

Closely associated with decomposition is the concept of parallelism whereby the separate aspects of a words meaning (modelled by spreading activation) may be computed concurrently.

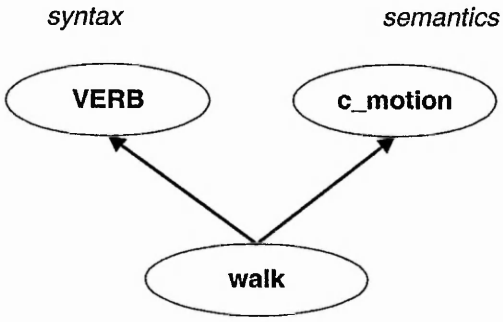


Figure 3-8 Parallelism

Note if desired this general principle may be used within the representation of the simple fact structure (see 3.1.2.2) in order to distinguish between actor and recipient when considering a typical binary predicate.

3.1.2.5. Context

There are many words that have multiple meanings. In a particular case it is only possible to identify which meaning is applicable by looking at the context. This is the linguistic phenomenon of polysemy or homonymy. We may model this in a network by a single child node linking to two parent nodes in parallel. Both of these parents will require input from a node representing the correct context also, before it, itself, becomes active. Take the classic example of 'bank', in a financial context it has a completely different meaning to a water/countryside context:

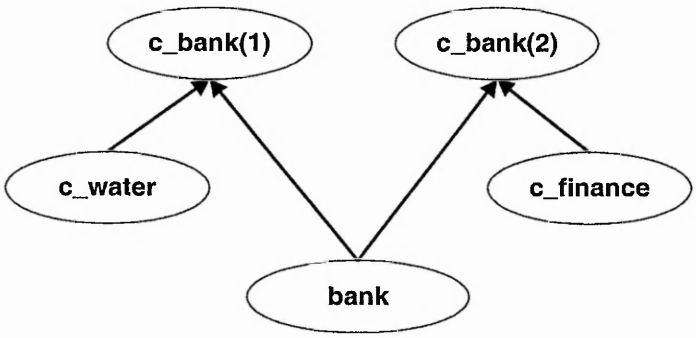


Figure 3-9 Context

3.1.2.6. Disambiguation – Mutual Inhibition

The mutual exclusivity of two hypotheses or perhaps two word meanings may also be modelled by an activation-passing network. Using the bank example again it will mean either a 'commercial building' or the 'side of a river' not both. Mutually inhibitory links between sibling nodes ensure that only the node receiving most supporting evidence remains active. These mutually inhibitory links are implemented with linkage possessing negative weightings.

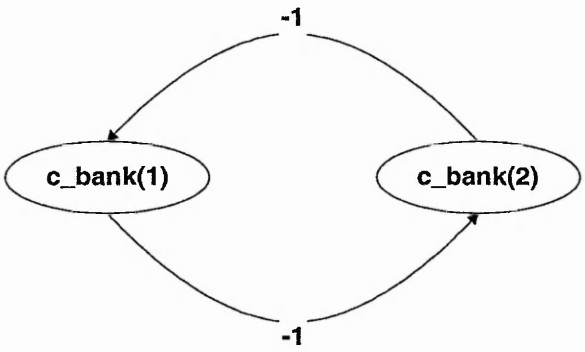


Figure 3-10 Disambiguation

3.1.2.7. Feedback

Sometimes the information necessary to disambiguate two hypotheses is only available at a higher level process. In order to make use of this information it is necessary to implement feedback links whereby a parent node can affect the activation of a child.

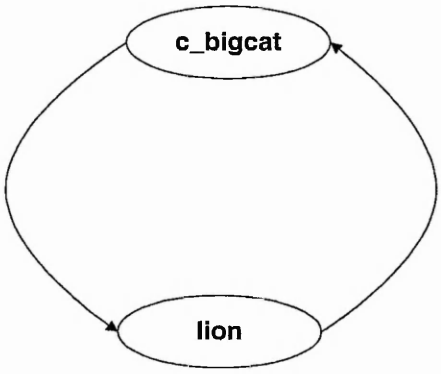


Figure 3-11 Feedback

3.1.2.8. Connectionist process

The computational properties that simple node and link constructs possess have been shown above. It is a small step to show that more complex node and linkage arrangements (with an appropriate activation function) can approximate to a feed forward connectionist network. As has been demonstrated in numerous connectionist papers a three layer activation passing network is capable of embodying an arbitrary mapping between any two sets of patterns provided there are enough hidden nodes

(Elisseeff and Paugam-Moisy 1997). Further, the Universal Approximation Theorem (Hornik, Stinchcombe and White 1989; Hornik 1993; Bishop 1995) demonstrates that a three layer network with one input, one output and sufficient nodes in the hidden layers can learn *any* function. This adds powerful computational power to our simple knowledge architecture. The following diagram 3-12 presents a typical topological arrangement of nodes that is capable of operating in a connectionist type manner.

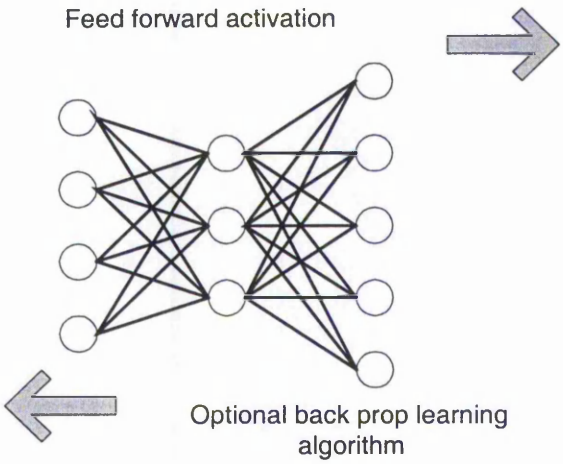


Figure 3-12 Connectionist Process

3.1.2.9. Logical Process

By virtue of the fact that all logical gates (OR, AND, NOT etc.) can be modelled by an interconnected network, and a standard vonNeuman architecture computer can be implemented entirely with such gates it should be possible to emulate any formal computation within the resource constraints.

The above points demonstrate that our chosen knowledge architecture has not only powerful representational flexibility but is capable of performing computation under both connectionist and symbolic processing paradigms.

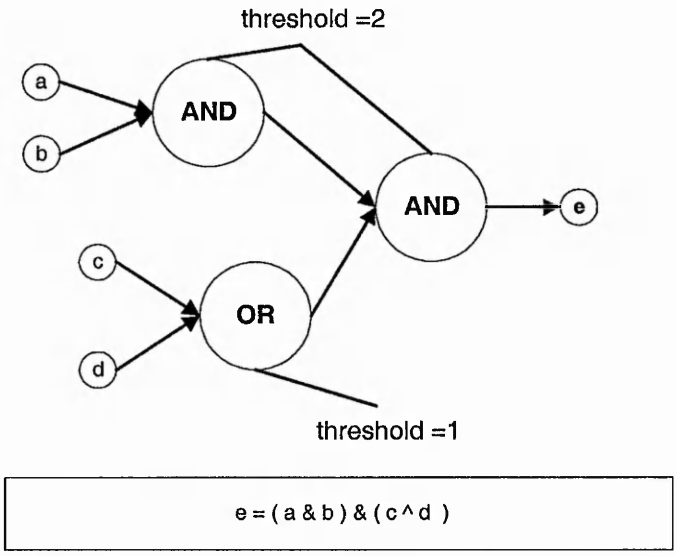


Figure 3-13 Logical Composition

3.1.3. Examples

To demonstrate how the above components can be bolted together into more complex functional units with interesting and useful computational and representational properties, three example models are presented.

3.1.3.1. Simple Phrase Modelling

Taking a phrase at random from the analysed student text, for example *"processed concurrently"*, the following demonstrates how a network could be used to capture some of the various ways in which this phrase may be expressed.

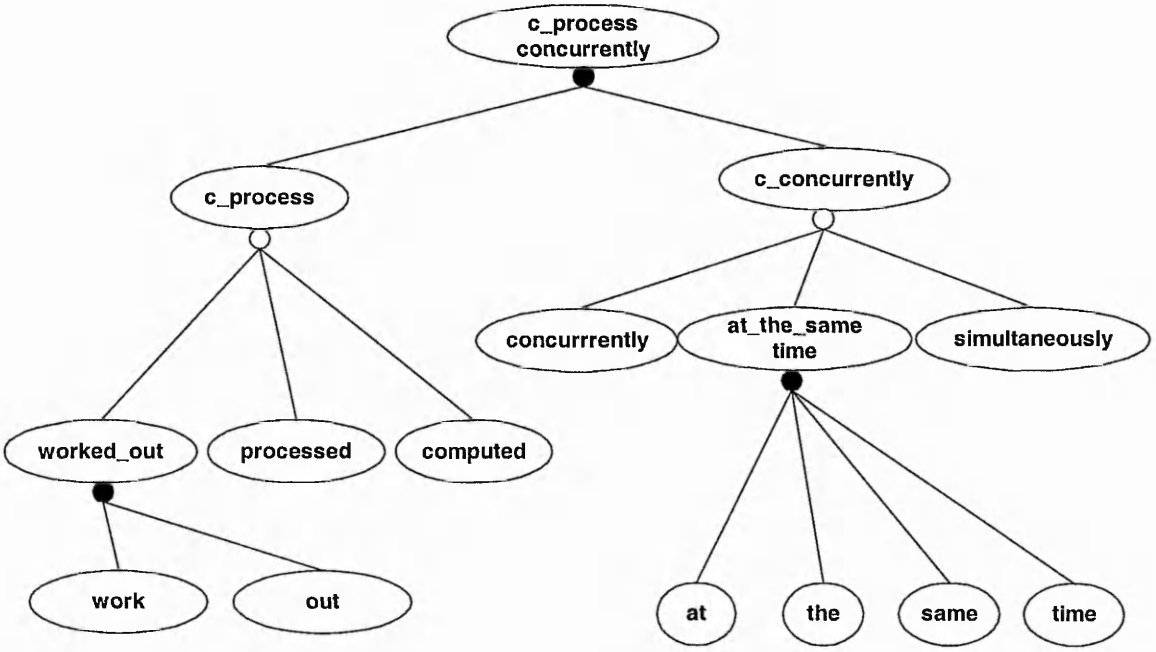


Figure 3-14 Phrase Modelling

In the above there are two broad synonymous groups, both modelled with abstracted nodes. The phrase itself is modelled as a compound node consisting of the two synonymous groups. The phrase "at the same time" is modelled as a composite node of the four individual words, as is the phrase "worked out". The phrase *work out* is interesting as there is potential ambiguity between this and the phrase *work out* meaning *exercise*. It would be possible to eliminate this potential ambiguity by using a combination of mutual inhibition and feedback from higher contextual elements.

The entire network, although crude, gives some idea of how multiple phrases may be mapped onto the same conceptual element.

3.1.3.2. Addition Mode

The following model demonstrates how a composite macro-node architecture can be used to check simple addition sums.

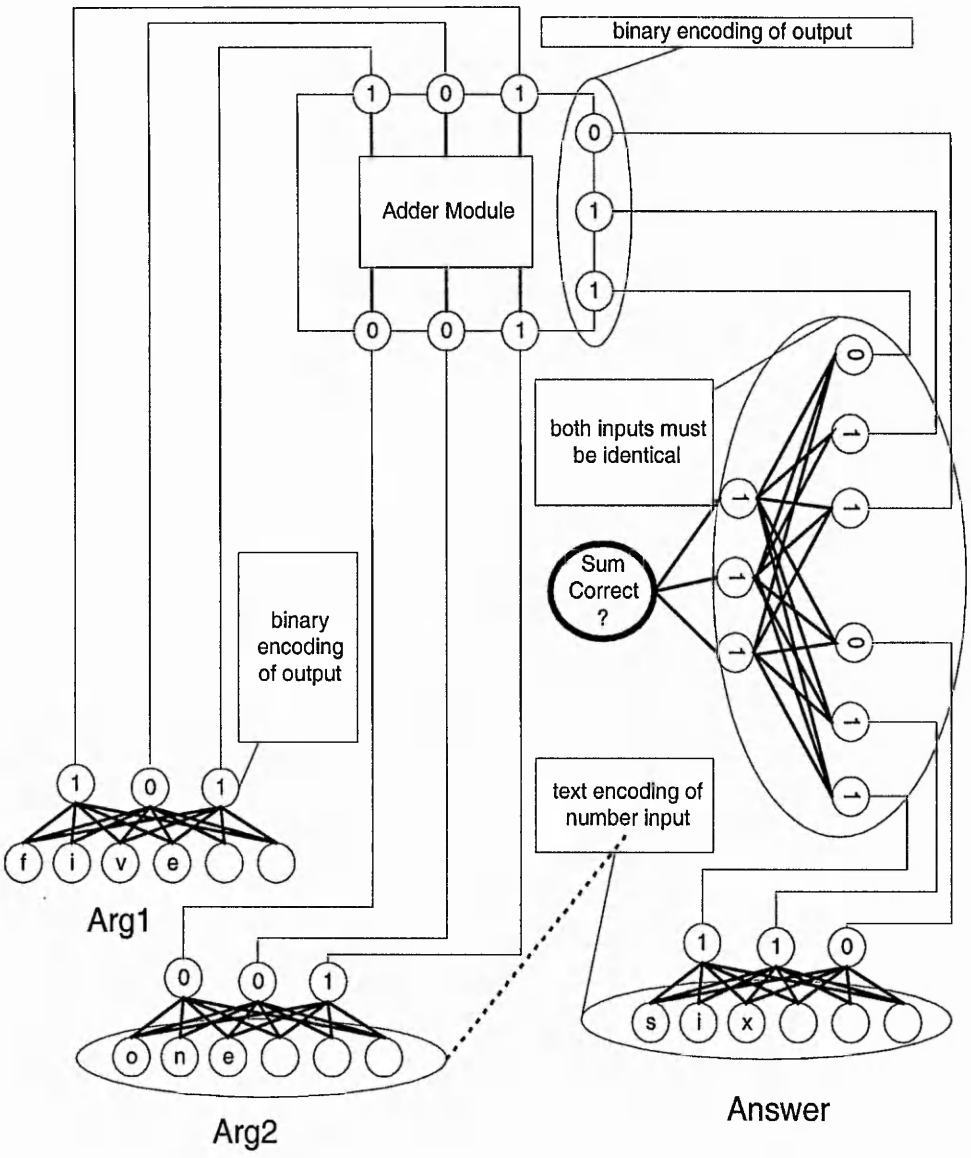


Figure 3-15 Adder Module

The above network is intended to validate (mark as correct/incorrect) statements of simple addition such as:

one plus one equals two	✔
three plus three equals seven	✘
4 plus 2 equals six	✔
2 plus 5 equals six	✘

For reasons of diagrammatic simplicity the macro node units required to validate the operator and equivalence sign (i.e. plus and equal) have been omitted, however it should be easy to see how these would be integrated into the system.

There are three inputs to the system (Arg1,Arg2,Answer) and a single output which validates "sum correct". The three inputs are encoded within the same form on a fixed string length encoding. If a variable activation node model is used which has *biased* inputs the characters may be encoded as partial activations on the node: (1/26 = 'a' 2/26 = b for example). This is the model demonstrated here, however the same is possible on a Boolean activation model. However more nodes have to be used for each character position and each character would have to be either *distributed* or *bucket* encoded.

Each input maps to a series of three nodes upon which the number is binary encoded (allowing the numbers 0-7 to be represented).[11] Two fully interconnected layers are used to perform this mapping, for it is considered a fairly simple mapping. If the mapping were to be more complex, for example if not only the text string "one" were to map to 001 but the text string "1" then hidden layers could be used. This would be necessary when the features in the input domain were non linearly separable (Rumelhart McClelland 1986).

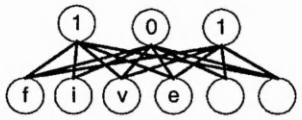


Figure 3-16 Text Recognition

The weightings of the networks necessary to perform this mapping could be learned by any of a number of learning algorithms (back propagation for example).

---

[11] Note of course that ANNs are finite; consequently they cannot model arbitrary numbers. Any modelled function must therefore be bounded.

The binary encoding resulting from the two arg values flow into the outputs of the *adder module* whilst the binary encoding of the result is indirectly validated against the output of the adder module.

The adder module is itself a network, with six input nodes and three output nodes. The internals are not represented in the above diagram since by treating it as a modular component it is possible to see two distinct implementations of the function.

Firstly it may be implemented as an adder circuit using nodes to replicate the functionality of AND and OR gates. Alternatively it may be implemented as a feed forward neural network where addition is solved by a simple pattern match process. Either way the end result is the same.

Using another network the output of the adder module can be compared against the binary encoded translation of the result string.

The end result is a network implemented purely from nodes and weights that is capable of evaluating sentences as above and judging them to be correct or incorrect. Further, a high degree of robustness is inherent within the system due to the network mapping between text string and binary encoding.

This is a network capable of representation and transforms between representations. Representations are chosen in accord with the functionality required and how it effects the desired process. It is capable of and uses both connectionist and logical processes.

3.1.3.3. Disambiguation Model

The third network demonstrates how the problem of bootstrapped contextual disambiguation may be solved with an appropriately configured network.

Take the phrase:

> *"get the train off the tracks"*

Let us concentrate on the two words: train and tracks. Further let us take just two senses of each of these words. The source used is Collins Concise Dictionary (1989)[12].

a) Train (n): a line of coaches or wagons coupled together and drawn by a railway locomotive.
b) Train (n): something drawn along, such as the long back section of a dress that trails across the floor.

a) Track (n): a rail or a pair of parallel rails on which a vehicle such as a locomotive runs.
b) Track (n): a course for running of racing.

If train and track are free to take on each of these senses there are four distinct possible interpretations of the above sentence.

The diagram below demonstrates a network with appropriate excitatory and inhibitory links that can bootstrap this disambiguation with no further information.

[12] Note in actual fact the above dictionary gives 20 distinct definitions for track and 12 definitions for train. This gives 240 possible interpretations of the sentence if only train and track are ambiguous. To complicate matters further the item "tracks" has four senses in its own right which are distinct from the pluralisation of the noun track.

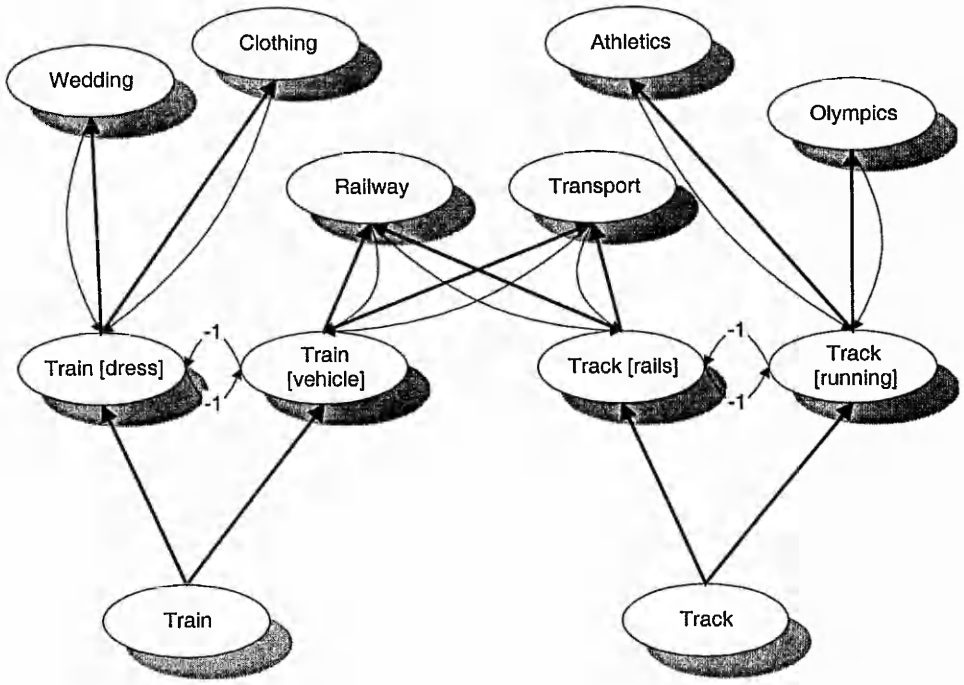


Figure 3-17 Disambiguation

The network is arranged hierarchically. Those items at the lowest level correspond to the lexical items 'train' and 'track' respectively. Both these items have excitatory links to both their respective senses. Each pair of senses are linked with mutually inhibitory weights to model the fact that a single lexical item can not simultaneously take on two senses. Each sense has excitatory links to its *decomposed* sense primitives. Each sense primitive has a mild excitatory feedback link to its child sense.

If train and track are simultaneously activated all four senses will receive equal activation. Because of the mutually inhibitory links, if either of a pair were slightly more active than the other, this differential would be exacerbated and that node would emerge as a definite winner. However, at this stage they are perfectly balanced. This activation will spread to the sense primitives and these primitives will feedback a proportion of this activation to their children nodes. Note that due to the semantic overlap the *railway* and *transport* primitives receive a greater input activation than the others. In turn, through the feedback links, the *Track[railway]* and *Train[vehicle]* sense node receive a disproportionately higher activation. Thus two clear winners will emerge.

3.1.4. Comments

In the previous section it has been shown how the primitive node/link constructs may be combined into specific structures of varying computational and representational power, all of which are relevant to representing language constructs. In the following section some more general comments are made on the nature of such networks and how they could potentially function with respect to the target domain of automated assessment task.

3.1.4.1. Understanding vs. Expectation Validation

It is our aim simply to discriminate between correct and incorrect responses to a question. To achieve this true *understanding* is not necessary. In this NLP problem where the context of the responses is highly constrained, it is possible to generate a strong expectation of what will be submitted. The underlying strategy then is to explicitly set down these expectations within the representational scheme offered by the chosen knowledge architecture. The submitted student sentence is then proffered as a set of evidence that either confirms or refutes this expectation.

The strategy then serves to focus attention on the pertinent information only. It carries with it, however, the disadvantage of being unable to perceive what it cannot expect.

3.1.4.2. Innate similarity of all knowledge representation schemes

The choice of a particular knowledge architecture is somewhat arbitrary. Knowledge bases are used to represent information. The architecture of a particular base is just more information. Over a particular level of sophistication most can be proven to be isomorphic with each other. The work of Schubert (1991) demonstrates this point very clearly, where he defined mappings between semantic nets and first order predicate logic in order to establish equivalence, and further propounds convergence of all the major KR schemes.

From a pragmatic point of view, the choice of architecture should be determined by two factors:

1. Simplicity for editing.
2. Appropriateness for problem domain.

The choice of representation can affect the ease of computation, for a particular problem within a particular problem domain. This is what is meant by appropriateness for problem domain.

To demonstrate, consider the problem of computing the rotation of an object around a particular origin in two dimensions. If the problem is represented in normal Cartesian co-ordinates the solution is a matrix multiplication:

$$\begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x' \\ y' \end{pmatrix}$$

On the other hand if the problem is formulated within a polar co-ordinate representation a rotation could be computed by simple addition, as follows:

$$(r, \phi + \theta) = r\phi'$$

Contrariwise, a simple linear translation would be extremely easy to compute using Cartesian co-ordinates but would be difficult using Polar co-ordinates.

The problem to be considered here is natural language processing. The contention is that many of the problems that natural language processing poses (such as representation of synonymity, disambiguation, bootstrapped processing, constraint satisfaction) are more appropriately solved by an activation passing network.

3.1.4.3. Knowledge Architecture Vs Knowledge Base

There is a distinction to be made between what a particular instance of a knowledge scheme can be used to represent and what the knowledge scheme itself (or the primitive architectural components that comprise that knowledge scheme) represents.

This distinction can be clarified by the use of two distinct terms: knowledge architecture and knowledge base. The knowledge architecture is the scheme itself; the knowledge base is a specific data realised within this architecture.

To illustrate, within predicate logic an atom and a predicate are distinguished architecturally, as are the slots and fillers within the frame formalism. Within logic, the distinction between a jaguar and a lion for instance would be made at the knowledge base level. However, the AND and OR operator are distinguished architecturally. Not so for a connectionist architecture, both AND and OR processes may be modelled by constructs built from the more primitive node, link and threshold, hence reside at the knowledge base level.

Implicit within the entire approach of this thesis is the working hypothesis that the less the information residing within the architecture of the knowledge base, the greater the flexibility of representation, and more specifically, the more powerful and more general the learning algorithms that can be applied.

3.1.4.4. Relation - Object Distinction

Careful consideration must be taken of the respective epistemological status of the knowledge architecture and the knowledge base. Common within many knowledge formalisms is an implicit Relation - Object distinction at the architectural level. This is undesirable for our purposes for the following reasons.

Firstly, it is difficult to express knowledge about relationships in the same way that we express knowledge about objects.

Similarly, complications arise if we try to represent knowledge about facts. With an implicit object-relation distinction a fact becomes a relation between n-objects. To then represent knowledge about facts we either give our (relation, object, object,..) the status of an object, or introduce a new entity type - a fact - and enhance the syntax (rules for combing entities) to allow facts into the (relation, object, object,..) construct. Either way the structure becomes unnecessarily complex.

Thirdly, we may wish to make a type/token distinction between relationships as we almost certainly do with objects. By removing the distinction we acquire a greater consistency.

By such a move nothing need be lost. Should it become essential to make such a distinction it is always possible to make this at the knowledge base level as opposed to the architectural.

3.1.4.5. Data Vs Process

In the above examples of knowledge representation schemes, the connectionist model is the only one capable of any form of process or reasoning. All the others are capable of the static representation of information only; a reasoning engine needs to be applied in order to make use of any of this information.

Within the connectionist model there is no real distinction made between data and process. It is probably no coincidence that the connectionist model is the best model for the application of generalised learning algorithms.

3.1.4.6. Segmentation of KR and NLP processes

Knowledge Representation and Natural Language Processing are traditionally thought of as two distinct but closely associated disciplines. In this application, as in others, they prove difficult to separate. Not only is the natural script being used to build and test the knowledge structures, but the language processing, itself, is a knowledge intensive process.

3.1.5. Legal Metaphor

To give a brief overview of the workings of the system, the overall technique is analogous to the workings within a court of law. The defendant is deemed innocent until proven guilty as the sentence is deemed incorrect until proven correct. The lawyer may put forward any number of hypotheses as to what the true sequence of events was on a particular day in the same way as our structure maps out the conjunctions of groups etc. that would indicate that the sentence is true. If enough mutually corroborative evidence is presented for a particular hypothesis the defendant/sentence is judged guilty/correct.

3.1.6. Mathematical Formalisation

To clarify the form of the proposed knowledge schema and to establish a formalism upon which the description of proposed algorithms can be based an attempt will be made to describe the knowledge schema mathematically.

A particular knowledge base can be characterised by a n-tuple K

$$K =< C, p, t, \Sigma, e >$$

where briefly

C	set of concepts
p	function that returns parents of a concept
t	function that returns the threshold value of a concept
Σ	the set of possible strings
e	function that returns the evidence strings of a concept

and from which we can further define

P(C)	power set of concepts
P(Σ)	the power set of strings

and more fully

C is a set of concepts and n is the number of concepts in C.

$$C = \{c_1, c_2, c_3, \ldots, c_n\}$$

and P(C) is the power set (the set of all sets of) C

$$C = \{\{\}\{c_1\}, \{c_1, c_2\}, \{c_2\}, \ldots\}$$

p is function that returns parents of a concept the range of which defined on the power set of C and the domain of which is defined on C.

$$p : C \rightarrow P(C)$$

t is function that returns threshold of a concept; the range of which defined on the set of real numbers and the domain of which is defined on C.

$$t : C \rightarrow \Re$$

Σ is a finite set of string (say taken from a dictionary) where m is the number of strings.

$$\Sigma = \{\sigma_1, \sigma_2, \sigma_3, \ldots, \sigma_m\}$$

and P(Σ) is the power set (the set of all sets of) Σ

$$\Sigma = \{\{\}\{\sigma_1\}, \{\sigma_1, \sigma_2\}, \{\sigma_2\}, \ldots\}$$

e is a function that returns the evidence list for a concept; the range of which is defined on the power set of Σ and the domain of which is defined on C.

$$e : C \rightarrow P(\Sigma)$$

Of use also is the function r which returns the children of a particular node and like the parent function the range of which is defined on the power set of C and the domain of which is defined on C. This need not be represented explicitly as it is redundant information. (However it is to be found in the implementation as redundant data for performance reasons.) So r is

$$r : C \rightarrow P(C)$$

and defined in terms of p as:

$$r(x) = \bigcup_{i=0}^{i=n} \begin{Bmatrix} c_i \text{ iff } p(c_i) = x \\ \varnothing \text{ iff } p(c_i) \neq x \end{Bmatrix}$$

where obviously x∈ C.

Further an activation function a is defined for each node c. Again the domain of this function is defined on the set of real numbers. Thus this activation function can be defined in abstract as:

$$a : C \rightarrow \Re$$

But specifically in the simple model discussed above where activation is Boolean and all weights are unitary, activation can be defined recursively as:

$$a(c) = \begin{Bmatrix} 1, \sum_{x \in h(c)} a(x) \geq t \\ 0, \sum_{x \in h(c)} a(x) < t \end{Bmatrix}$$

Using this formalism to firm up the definition of some of the nodes discussed above.

An *evidence* node has a non empty evidence list

$$|e(c)| \neq \varnothing$$

A *compound* node has a threshold value equal to the sum of its children.

$$|r(c)| = t(c)$$

An abstract node has a non zero threshold value less than the sum of its children.

$$0 < |r(c)| < t(c)$$

This concludes the discussion and the definition of the knowledge architecture to be developed. Experiments that utilise and test the effectiveness of this architecture can be found in the first section of Chapter 5.

3.2. Learning Algorithms

3.2.1. Introduction

In the previous section knowledge representation, in general, has been considered and a specific schema has been detailed which is capable of representing information at many levels in the knowledge hierarchy: both at the abstract "fact" level and the shallow lexical level. It is hoped that that this knowledge structure shall be capable of embodying information in a sufficiently expressive and flexible manner to discriminate to a reasonable degree of accuracy between correct and incorrect student responses. This shall be the focus of subsequent experiments

Considering now the pragmatics of automated assessment it is essential that the issue of knowledge base creation be addressed. This has not been considered yet, as it is anticipated (in the initial stages at least) that this should be a manual process (with a few automated aids, such as various statistical tools and an integrated Rogets Thesaurus). But in the main a great deal of judgement is required by the tutor in order to construct these "*sufficiently expressive*" knowledge structures.

It is a natural extension of the work undertaken so far to consider how these structures may be created automatically. The hypothesis is that there will be statistical regularities within the student-generated responses that may be utilised in order to construct useful elements of the knowledge base.

Within this chapter the theoretical foundations for the implementation of two such algorithms will be laid down (in fact as we shall see later rather than two distinct algorithms they can be seen as two complementary aspects of the same algorithm.) Later, assuming that such structures can be created, we must consider how we can evaluate the usefulness of these structures. In order to do this we will introduce the subject of Latent Semantic Analysis, a complex mathematical technique, which is capable of automating

the text evaluation procedure and so apply the produced networks to the target application of automated assessment.

3.2.2. Terminology

In order to clarify the description of the networks and later the experiments to be performed on these networks, let us introduce some new terminology. Firstly, we have:

Activation Passing Networks (APN) – this is a general term used to encompass the networks described above, and relates specifically to the fact that the individual nodes that comprise the network are capable of passing activation to one another.

These networks can be subdivided using two terms that relate not to the functionality of the network, but to its origins and specifically the manner in which it was created.

Hand-Crafted Network (HCN) – these are networks that have been produced by hand (using the tools available in the graphical user interface) therefore their construction is labour intensive.

Algorithmically Generated Networks (AGN) – these are networks that have been produced automatically by particular algorithms. The only human intervention in the generation of these networks is the selection of the data to which the algorithms are exposed and the configuration of a variety of algorithmic parameters.

Finally, we introduce a new term that relates not to the networks but to the function to which the networks (specifically the AGNs) are put.

Perceptual Augmentation Process (PAP) – this describes a function which takes input data in an initial form and translates it to a new augmented form, where it is implied that this new form offers certain advantages, for example allows the data to be more effectively or efficiently processed, with reference to a specific computational function. In the cases to be considered in this thesis this computational function is that of automated assessment. Generally a Perceptual Augmentation Process adds information.

3.2.3. Form of Automatically Produced Knowledge Structures

First we must consider what type of knowledge structures to produce.

Within the handcrafted, activation passing networks, previously defined, two distinct macro-node types (as shall be shown in subsequent experiments) are particularly useful. These are the composition structure and the clustering or abstracting structure. The compositioning and clustering structures are symmetric operations for the former is the manner in which raw-evidence is generalised, and the latter is the manner in which raw data is specialised.

It is intended that the networks to be produced will consist entirely of these nodetypes. Also, the networks will be active knowledge structures in that, due to their activation passing capability, they will be capable of limited functionality. In this sense they are similar to the hand-crafted knowledge structures produced within the previous chapter. They will differ in the extent and purpose of this enabled functionality. Specifically the handcrafted networks, by virtue of their topological arrangement and activation passing capability, are capable of functionally discriminating between correct and incorrect student responses (within the empirically derived tolerance levels). These new networks are to be generated from unsupervised[13] learning algorithms. As such they are unlikely to be capable of such extreme function. Rather, it is anticipated that the grown network will uncover regularities within the problem domain. The complete network will therefore provide a perceptual function, which adds new analysis granularities to the raw atomic data. This theme will be expanded upon later.

3.2.3.1. Compositioning and Clustering – Why?

What is the justification for attempting to construct a network out of only clustering and compositioning nodes? The justification is threefold:

1. Networks – it will be shown that: the handcrafted networks adequately perform the correct/incorrect discrimination function. Given that these networks consist largely of these node-types, we have empirical justification for using compositioning and clustering as building blocks.
2. Neural Plausibility – this architecture has a high degree of neural plausibility. The entire architecture is connectionist inspired. It is quite easy to see how real nerves could be arranged in these same collaborating structures.

---

[13] The reason for the choice of developing unsupervised learning algorithms is quite simple: it is the intention to develop an automated assessment system that should be "potentially" viable in a real world environment. If the algorithms to be developed were supervised this would presume that the answers were pre-marked by an external entity (tutor). This somewhat reduces its real-world viability.

3. Language Domain – as discussed in the previous section, these structures have been designed to exhibit functionality that is particularly suitable to the problems within the natural language processing domain.
4. Tenability of learning algorithms – although somewhat a backward argument, a network consisting of composite and clustered nodes is justified by the fact it is possible to construct an algorithms which can construct such networks.

Why must the generation of such networks be integrated into the single algorithm?

This is a more subtle issue. To elucidate, let us first review the merits of clustering and compositioning one at a time. In isolation the merits of clustering input into higher order structures are fairly clear. Firstly, it is a form of lossy compression; a way of representing the same information in a lower dimensional construct. Secondly it is a mechanism for identifying families of atomic data; further these families may be recursively defined from previously identified clusters and composites. Thirdly, it is a way of representing abstracted data against which further information can be stored. This means, if three different atomic units share the same property, an abstracted node may represent all three of these units and the property may be marked up against the abstracted node rather than the three individual units. This is also a form of compression, but this time a lossless form.

Similarly, the merits of compositioning may be considered in isolation. In a basic sense it addresses the issue of the representation of context. It is a way in which two individual, but co-occurring units of atomic data can be considered as a whole, which compensates for errors that are introduced from an overly reductive analysis of incoming data. This is another way of saying that it is possible to capture information in transition matrix of bi-grams that is not available in a transition matrix of uni-grams. This is because the behaviour or distribution of elements of length two is not predictable from the bi-grams constituent uni-grams. The uni-gram analysis is overly reductive.

Combining the two node-types into the same integrated network adds representational and computational richness to the network. In an integrated network it is possible to consider, not only clusters of composites and composites of clusters but composites of clusters of composites, etc. etc. Its constituent parts and its relationship to those parts define every new node, which is added to a network. These parts, or nodes are the language in which new nodes are phrased. When the language is richer it is possible to say things that were not possible before.

Combining the two processes into the same algorithm solves the pragmatic problem of avoiding a functional stratification of the network. To explain: the algorithms to be considered address the issue of network growth. If one algorithm were to be applied and then the other, we would get networks that consist of a clustered stratum then a composite stratum. This may not be the most natural representation of the problem domain. The intended strategy is to interleave the processes so that compositioning or clustering may be applied at any point in the network at the most opportune time.

3.2.3.2. An Evolved Perceptual Framework

The unifying vision behind the creation of these networks and the growth algorithms is that of a naturally evolved perceptual framework. Starting from a raw atomic, reductive description of a problem domain, a process will be created, which by analysing corpora of data, will identify high order constructs and therefore provide a richer description of data. The networks shall become a perceptual augmentation process. The activation passing properties of the network make it a cohesive functional process in its own right that is capable of accepting input information in the raw atomic form. The output data consists of all activated nodes. These should be the high order constructs that have been "perceived" within the input string.

3.2.3.3. A Note on Computational Complexity

This is a short note on the pragmatics of generating such networks. The algorithms to be developed are passive and unsupervised. New nodes will be identified and added to the network whenever a compositioning or clustering statistical criteria is satisfied. The compositioning and clustering relationship may be held between any identified node, therefore as each node is added it becomes a new node between which information must be stored, and against which relationships must be identified. The performance of the network growth algorithms is likely to decay exponentially with the size of the network.

This, unfortunately, is the corollary to the neural plausibility of the system. Neurally inspired algorithms tend to have a native parallel implementation. Implementing such processes on a serial machine usually results in exponential decay in performance.

The implication of this for the implementation is that, firstly, all algorithms will have to be highly optimised, but more importantly that the scale and scope of investigation will have to be limited to a low dimensional domain.

3.2.4. Putting Clustering/Compositioning in Context

To place this work in context, brief mention of the clustering and compositioning works that are available in the computational linguistics literature is needed. Although they are rarely phrased as such there are close parallels to both processes. Compositioning is extremely similar to N-gram analysis, Markov transition matrix analysis and collocation analysis. Whereas, on the clustering side there are various works, mostly in the field of syntactical analysis, that look at recursive word clustering. The algorithms to be considered in this thesis differ in that:

1. The two process, clustering and composistioning, are combined within a single algorithm
2. The output: rather than a matrix, a list of N-grams or a dendrogram (as would be produced by Markov, N-gram and hierarchical clustering analysis, respectively) will be an active, information rich network from which much the same information can be derived.
3. The granularity of analysis is dynamically and locally determined.

The following is not intended as a full literature survey of these subjects, for this would not only be impossible, but would not be strictly relevant. It is sufficient merely to draw the necessary parallels that will help give an intuitive understanding of how the clustering and compositioning algorithms work.

3.2.4.1. N-gram, Markov Transition Matrix

N-gram analysis, Markov transition matrices and collocation analysis are all ways of analysing the statistical regularities to be found in the problem domain. Within Natural Language Processing these techniques have been used at virtually all levels of granularity: orthographic regularities for text recognition, phonetic regularities for speech recognition, syntactic regularities as feedback to both of the above, and word level regularities through collocation analysis.

The principle behind all these techniques is that the statistical distribution of the distinct units of analysis is non-uniform. This non uniform distribution may then be utilised to enhance the decision procedure of some probabilistic task – such as word recognition.

It is the norm in all these techniques to have a fixed granularity of analysis. For example an individual Markov model will be produced for fixed length entities within the problem domain, for example a unigram matrix, a bigram matrix or a trigram matrix. Note, due to the increasing combinatorial complexity as we progress from unigram to trigram the

matrix itself gets exponentially bigger. This granularity is generally picked before the analysis takes place and is constant across the entire analysis.

3.2.4.2. Recursive Clustering Algorithms

There are two highly relevant pieces of research in this area. Both originating from two similar but distinct PhD theses (Finch 1993, Hughes 1992). The first by Finch is effectively summarised in the papers Finch & Chater (1992a) Finch & Chater (1992b) Finch & Chater (1992c). Whilst the second by Hughes can be found in Hughes (1992) Hughes and Atwell (1993) Hughes and Atwell (1994). Both concern themselves with the problem of automatically determining syntactic categories from corpus data.

The broad approach in both is similar. Starting from an arbitrary corpus, identify a subset of words that are to be analysed. Each of these focal words is then characterised in an N-dimensional vector. This characterisation vector is derived from a statistical analysis of the corpus. What each dimension actually represents varies. However usually it relates to the distribution of other words (or more usually word groups) in relation to the focal word. For practical reasons of computational analysis the size of the characterisation vector is limited to a reasonable value.

Once each word is plotted in N-dimensional space it is necessary to develop some formal measure of similarity. A variety of measures are possible: simple Euclidean distance, angle between normalised characterisation vectors, Manhattan Metric, Spearman's Rank Correlation Coefficient etc. Specific definitions of these will be found later.

Recursive clustering then proceeds as follows, a procedure first defined by Sokal & Sneath (1963):

1) Place each item in its own cluster
2) WHILE there is more than one cluster remaining
3) Find the two closest clusters.
4) Create a new cluster
5) Delete the two originating clusters
6) END

Left to its own devices this algorithm would keep clustering till just a single cluster is left.

A hierarchical taxonomy is produced which is called a dendrogram. This tree recursively decomposes (or constructs, depending upon whether you are starting from the tree's root

or a leaf) from the single top-level abstraction into the actually occurring words. In many cases the clustering procedure is halted at a predetermined "alpha-cut" level to stop the single rooted tree occurring.

For the specific application that was considered in both these theses, which was syntactic classification, the clusters that were produced at several levels within the classification hierarchy were compared with the orthodox grammatical classifications. Both applications seemed to meet with a fair degree of success in that this empirically derived classifications compared well with the orthodox syntactic classifications.

3.2.5. Compositioning Algorithm

We now define the first half of the algorithm to be developed.

3.2.5.1. Introduction

This section describes a formal symbolic algorithm for producing context rich networks by processing corpora of linear strings of discrete units, identifying from the statistical co-occurrence of elements' probable compound units and the transition probabilities between the same. The networks produced this way are essentially connectionist in nature, in that the network comprises multiple inter-linked nodes each of which has a distinct activation level. However the individual nodes are symbolic in that each node represents unique discrete phenomena within the problem domain. The produced networks are deeply structured embodying useful contextual information from the problem domain, which is ideal for applying to recognition type tasks.

The outlined process closely parallels other statistical techniques that infer transition probabilities from statistical co-occurrence such as Markov models or N-Gram analysis, Brown (1982), Riseman & Hanson (1974) Jelinek et al (1983). With these techniques, however, when an attempt is made to analyze context over a wide field (ie. large N) the size of the corpus necessary to provide a good esimate of the transition probability rises exponentially, Keenan (1992). This problem is attributable to the need to specify N globally. The tree producing algorithm outlined here, by the use of some modifiable heuristics, estimates a unique value of N dynamically and locally. That is, we estimate the transition probabilities to and from a node if, and only if, our heuristic identifies the node as a suitable candidate.

3.2.5.2. Description of Algorithm

A standard statistical procedure attempts to identify a relationship between N variables. The algorithm outlined here does this with a set of primitive nodes, however when a

strong relationship is found between two or more nodes they are concatenated into a new node and the procedure will then look for relationships with this node also. The number of variables is therefore constantly growing.

Relationships between nodes are identified using a context history which records the context for each node. Conversely a node can be defined as something for which a context history is recorded. Take the example where we are analysing written language, we will start with a set of primitive nodes (in this example the 26 primitive graphemes) for which we will record the context history. When, for example, the letter 'h' has been found in the context of 't' many times 'th' becomes a node in itself, and therefore against which context is now to be recorded. After some time we may identify 'e' as a candidate link for creating the 'the' node. The nature and depth of the produced networks may be modified through the use of two re-definable heuristics: the linkage heuristic and the compositing heuristic.

In outline, a new unconnected primitive node is instantiated for each primitive unit encountered in a string. Whenever this node is subsequently encountered within a string this node is made salient, and remains salient for a specified lifespan. That is, instance information is recorded temporarily for it on a list. The salient list is monitored at all times, so should two nodes be discovered which satisfy the linkage heuristic the appropriate context history is immediately attached to the concerned nodes.

When the link/context history combining two (or more) nodes exceeds a value specified by the compositing heuristic a new compound unit is instantiated that describes the conjunction of the children nodes.

Certain metrics are attached to the various entities in the tree. It is with respect to these metrics that the compositing and linkage heuristic are defined. Of significance is the measure of frequency (attached to nodes, links and global measures). An absolute measure of frequency would be inadequate for inter-node comparisons as new nodes are being created all the time and the new nodes are unaware of how often they occurred before they were created. More suitable would be a measure of acquisition velocity, that is to take an estimate of the differential of frequency over time. It was found that the regular resetting of all frequency values served as a satisfactory approximation to this.

The algorithm as discussed may be defined in pseudo code as follows:

```
LOOP (for all strings)
      {
```

```
LOOP (for all units)
        {
        IF (node not previously seen)
                {
                DO Instantiate_Node
                }
        ELSE (node already seen)
                {
                DO Activate_This_Node
                DO Pass_Activation_To_Parents
                DO Make_Node_Salient
                LOOP (all salient nodes)
                        {
                        IF (Nodes match LINKAGE HEURISTIC)
                                {
                                DO Add_Link_To_History
                                IF (Link > THRESHOLD HEURISTIC)
                                        {
                                        DO Make_Nodes_Into_Compound
                                        DO Link_Children_To_Parent
                                        }
                                }
                        }
                }
        DO Delete_Salient_Nodes
        }
}
```

where Pass_Activation_To_Parents is further recursively defined as

```
Pass_Activation_To_Parents
{
        if (Incoming_Activation_Reaches_Threshold)
        {
                DO Activate_This_Node
                DO Pass_Activation_To_Parents
                DO Make_Node_Salient
        }
}
```

The nature of the parent or compound nodes needs some clarification. Firstly, note that a parent node receives activation from all its child nodes. The parent itself though, will not

become active until it receives activation from all child nodes such that the combination satisfies the linkage heuristic. The linkage heuristic is therefore defining the nature of the network e.g. the conditions that two nodes must satisfy for the two to be considered a compound node.

Secondly we need to make a distinction between the type and token instance of a compound node. This is because unlike primitive nodes a compound can exist in overlapping positions within a string. Take the example of 'banana': the composite 'ana' exists twice in non-distinct positions.

3.2.5.3. Saliency

The distinction between the type and token instance of a node relates to the principle of saliency mentioned earlier, this probably needs a few words of explanation.

Within a standard neural net architecture it is difficult to pass on the positional information of a node explicitly within the activation it passes to its linked nodes. This is because an individual node is not computationally powerful enough to discriminate its input and effect the functional changes that a new node position would imply. It would be easier to pass on the positional information implicitly by only letting nodes in position 1 pass on their activation to nodes in position 2 etc., however the number of nodes necessary to encode the input string and model the relationship between composites in their various positions would rise exponentially with the string length.

The network discussed in the previous section is in essence only recording the relationship between nodes of a particular type. However in order to both build and use the net we need to be able to identify instances of a node within an input string and be able to propagate this instance information through the network. The principle of saliency is a mechanism by which this information may be stored and passed throughout the network. The type of instance information that is stored with the node identifier on the salient list is dependent upon what is required by the linkage heuristic. In most simple cases this will simply be the position of the node within the string.

Saliency is in many ways a type of working symbolic memory. This contrasts interestingly with the essentially connectionist network. This is not a contrived distinction but one that appears essential to handle the highly time dependant input data.

This whole issue of handling time dependant data becomes quite complicated. For the purposes of this application the simple concept of saliency proves sufficient. For a more in

depth discussion of both the nature of the problems and postulated solutions see Allott et al (1997b) and Allott et al (1997c).

3.2.5.4. Modifying the Network

As mentioned the nature and quality of the network is modifiable through two heuristics. To illustrate this consider the following examples. Firstly, the linkage heuristic which affects the quality of the network. In the simplest cases the metrics passed to salient nodes on instantiation will be node identifier and position found in the string. From this we could define linkage as simple left or right adjacency giving immediate left or right context. Alternatively we could extend this to give context within a pre-specified window i.e. found within 5 units.

Secondly, the compositing heuristic, which determines the depth. The simplest definition is a measure of the absolute frequency for a node divided by the total number of primitive units encountered. The next probable extension to this would be to modify this value with respect of the length of the compound, on the legitimate assumption that smaller compounds are likely to occur more frequently. This will stop trees becoming bottom heavy.

The second level of sophistication is to introduce a measure of a node's suitability for a task into the network modification procedure. When the information necessary to do this is available at node creation this measure may be readily incorporated into the linkage heuristic. However, this is not always practical. Take the recognition task: a good measure of a node's usefulness is whether the distribution of transition probabilities to new nodes is flat or not. A flat distribution can be used to no predictive effect. This information is available only after the node has existed for some time. In such cases we should consider tree generation as a two stage process. The first to generate a bushy, deep tree on pure statistical grounds, a second to prune the tree on functional grounds.

3.2.5.5. Formal Specification of Data Structure and Heuristics

A particular network can be characterised by a 6-tuple

$$\langle P, Q, h, p, t, ev \rangle$$

Where:

$$P = \{p_0, p_1, p_2, \ldots, p_n\}$$

is the initial set of primitive nodes and *n(P)* is the number of primitive nodes.

$$Q = \{q_0, q_1, q_2, \ldots, q_n\}$$

is the set of all newly identified and *n(Q)* is the number of new nodes. *P(Q)* is the power set of all newly identified (i.e. non-primitive) nodes.

From this we can define

$$I = P \cup Q,$$

that is I is the set of all nodes. Where *n(I)* is number of all nodes and so:

$$n(I) = n(P) + n(Q).$$

Further *P(I)* is the power set of all nodes.

In order to model context histories the h function is used:

$$h : I, I \rightarrow N$$

That is a mapping is defined between pairs of nodes and the natural numbers.

In order to model parental linkages the p function is used, where

$$p : I \rightarrow P(Q).$$

Which defines a mapping from each node to the power-set of all non-primitive nodes. Note, that a parental link may not be created to a primitive node, but may be created from a primitive node.

From this it is possible to define a child linkage function c, where

$$c : Q \rightarrow I$$

And specifically:

$$c(q_i) = \bigcup_{q_x \in Q, q_i \in p(q_x)} q_x \quad .$$

The number of children of a node therefore being $n(c(q_x))$.

Activation is a dynamic property of the network and can be modelled by a simple function

$$act : I \rightarrow [0,1]$$

mapping each node to number between zero and one in the general case and

$$act : I \rightarrow \{0,1\}$$

for the simple case where nodes take on only Boolean activation values.

We must now further define

$$A = \{a_0, a_1, a_2, ..., a_n\}$$

as the alphabet of simple words that we are to consider and that are present in our corpus. P(A) it follows is the power set of all strings.

An individual presentation of a sentence is modelled by a stream operator Ψ. Where

$$\Psi : N \rightarrow A,$$

that is Ψ maps natural numbers to individual words. $n(\Psi)$ is therefore the size of the string. It follows therefore that

$$\Psi(n) = \varnothing, \text{ where } n \leq 0 \text{ or } n > n(\Psi).$$

Primitive nodes differ from composite or clustered nodes in that they are activated if their "evidence" is found within the input string. This "evidence" is modelled by a function

$$\text{ev} : \text{P} \rightarrow \text{P(A)}.$$

The activation of primitive nodes can be described procedurally as follows

```
for (i=1 to n(Ψ) )
{
    for ( p_i ∈ P )
    {
        if ( Ψ(i) ∈ ev(p_i) ) then
            act(p_i) = 1
    }
}
```

In order to consider the activation of a composite node we must first describe incoming activation, this is generally described by the function:

$$in : Q \rightarrow R.$$

which maps all nodes to a real number. Specifically it is defined as:

$$in(q_x) = \sum_{y \in c(q_x)} act(y)$$

All threshold functions are generically described as

$$g : R, Q \rightarrow R$$

which simply maps a real and node onto a real. A set of threshold functions is defined

$$G = \{g_0, g_1, g_2, \ldots, g_n\}$$

in general, where the form of g is essentially defining the type of the node. In the specific implementation we are considering here there are only two threshold functions; the threshold function for composite nodes and the threshold function for clustering nodes. It follows then that here: $G = \{g_{composite}, g_{cluster}\}$

The t function maps a node onto a specific threshold function,

$$t : Q \rightarrow G.$$

This makes the generic definition of the activation of all non primitive nodes

$$act(q_x) = t(q_x)(in(q_x), q_x).$$

For the specific case of the compositioning node the threshold function is defined as:

$$g_{composite}(x, q_y) = \begin{Bmatrix} 1, x \geq c(q_y) \\ 0, x < c(q_y) \end{Bmatrix}$$

Before we can consider how to define the linkage heuristics we must model the principle of "saliency". Saliency is a set of 3-tuples.

$$S = \{s_0, s_1, s_2, \ldots, s_n\}$$

where each

$$s_i = \langle node, start, stop \rangle \text{ and } node \in I, start \in N, stop \in N.$$

As each node becomes active a new node is added to the salient list i.e. $S = S \cup s_i$.

The form of this tuple for a primitive node p_i is trivial:

$$s_i = \langle p_i, t, t \rangle,$$

where t is a time variable, which is incremented as the elements of Ψ are iterated over.

For a non-primitve node q_i it is a little more complicated:

$$s_i = \langle q_i, start, stop \rangle,$$

where *start* is the time, t when the node received its first contributing activation and *stop* is the time t when it received its last.

We are now in a position where we may define some possible linkage heuristics. Remember this is the heuristic that must be satisfied in order for a context history to be recorded against a node. A link heuristic maps from two salient tuples to a Boolean:

$$link : S, S \rightarrow \{0,1\}$$

Specifically simple right adjacency becomes:

$$link(s_x, s_y) = \begin{Bmatrix} 1, stop(s_x) = start(s_y) + 1 \\ 0, stop(s_x) \neq start(s_y) + 1 \end{Bmatrix}$$

And left adjacency

$$link(s_x, s_y) = \begin{Bmatrix} 1, stop(s_x) = start(s_y) - 1 \\ 0, stop(s_x) \neq start(s_y) - 1 \end{Bmatrix}$$

A windowed left right adjacency may be

$$link(s_x, s_y) = \begin{Bmatrix} 1, \left|stop(s_x) - start(s_y)\right| < 3 \vee \left|stop(s_y) - start(s_x)\right| < 3 \\ 0, otherwise \end{Bmatrix}$$

In order to define the compositioning heuristic two further measures will be required. The first is a measure of how frequently a node has been seen within the corpus. This is a simple function mapping a node to a natural number:

$$freq : I \rightarrow N.$$

And specifically for a node i_x:

$$freq(i_x) = \sum_{i_y \in P} p(i_x, i_y)$$

The second is a global measure of how many primitive units have been presented to the system.

$$Global = \sum_{i_x \in P} freq(i_x)$$

Compositing heuristics are also Boolean returning functions. Below we list three simple useful ones. All are defined with respect to a compositing constant: $k_{composite}$

A heuristic relying on absolute frequency

$$Composite(x, y) = \begin{cases} 1, \dfrac{p(x,y)}{Global} \geq k_{composite} \\ 0, \dfrac{p(x,y)}{Global} < k_{composite} \end{cases}$$

A heuristic that is dependent upon the frequency of the parent node:

$$Composite(x, y) = \begin{cases} 1, \dfrac{p(x,y)}{freq(x)} \geq k_{composite} \\ 0, \dfrac{p(x,y)}{freq(x)} < k_{composite} \end{cases}$$

Or where a heuristic which makes an adjustment for the proposed length of the compound node (on the assumption that longer composites are less frequent):

$$Composite(x, y) = \begin{cases} 1, \dfrac{freq(p(x,y))}{freq(_x)} \times (length(x) \times length(y))^p \geq k_{composite} \\ 0, \dfrac{freq(p(x,y))}{freq(_x)} \times (length(x) \times length(y))^p < k_{composite} \end{cases}$$

where

p = constant to offset decreasing frequency

$length : I \rightarrow N$ is a function that returns the length of a node in terms of the primitive nodes from which is made.

The entire process of network growth is essentially the identification of a new node x. Such that

$$Q = Q \cup x.$$

An x is created for any two nodes for which the criteria

$$composite(i_x, i_y) = 1$$

holds true. Further two links are instantiated such that

$$p(i_x) = p(i_x) \cup x \text{ and } p(i_y) = p(i_y) \cup x.$$

3.2.5.6. Nature of Links

It is important to stress that in the outlined algorithms there are in fact, unlike conventional connectionist networks, two distinct types of links: context links and activation links. The activation links are akin to those that are found in conventional neural networks, across which activation can be passed, and which can lead to the subsequent activation of connected nodes. They do, therefore, define a strict causal mapping between nodes. Context links are a softer link; they record the history of a node. In many senses that are a pure artefact of the learning algorithm, however because they do embody useful contextual information they are never actually removed. The process of node growth is interesting; it can be a visualised as a process within which a context link gets stronger and stronger, thickening in the process, as an association is established between nodes. Upon reaching a critical threshold the context link mutates: the centre of the link becomes a node in its own right and the two halves of the context link left become activation links.

We shall now consider the other half of the equation, the clustering algorithm:

3.2.6. Clustering Algorithm

3.2.6.1. Introduction

This section describes the sister algorithm to the compositioning algorithm. From the same starting data (corpora of linear strings of discrete units) a network is to be produced which consists of clustering units. The obvious criteria by which these clusters may be made is commonality of contextual history. Fortunately the history links required by the former algorithm is the ideal resource with which to evaluate contextual history.

In words the clustering algorithm can roughly be described as follows. First some assumptions must be made: assume that the compositioning algorithm is in place; that it is already running and that context histories have been recorded for nodes; further that the contextual histories will continue to be added to whilst the clustering algorithm runs. Now take a node at random. Compute its "difference" with its nearest neighbours. If the difference calculated between any two nodes is less that the objective "clustering criteria" defined, then instantiate a new cluster and the appropriate activation links between child nodes and the new parent cluster. The precise manner in which this "difference" is calculated is discussed in the section below (Section 3.2.6.3).

When a new clustered node is instantiated a context history can be automatically generated for it, which is the aggregation of context history of the child nodes. Further when a new context link is identified for a child node it may safely be added to the context history of the parent, clustered node. In this manner clustered nodes are significantly different to composite nodes. This is because a clustered node represents the aggregation, generalisation or the super-set of its respective child nodes. Therefore what is true for the child node must be true for the parent node. A composite node is different; it represents a specialisation of its respective child nodes. What is true for the child node is only true for the parent composite node when the child node is appearing in a certain context. In composite nodes, therefore, the parent composite may not inherit context histories.

Note, there is an important distinction to be made between the clustering algorithms (see Section 3.2.4.2) that were described for the syntactic classification tasks and the clustering that takes place here. Both of the syntactic classification tasks used an algorithm that for each iteration compares every node against every other and clusters only those two nodes that are closest. The algorithm then proceeds on a best match basis with no concern for the absolute similarity of the two nodes to be considered. The clustering algorithm described in this thesis differs in that nodes are clustered only if they reach some absolute criteria of similarity rather that highest ranking. In this sense the clustering criteria is autonomously and locally implemented at the node level rather that at the global network level.

A further difference is that the syntactic classification algorithms produce single rooted trees (dendrograms). This means that a single child node can belong to only a single immediate cluster. There is no such restriction in the algorithms described here.

3.2.6.2. Formalisation

The formalism for the base network is identical to that defined above for the compositioning algorithm.

The major addition is the definition of a new threshold function for clustering, this is simply:

$$g_{cluster}(x, q_y) = \begin{Bmatrix} 1, x \geq 1) \\ 0, x < 1) \end{Bmatrix}$$

The process of clustering is to identify new node x such that

$$Q=Q \cup x,$$

and x is to represent the cluster of two nodes n_1 and n_2. The threshold for the new function will be

$$t(x) = g_{cluster}.$$

And two linkages must hold between the children and the parent composite:

$$l(n_1) = l(n_1) \cup x \text{ and } l(n_2) = l(n_2) \cup x.$$

In order to identify which nodes are to be clustered we need to define a clustering heuristic. Generically

$$cluster : I, I \rightarrow \{1,0\},$$

that is it maps from a pair of nodes to a Boolean value.

This *cluster* function is usually defined in terms of a distance function δ. This maps a pair of functions to a range between zero and one:

$$\delta : I, I \rightarrow [0,1].$$

Several distance functions are discussed below in the similarity measure section.

The definition of the cluster metric will generally require that the distance function report back a value greater than a predefined similarity constant $k_{similarity}$ and that a crude measure of statistical significance be satisfied. This is usually a simple alpha cut on the frequency of the two clusters defined in terms of $\alpha_{frequency}$. So:

$$cluster(x,y) = \begin{Bmatrix} 1, \delta(x,y) \geq k_{similarity} \wedge freq(x) \geq \alpha_{frequency} \wedge freq(y) \geq \alpha_{frequency} \\ 0, otherwise \end{Bmatrix}$$

3.2.6.3. Measure of Similarity

In order to measure the similarity of two nodes some measure of the difference or distance between these two nodes must be mathematically defined. The most common way to do this is to characterise the two elements between which distance is to be calculated as a specific vector in a multidimensional space. This was the approach that was taken with both of the syntactic classification tasks described in Section 3.2.4.2.

In both of these applications the individual dimensions of this space mapped onto the different items that could be found in a words context (this could be words or word groups). The weighting of a dimension within a specific characterisation vector is proportional to the frequency that the contextual word appears in the focal word's vicinity.

Precisely the same approach is to be taken here. The context history of a node (i.e. the set of context links) can be interpreted as a sparse vector that expands onto this very same space. For practical purposes the expanded vector will be normalised to facilitate comparison of vectors of differing frequency.

Once we have two characterisation vectors for the nodes there are a variety of distance measures that can be applies in order to ascertain a measure of similarity. In both of the syntactic classification tasks four measures were defined: Manhattan, Euclidean, Spearman Rank Correlation Coefficient and Cosine measure (or dot product). These shall now be defined in greater detail:

Manhattan Coefficient

$$D(x,y) = \sum_{i=1}^{n} |x_i - y_i|$$

Euclidean Metric

$$D(x,y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

Spearman Rank Correlation Coefficient

$$D(x,y) = \sum_{i=1}^{n} \left(R_i^x - R_i^y\right)$$

Where Rx and Ry are new vectors computed from x and y by replacing each element by its ordering (the elements magnitude) in its parent vector.

Cosine Measure

$$\text{where } D(x,y) = X \bullet Y = |X||Y| \cos\theta = \sum_{i=1} x_i \times y_i$$

For the purposes of this application a process of what we call *bounce back activation* is used. This is done as follows. Select a node at random and activate this node fully. Pass the activation out through the "*history links*" to the others and let these other nodes take on activation proportional to the strength of the history link. Once received, bounce back this activation, again down the history nodes but this time let the strength of this activation be proportional to the difference between the strength of the activation of the node and the strength of the history link. The emitting node should of course be fully activated. But the most similar nodes will also take on a high activation.

When interpreted spatially so that the context history list is converted to a normalised N-dimensional vector, where N is the number of nodes in the network it is obvious that the bounce-back metric is equivalent to the dot product. Interestingly, where the vector is sparsely populated, which is usually the case, an optimised implementation can use the container of links pertaining to a node to reduce the number of dimensions that need to be considered in producing this metric.

What this means is that:

1. Bounce back activation is mathematically equivalent to the cosine measure.
2. Bounce back (for sparsely populated vectors) is a highly optimised implementation of the cosine measure.
3. Bounce back maintains a higher level of neural plausibility due to its locally autonomous implementation. (For discussion on Local Autonomy see Section 3.2.8)

3.2.7. Interaction of Clustering and Compositioning

To consider how these algorithms co-operate it is easiest to describe (and indeed implement) the processes as parallel multi-threaded operations.

The following pseudo code describes this collaboration:

```
Network n;
Time t;
For (Sentence s = first_sentence TO last_ sentence)
{
   SalientList l;
   For (Word w = first_word_in_s TO last_word_in_s)
   {
          t =t+1;
          IF (w NOT in n)
          {
                 Add w to n as primitive node;
                 Activate w;
          }
   }
}
```

Which calls the activate process

```
Activate(Node n)
{
   if( Threshold reached)
   {
          Make node n salient at time t;
          Pass activation to parents;
   }
}
```

Asynchronous processes

```
Linkage_Monitor
{
   Scan current list of salient nodes
   If (any two node satisfy linkage heuristic)
   {
          record history links against correct nodes
   }
}

Compositioning_Monitor
{
   If (any link history satisfied compositioning heuristic)
   {
          Add new composite node to n
          Add activation links from children to parent
   }
}

Clustering_Monitor
{
   If (any two nodes satisfy clustering heuristic)
   {
          Add new cluster node to n
          Add activation links from children to parent
   }
}
```

This concludes the definition and discussion of the algorithms that may be applied to the activation passing networks from the previous section. The application and configuration of these when applied to real data will be discussed in Chapter 5.

3.2.8. Contrasting Learning Algorithms with Neural Networks in General

It is necessary to explain how the outlined algorithms compare to neural networks and connectionist systems in general. Note, however, such a general discussion is not easy due to the enormous variety and flavours of connectionist systems. But we will start with the obvious commonalities, which are that the outlined system is node based, with interconnecting links and nodes that pass activation between each other along these links. Unlike many connectionist systems, however, the links are un-weighted, but there is no reason in principle why the system could not be extended along these lines. A distinction is preserved between input nodes, which are connected directly to external stimulus, and hidden nodes (in this case not strictly hidden), which take their activation indirectly from the external stimulus. The implemented algorithms are un-supervised and there is no direct analogy to the output node layer. All nodes have a localised (non-

distributed) encoding and so map onto discrete identifiable features of the input domain. The activation function, which determines whether a parent node becomes active or not, comes in two forms for the two varieties of parent nodes that exist: composite and clustering. In both cases this activation function is a simple linear function. Possibly of most interest however is the fact that the network grows. That is the algorithm identifies new potential nodes and reconfigures the network over time.

In the panels discussion at the ICNN97 conference (Kohonen, Hinton, Taylor, Baum et al (1997)) classical connectionist learning, it was suggested, was based on two key ideas:

a) Memory less learning: that is no training examples are to be stored by the learning algorithm, such that each presentation of information can be considered in isolation.
b) Local Autonomy: the nodes of the network are autonomous learners, that is the only information that an individual node can use is that which it can obtain from its immediate local connections. To quote directly: "Local learning embodies the viewpoint that simple, autonomous learners such as single network nodes, can produce complex behaviour in a collective fashion."

The focus of this discussion was to question the validity of these two key qualities in the development of novel connectionist systems.

Within this framework an interesting contrast can be made between so-called "Classical Connectionism" and the algorithms and data structures outlined in this section. This contrast is to be made on the principle of "Local Autonomy". Basically the proposed algorithms break the constraint of local autonomy. It is within the outlined model of "saliency" that this constraint is broken, and necessarily so given the task attempted. The algorithm aims to identify new nodes that combine (through composition or clustering) other nodes within the system. These nodes are by definition originally unconnected. The connectivity that facilitates local autonomy, in the initial case, is simply not there. Saliency (or local memory, as the analogy has been made above) is the mechanism by which this connectivity is bootstrapped into existence.

We can also explore the differences between neural networks and the defined networks and algorithms from another angle. First let us make the distinction between an "active functioning network" and a "training algorithm."

Consider first the "active functioning network." Again the full spectrum of "connectionist" systems is very large, so let us constrain ourselves initially to considering only feed

forward networks. A simple feed forward network by virtue of its topology, link weights and activation function defines a non-linear (possibly linear – depending on the activation function) mapping between a predefined multi-dimensional input space and a predefined multi dimensional output space. The dimensions of the input space are defined by the input nodes and the dimensions of the output nodes are defined by the output nodes. Contrast this with the clustering/compositing networks defined in this thesis. Again it is the definition of a mapping from a multidimensional input space to output space. But over time the number of dimensions within this output space grows as new nodes are added to the system. The mapping is, however, typically simpler, consisting of simple threshold functions. It is also easier to decompose. Unlike a fully connected feed-forward network where the effect of a particular sub-space of the input space is only analysable by tracing the strength of the weights connecting almost all nodes, the effect of specific input subspace of a clustered/composite network is more discrete due to the selective connectivity of the links.

Consider now the training algorithms, specifically we shall consider the simple back propagation algorithm, however most connectionist learning algorithms are similar to this. The back propagation algorithm takes an initially configured feed forward network that defines a specific (and initially often random) mapping from an N-dimensional input to M-dimensional output, and adjusts it by changing the value of the weighted links. The manner that it adjusts the links is best explained by taking another multidimensional metaphor. This time each weight is to be viewed as a distinct dimension in the "error space". The landscape of this error space is defined by the Error measure, which is a global measure that reflects the goodness or the effectiveness of a specific definition of the multidimensional mapping with regard to a specific task. The measurement necessarily requires a-priori knowledge of the specific output vector to which a specific input vector maps. This is another way of saying the algorithm is supervised. The weights of the network are adjusted by attempting to descend this error space by means of gradient descent to reach a point of optimum "goodness".

To contrast, the learning algorithms for the clustering/compositing network are, critically, unsupervised. Also the algorithm is not adjusting a pre-ordained N to M dimensional mapping. Rather it is refining an N dimensional mapping to an output space that is constantly changing. The criteria by which the output space changes and the mapping gets adjusted, (because the algorithm is unsupervised) can not be effected by some global notion of goodness. Rather a very local notion of goodness is employed at
each node in the system. This local notion of goodness may perhaps be generally characterised as "coherence". That is inducing a logical orderly relationship of parts by analysing statistical regularity of co-occurrence. But specifically this global notion of

coherence is embodied in the algorithms that define the clustering and compositing heuristics.

3.3. Conclusions

3.3.1. Knowledge Architecture

The knowledge representation scheme that has been chosen is essentially a localist connectionist network, where each node represents a distinct feature within the problem domain. In the initial case the activation function is Boolean as are the linkages, this is to simplify both the representation and the processing. This architecture was chosen over the alternatives for the following reasons:

1. Simplicity: it is essential that the knowledge representation scheme be simple so that the tutor can easily specify and refine the criteria by which the questions are judged. A connectionist type network maps very well onto a graphical representation, which is intuitive for the tutor to manipulate.
2. Extendibility: as has been discussed, the outlined network maps onto both connectionist and symbolic/logical processing paradigms. This means any structure may be enhanced with the appropriate computational properties deemed necessary at a later stage.
3. Macro structures: considerable effort has been made above to develop a set of macro structures (themselves constructed from the primitive connectionist units) which specifically map onto the perceived functional requirements of the problem domain.
4. Communication: natural language processing is traditionally a modular process consisting of several sub disciplines. As has been shown earlier, to attempt to perform these processes in a serial order is doomed to failure. There are innumerable examples of deadlocks; where syntactic information is needed to resolve semantic ambiguity or semantic information is needed to resolve syntactic ambiguity. Even within a single of these disciplines there are examples of deadlocks. Take for example the semantic ambiguity problem. The connectionist processing paradigm offers a solution to this problem. The parallel propagation of activation can allow a bootstrapping of a solution, similar to the constraint satisfaction problems solved with the interactive activation and competition models. (The symbolic alternative is the use of blackboard type architectures e.g. Hearsay II, Erman (1988))
5. Transparency: the localist representation of the network provides transparency to the processing. In other words, when the network has made a decision about a sentence it is possible to trace through the activations giving an intelligible

explanation of the process. This is the much vaunted advantage that traditional symbolic expert systems have over distributed connectionist systems.

6. Learning: the knowledge architecture has been designed to be amenable to application of learning algorithms. Without having any hard evidence to support this claim, it does seem that knowledge schemas of architectural simplicity are more amenable to the application of generalised learning algorithms. The reasoning behind this being that the more architectural distinctions made within the knowledge base, the more complex the algorithm must be. Also incoming data must be categorised more finely.

As a final point, by choosing a simple architecture where the primitive elements have low computational and representational power, and bearing in mind that the entire structure will be iteratively refined in accordance with the strict functional requirements of the application, it should follow that any structure inherent within the problem domain should naturally emerge within the refined network. This means that, if indeed, one of the higher order knowledge description languages is more appropriate to the application, then the relevant high order qualities should be emergent properties of the evolving network. If recognised as such, this will be a useful exercise in itself. This work can be found within the publications Allott et al (1994a, 1994b, 1994c).

3.3.2. Learning Algorithms

Within the second section two algorithms have been defined for the production of activation-passing networks, which consist of compositioning and clustering nodes. This work has been published[14] in Allott et al (1995), Allott et al (1997a), Allott et al (1997c), Allott et al (1997d).

Intuitively the richness of the representational structure and the consequent functional power that is due to the activation-passing properties of the network is very appealing. However, what remains to be proved is the empirical utility of these knowledge structures within the automated assessment application (or any other application for that matter.) The development of a framework within which this assessment of empirical utility can take place and the performance of this assessment will be the focus of the second section of Chapter 5.

---

[14] In a further paper (1999b) some specific problems and solutions to the compositioning of highly time dependent data have been outlined, however this work has not been discussed in this thesis

4. System Architecture and Implementation

4.1. Architecture

In order to generate and evaluate knowledge bases within the automated assessment domain a full modular system must be developed. This system must facilitate: the generation of HCNs (hand-crafted networks); the persistence of knowledge bases; the collation and processing of student data; the application of HCNs to student data; the integration of any resources to aid in the generation of HCNs; the application of learning algorithms to produce AGNs (algorithmically produced networks); the application of AGNs to student data and the statistical analysis of results. Such a system may be analysed architecturally in terms of three distinct component types: external entities; data resources and data processes.

There are six primary data processes:

1. Data acquisition process.
2. HCN Creation – knowledge acquisition.
3. HCN Decision process.
4. AGN Creation
5. AGN Decision Process.
6. Statistical Analysis Module.

There are three primary data resources:

1. Questions to be answered
2. Answers to questions given by students
3. Knowledge resource

And there are two external entity types (external entities that introduce data into the system):

1. Students
2. Experts

The data resource is the bank of student responses to questions and the questions asked themselves. Whereas the knowledge resource is the tutor created knowledge base which

the decision process uses to discriminate between incorrect and correct responses. The data acquisition and knowledge acquisition processes are the mechanisms by which the two respective data bases are created and maintained. The experts are the external entities responsible for the knowledge base, whereas the students are the external entities from which the data base is derived.

The following diagram depicts these entities and the relationships that hold between them.

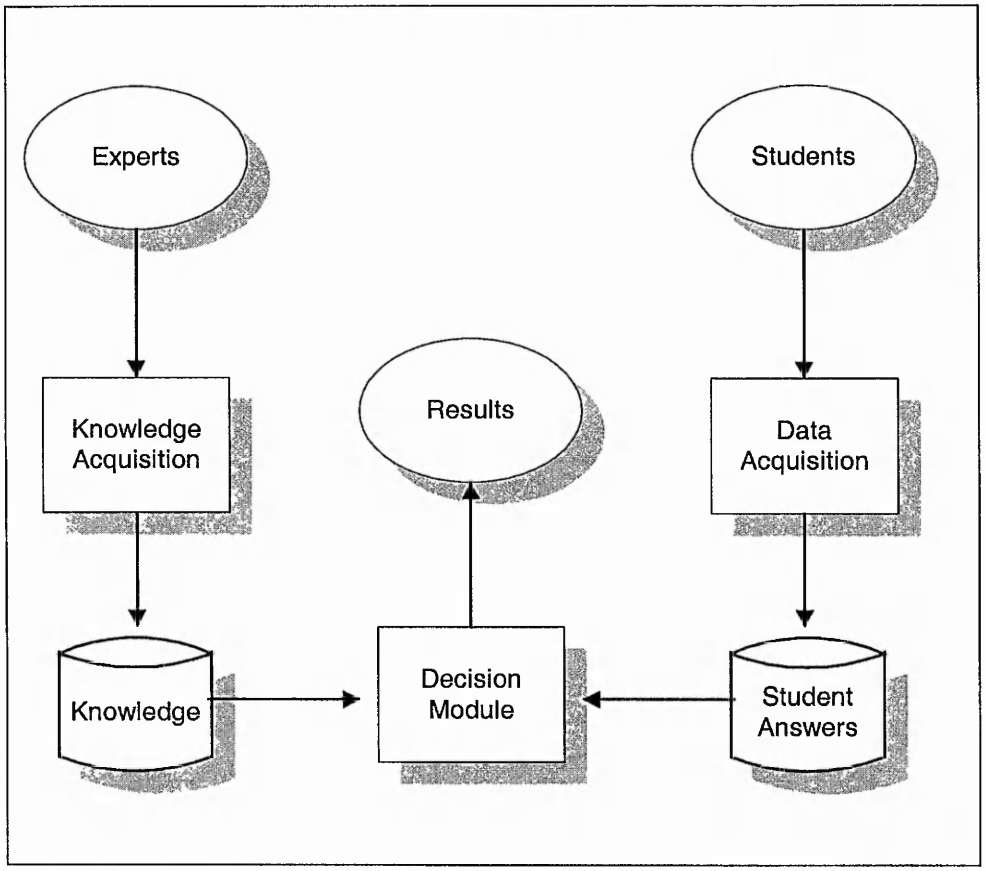


Figure 4-1 Entity-Process Diagram

We shall examine the specifics of each of these modules as well as the implementation. Mention may be made of the final task or use to which it will be put, but a fuller exposition will be found within the experiment section.

4.1.1. Resources

4.1.1.1. Representation Module

The architectural qualities of the representational schema to be used have been discussed in depth within the previous chapter. Aspects considered here are: implementation, specifically issues of persistence, serialisation and sample constructs

A node of the form discussed in the previous chapter can be characterised within the following C++ class definition.

```
class CNode : public CObject
{
private:
   int    m_NodeID;
   CList  m_Parents;
   CList  m_Children;
   float  m_Threshold;
   float  m_Activation;

public:
   void   Activate      (float iIncomingActivation);
}
```

Of the member variables only activation is a non persistent variable. In other words activation is a variable which is required at run-time only. All other variables must be serialisable so that a permanent record of the nodes state may be made. Using standard object oriented techniques this is done by streaming out the contents of the object, to a file, with stream operators. The format of the text file into which a permanent record of variable states is as follows:

```
Node NodeID
{
   Parent        OtherNodeID1
   Parent        OtherNodeID2
   Parent        OtherNodeID3
   Threshold           ThresholdValue
}
```

A larger sample of such a file may be found in Appendix C.

Note that child links are not recorded within the file. This is because the child/parent relationship is assumed to be a reciprocal relationship, as such child links are redundant and are instantiated at run-time only for purposes of computational efficiency.

The log file is a standard text file which can be edited by the tutor directly. However the preferred method of access is the graphical interface (see Section 4.1.2.2).

The entire log file is parsed as a two pass process. The first pass instantiates all nodes recording their unique ID within the instantiated object. Within the second pass all node linkages are resolved from node ID links to run-time object pointers.

It is apparent from the file structure that the instantiated network will form a tree-type hierarchy. There is at present no checking for recursive structures within this hierarchy.

4.1.1.2. Data Resource

The data resource used is the master document file generated by the QuestionMark application. See section on data acquisition process for a full description.

4.1.2. Processes

4.1.2.1. Data Acquisition Process

To set the tests to the students and to collate the student responses a commercially available assessment package called QuestionMark was used. The test could be sat by all students simultaneously, with the questions being disseminated and the answers being retrieved across the network.

The test itself was set to two sets of students, actually from subsequent academic years. There were approx 60-70 students in each set. 20 questions in all were given on a general "end of first year" computer science topics. The questions were given in test like environments and the students had approximately 45 minutes to complete the test. Students typed their responses directly into the computer. (Full details of the questions may be found in the appendix.) The students were not told nor were they aware that their responses would be analysed by a computer. As far as they were concerned there responses were simply going to be marked by their tutor (and indeed they were).

The QuestionMark application has the facility to generate a master document which concatenates the answers to all questions from all students. The format of this file is:

1. Student name/ student number
2. Question
3. Answer

For a full example see Appendix B. The format of this document is syntactically well defined. Making use of the distinctive text strings that exist within the master document, a fairly robust context sensitive grammar was identified which could parse the master document into its component parts. An API was provided that through a real time parser would access the master document and pull out a particular response from a particular student on request.

The principal calls in this API are:

```
char* GetStudentName(int iStudentNo);
char* GetQuestion(int iQuestionNo);
char* GetAnswer(int iStudentNo, int iQuestionNo);
```

4.1.2.2. Knowledge Acquisition – HCN creation

The tutor views and edits the knowledge base through the knowledge editing module. This is a custom built Windows program which allows the knowledge structure to be edited with click and drag movements of the mouse.

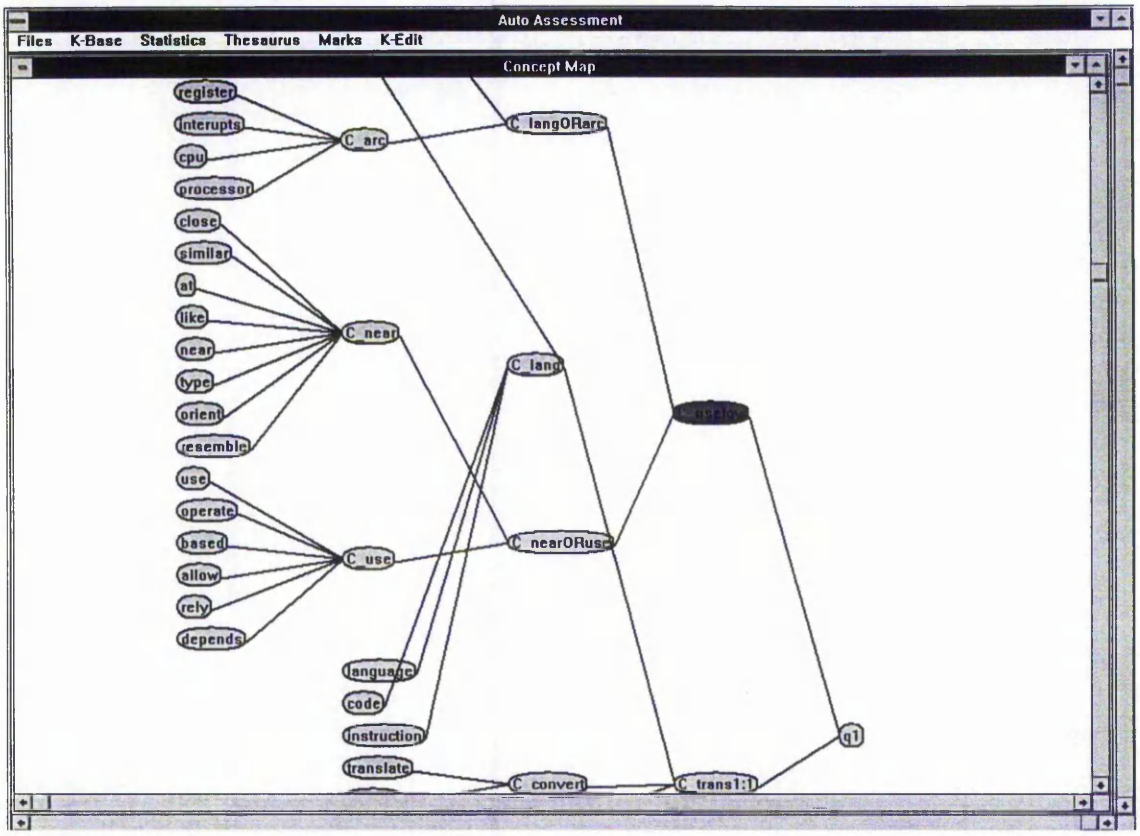


Figure 4-2 Screen-shot Knowledge Base Editor

The hierarchical network is presented with the lowest nodes within the tree on the left hand side of the tree. This is because most produced trees are wider than they are deep. The breadth of the network is explored through vertical scrolling which is seen as more natural behaviour.

Double clicking upon a node accesses the properties of a node. This will bring up an editable dialogue box as below:

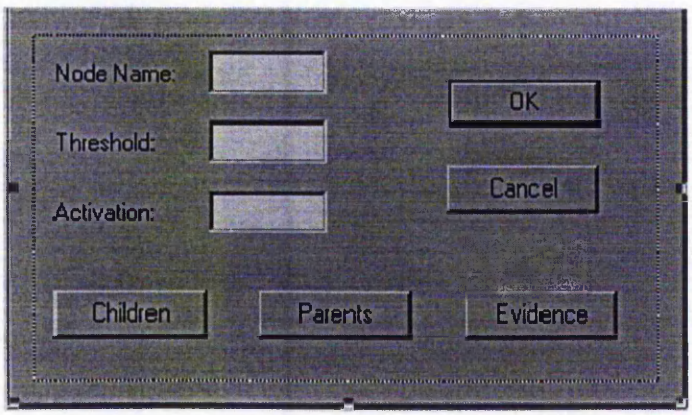


Figure 4-3 Screen-shot Dialog Node Details

The qualities of a link are accessed by dragging between the relevant nodes.

The two dimensional presentation of the network on screen is governed by a simple constraint satisfaction algorithm, the prime constraint being to minimise the number of crossing linkages.

All changes made to the knowledge base through the GUI are made permanent when the knowledge base is saved. The form of the saved files is the file specified above. This is another way of saying that the text description file and the GUI network are two realisations or views of the same underlying knowledge base.

This covers the use of the point and click graphical user interface to produce hand crafted networks. Further details of tools that were used to aid this process can be found in sections 4.3,4.4 and 4.5.

4.1.2.3. HCN Decision Module

As has been explained within the Knowledge Architecture chapter, due to the nature of the activation passing nodes, the process of sentence evaluation is implemented as a distributed algorithm across the network. There is, unlike a conventional expert system, no central decision making component. The evaluation process works as follows. The words that constitute the sentence to be evaluated are passed to the knowledge base. Those primitive surface nodes that most closely match the individual nodes are activated. (Close matching in this sense is determined as those node who have an evidence list member that exactly matches the input string, see Section 3.1.1 for further details.) The contextual disambiguation that is a property of the interactive activation and computation model will lead to the appropriate deep nodes being activated. Deep nodes are combined through node linkage to trigger a full question node. Activation of a question nodes indicates that the presented question matches.

To simulate the sequential nature of the input sentences, input nodes were activated at consecutive time slots, where a single time slot was equivalent to the time that it should take an activation to pass along a link from its child to its parent node.

The implementation of the activation passing between nodes went through two phases of development. As is typical with such interconnected node based systems there can be a combinatorial explosion, which leads to an extremely slow simulation of the activation passing process on serial computers. To counter this an initial highly optimised algorithm was used, which by producing temporary data structures at each node, could dynamically

reduce the number of links and states that need to be considered by making certain assumptions about the binary nature of the activation, and the directional nature of the linkages that passed only upwards through a bottom heavy hierarchy. In essence it was a serial algorithm that was aware of the topology of the network when passing activation from one node to another.

This optimised algorithm was very fast and efficient, two qualities that proved essential on the initial incarnation of the system as it was running on relatively poor hardware. However, later incarnations ran on much faster hardware and these considerations proved far less important. Further, the assumptions upon which the optimised algorithm depended upon, broke down when considering the algorithmically produced network. Specifically, the upward-directional nature of the linkages and even the bottom heaviness of the hierarchy could not be depended upon. For these instances a second algorithm was used. This was a far simpler algorithm, distributed in nature, which made no assumptions about the topology of the network. Consequently, the algorithm was slower, but was both simpler to implement and allowed far greater flexibility in the type one nodes that could be used and the topology of network that was permissible.

4.1.2.4. AGN creation – the application of the learning algorithms

The composition and clustering algorithms have been discussed (Section 3.2). The creation of a particular algorithmically produced network simply involves (a) the configuration of the algorithm and (b) feeding the algorithm with the student response data. An activation passing network is produced which is similar to the HCNs in that it is made from the same architectural constituent parts, but the typical topology is entirely different. The process is implemented as a simple function which is fed with a set of student answers and produces a composite object which functions as an activation passing network. The produced network is an instantiated runtime object model, but may be persisted using standard streaming operators.

4.1.2.5. AGN Decision process

The AGNs produced from the learning algorithms, as has already been mentioned, cannot be used as a sentence judgment procedure. This is because unlike the HCNs, they do not have the single rooted tree-like topology where a root single node represents a specific answer's truth status. Such a structure would be impossible for the stipulated algorithms to produce, as they are essentially unsupervised processes. Consequently a different sentence judgement process was needed. As will be discussed in depth in Chapter 5, the

technique of Latent Semantic Analysis was used to perform this sentence judgement function.

In order that the two decision procedures (HCN and AGN) would appear the same to the outside world, and specifically to the statistical analysis module – so that they may be compared – the object-orientated technique was used of separating interface from implementation. Specifically a simple "*Marker*" interface is defined which not only do the HCN and AGN decision processes implement, but also the module that encapsulates the human markers decision of a sentences truth state.

4.1.2.6. Statistical Analysis Module

A whole battery of tests are developed in Chapter 5 for the analysis and comparison of the various decision modules. A standard analysis engine was implemented into which various statistical tests could be plugged as separate components. This engine facilitated the uniform reporting of the results to file or GUI as appropriate.

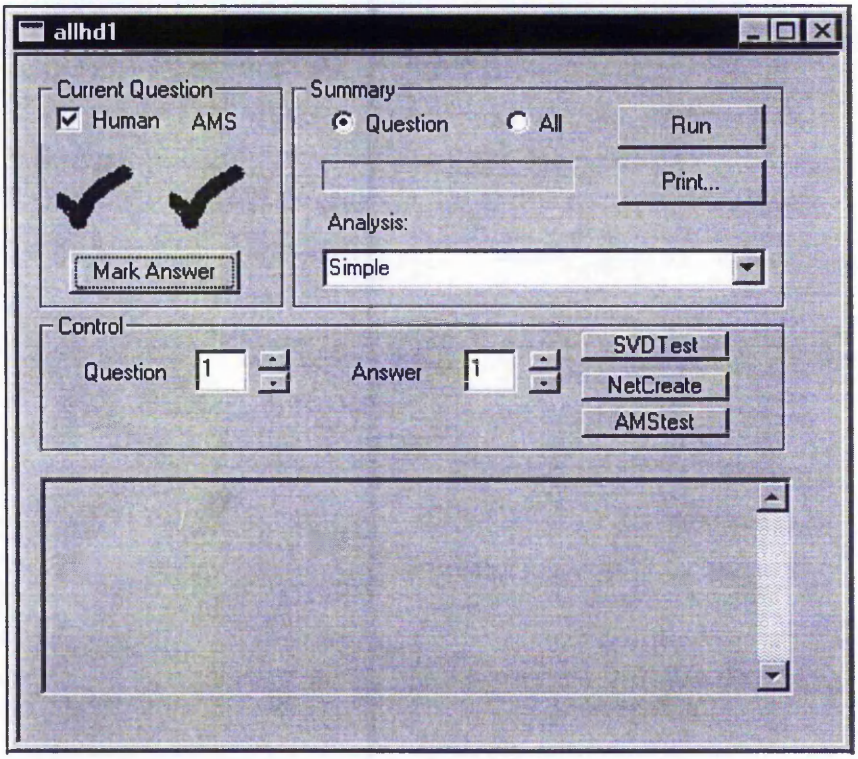


Figure 4-4 GUI Interface used for Controlling Statistical Analysis

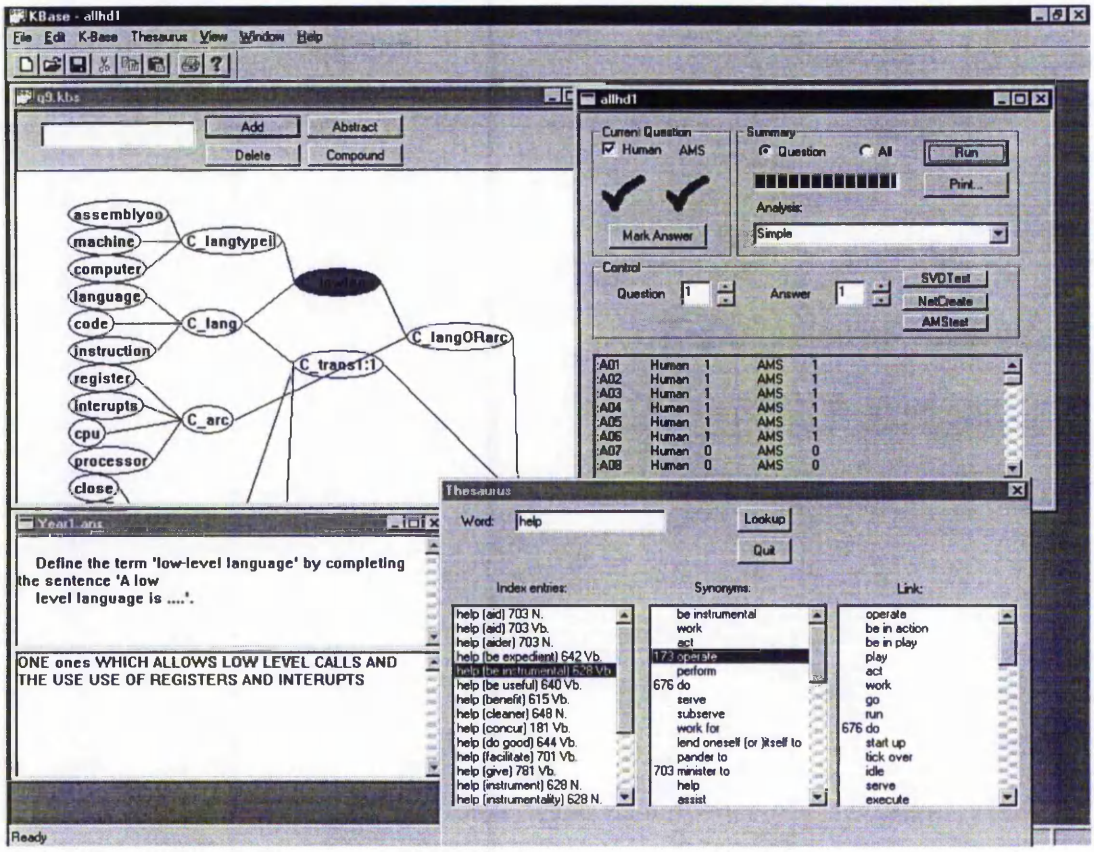


Figure 4-5 Integrated GUI

4.1.3. Entities

4.1.3.1. Tutors

Tutors are assumed to be experienced computing science lecturers with extensive knowledge of the problem domain. Typically there is only one tutor associated with the generation of a particular knowledge base.

4.1.3.2. Students

Little previous experience of computers was assumed. Each of the students was taken from the second term of the first year, and each has completed an introductory course on computers.

4.2. Implementation

The entire system was written in C++, within three distinct modules:

1. Core engine
2. GUI module
3. Text interface module

The core engine dealt with such issues as storing, serialisation and implementation of node dynamics which give rise to the decision process. The GUI module is a windows system which both presents the information and allows the user to trigger activities. The advantage of such an implementation was that the isolation of the two functions allowed a distinct DOS based GUI interface to be implemented which hooked into the same essential run-time activities that the GUI module did. This facilitated rapid development/testing for run-time components as well as giving users the flexibility to run the program on different operating systems.

The initial target platform is a PC compatible computer with 386 33Mhz or above, 4Mb memory and 10Mb hard disk space. (Note this was the original PC upon which the network was developed). The later systems upon which the algorithms were run were dual Pentium 500 Xeon processors, 256Mb of memory and utilising up to 100 Mb of hard disk.

The initial development environment was Windows 3.1 using a Borland compiler and the OWL (Object Windows Library) for the graphical user interface, and custom container libraries were used for implementation of the networks.

Approximately half way through the entire system was re-implemented. This time the target operating system was Windows NT Workstation 4.0. The development environment progressed through MSDev 4.0/5.0/6.0. Graphics components were implemented using a combination of raw Windows SDK, MFC (Microsoft Foundation Classes) and ATL (Active Template Library). COM (Component Object Model) and similar were used throughout to manage the architectural complexity. STL (Standard Template Library) was used for the container classes with which the networks and algorithms were implemented. Perl was used for a variety of pre-processing tasks.

4.3. Integrated Roget's Thesaurus

To aid the creation of the knowledge base a custom interface onto a version of Roget's Thesaurus (Roget (1987)) has been integrated into the system. The various methods in which the thesaurus was employed in the creation of the knowledge base will be described below. Here a few paragraphs will be spent discussing the logical structure of the thesaurus and the interface that was implemented to aid navigation through this valuable data resource.

4.3.1. Structure

Roget's thesaurus is arranged into a three deep hierarchy, going top down Section, Class and Header.

1. **Section:** there are six of these. In order: abstract relations, space, matter, intellect, volition and emotion. These are the primary categories into which each word is classified.
2. **Class:** these are finer subdivisions of Headers. There are between 3 and 8 of these per header. These, for example, divide the header abstract relations into order, number, time etc.
3. **Header:** there are just under 1000 of these in the entire book (exactly how many depends on the particular revision). They consist of lists of words that fall under this category. In fact each header is further subdivided into the following sections.
    a) **Part of speech**: the list of words under each header is first classified into the various parts of speech. Roget used five distinct parts of speech: nouns, verbs, adjectives, adverbs and interjections.
    b) **Paragraph**: within each part of speech the words are grouped into paragraphs each of which has a keyword at its head. This keyword is supposed to give a key to the types of words that are to follow it, not to necessarily act as a synonymous grouping.
    c) **Semicolon groups**: finally although all words are separated with commas, semicolons are used to group together subsets of words that have similar context or level of usage.

Apart from this structure the whole thesaurus is heavily cross indexed. The form of this cross indexing is that if any word occurring in a paragraph is itself a keyword of another paragraph, it is italicised and the number of the header to which it is linked is prepended.

4.3.2. Navigation

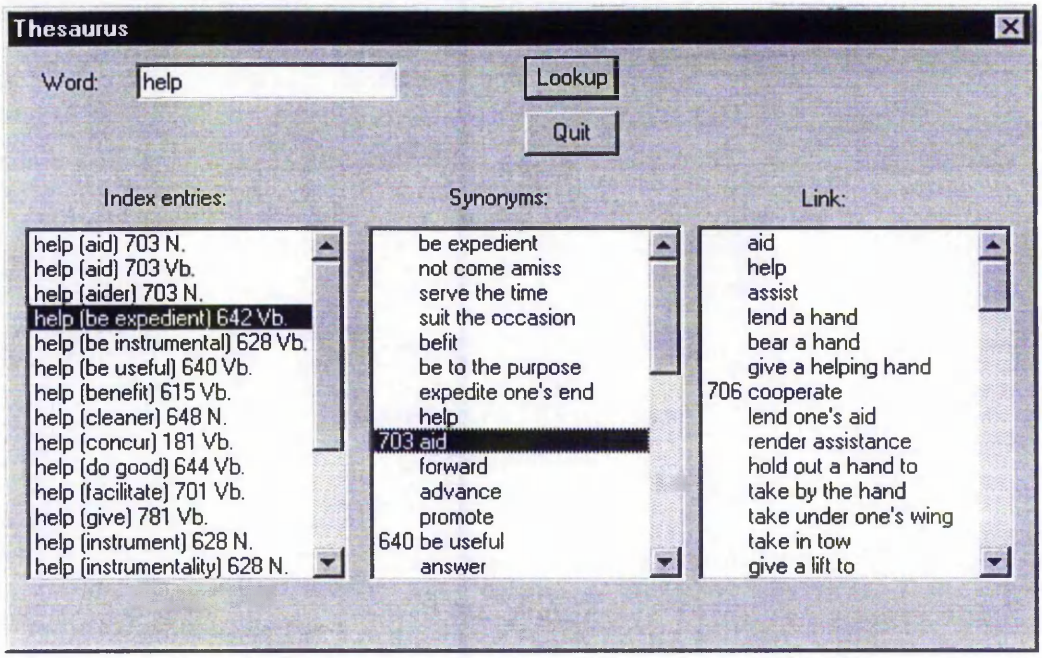


Figure 4-6 Custom GUI for Thesaurus

A simple user interface has been provided for to the tutor to navigate around the thesaurus and to extract the appropriate information. It consists of a simple dialog box containing multilevel listboxes. The first two levels display the current section and class respectively. Navigation is possible by clicking on the contents of these boxes. A click on a particular section displays the associated classes. A click on a particular class displays the list of associated headers in a third listbox. Two further listboxes are used to display the contents of a particular header once selected. Two are used here because if cross indexes are used to navigate, rather than the structured hierarchy, then both sides of the linkage are visible at the same time giving the user a visual representation of the context in which it was found.

Word lookup is also possible through the use of an edit box. Characteristically a word look up will match many distinct headers (again the problem of ambiguity). The various uses will be displayed within the header list box allowing the user to explore the possible variations.

Information may be extracted from the thesaurus in one of two ways. Individual words may be extracted as evidence nodes, or whole synonymous groups may be transplanted

directly into the knowledge base. This is performed by the simple drag and drop of the appropriate elements.

4.4. Knowledge Base Creation

The physical tools used to develop and modify the knowledge base have been described above. Below that the actual procedure for generating the knowledge base shall be described. The process to be described is a bottom up process where the low level items are identified first, and the macroscopic structure is built up from this.

The process of knowledge base creation can be divided into four stages:

Lexical acquisition: at this stage the core evidence nodes are identified, that is the actual strings and string variations that occur frequently within the data.
Synonym grouping: this is where clusters are built up from the actual string forms. These clusters are intended to represent commonality of meaning. This is a somewhat looser criteria than actual synonymity and can be used to map the implied contextual meaning of lexical items rather than the orthodox meaning.
Fact definition: here the clustered groups are built up into compound nodes that are intended to model facts.
Iterative refinement: this is an optional stage that requires some measure of the performance of an operating network (such as a pre-marked set of answers). With this performance metric the network can be iteratively refined.

Each of these stages can be further broken down.

4.5. Lexical acquisition

4.5.1. Surface inspection

The first and simplest method is a simple surface inspection of the data. By sight the most frequently occurring words can be identified and turned into nodes. To make this process easier there is, built into the interface, an option by which a right mouse click on a selected word will immediately instantiate a new evidence node within the currently open knowledge base. A further utility is available which simply summarises word counts across answers for a specific question. In the cases where data has been pre-marked by a human these word counts can be clustered in terms of words occurring in correct answers and words occurring in incorrect answers. Similarly these words may be turned into evidence nodes by simply dragging them into an open knowledge base.

4.5.1.1. Lemmatisation

The results of the surface inspection can be greatly enhanced if the individual words are lemmatised before hand. This lemmatisation process was crudely implemented in a simple C program that performed elementary string replacements. Evidence nodes instantiated from the lemmatised version automatically have their prime anticipated morphological variants added to their evidence list.

4.5.2. Synonym grouping

4.5.2.1. Hand picking

Again the first method is simple manual inspection. Evidence nodes that are to be clustered are multiply selected then on a right click menu selection the entire set are attached to a newly instantiated parent synonym node. The criteria for clustering is the tutors own intuition as to what nodes share meaning.

4.5.2.2. Thesaurus inspection

The integrated Roget's thesaurus described above can be used to partially automate this process of synonym grouping. A particular word may be selected within the data or typed directly into the thesaurus' input editbox. This will essentially perform a look-up into the thesaurus index, producing the list of possible headers. Appropriate headers can then be selected by the user and relevant synonymous groups be inserted into the knowledge base in the manner described above. Note user intervention is required here to make the selection of appropriate headers, this is because the potential ambiguity here is difficult to resolve automatically.

4.5.2.3. Thesaurus head count

The thesaurus head count procedure is intended, in part, to overcome the problem of ambiguity which makes the automated integration of the thesaurus problematic. Simply, it is a process by which every word occurring in answer to a question is noted and looked up within the thesaurus index. Typically such a look up will return multiple hits corresponding to the multiple possible senses of a word. The hope is that most hits will have senses that most closely correspond to the senses used and implied by the answers to the questions.

The reasoning behind this process is that many different words will be used to express the same general meaning. Each word will, more than likely, have many possible meanings. However, each of these words is unlikely to share the same sense distribution. The senses that are most frequently shared, that is those with the greatest overlap, are likely to be the best choices.

4.5.3. Fact definition

The definition of the fact that is to correspond to the correct answer template, typically occurs after the appropriate synonymous groups have been identified. In the simplest and most frequent cases this corresponds to the specification of a conjunction of synonymous groups. More complex issues are addressed on an ad hoc basis using a variety of the techniques discussed in the knowledge base section.

4.5.4. Iterative refinement

If there exists some mechanism by which performance can be evaluated, the knowledge base may be iteratively refined. Such performance measures will typically require that all or part of the answer data be pre hand-marked. Correlation measures that are particularly appropriate to the automated assessment task will be discussed in Chapter 5.

In order to actually refine the knowledge base, over and above the generalised correlative measure that indicate the performance across the data as a whole, it is extremely useful to implement exception lists. These are simply lists of the answers that are actually contributing adversely to the distinct error measures. In practice this works as follows.

1. A first pass at creating an adequate knowledge base is made.
2. The knowledge base is applied to the data set.
3. Correlative measures and exception lists are generated.
4. By looking at only the exception lists whilst refining the knowledge base, the present deficiencies are addressed directly.

This process addresses the issue of knowledge base creation in an efficient divide and conquer manner.

4.6. Conclusions

The description above documents an integrated system for both the generation and application of knowledge bases to collated student data. In the design and implementation great emphasis has been placed upon making the knowledge bases easy to create, and incorporating useful feedback on the performance of particular knowledge bases.

With this in mind the graphical user interface for the creation of knowledge bases has proved particularly successful. It has reduced the potentially horrendous problem of specifying a knowledge base in some arbitrary and complex formal syntax to a simple point and click procedure where the output is readily interpretable.

The embedded functionality derived from the Roget's Thesaurus has also proved useful, however there are a few caveats. Firstly, automated integration is difficult due to the recurring problem of ambiguity. Although not fully automated, the partial solution of the thesaurus word head count in conjunction with user intervention for manual ambiguity resolution does provide a workable solution. Secondly, and perhaps more importantly, the lack of domain specificity within the thesaurus means that the technical lexical requirements of the data set are completely unsatisfied by the resource. If further knowledge base construction improvements are deemed necessary, this issue of technical lexical acquisition and grouping must be a prime area for exploration.

5. Experiments

In this chapter metrics of evaluation are established and experiments are devised and reported on which directly investigate the validity of the theoretical data structures and learning algorithms developed in Chapter 3.

There are four experiments in all. They are divided into the evaluation of knowledge architecture and evaluation of learning algorithms.

Evaluation of Knowledge Architecture:

- Experiment 1 – hand crafted knowledge bases conforming to the novel knowledge architecture were produced. These knowledge bases were applied to the task of the automated assessment of students' single sentence responses to questions. Using human marked student responses as a control the performance metrics of the system were established.
- Experiment 2 – the knowledge bases produced from the previous experiment were applied to new unseen data in order to establish the system's capacity to generalise to novel data. The performance metrics of this system were contrasted with those from Experiment 1.

Evaluation of Learning:

- Experiment 3 – uses the technique of Latent Semantic Analysis (LSA) in order to establish the control performance metrics necessary in order to evaluate the usefulness of the information provided by the learning algorithms. Incidentally, the performance metrics are of interest in their own right and can be compared to those from the previous two experiments.
- Experiment 4 – applies the learning algorithms devised in Chapter 3 Section 3.1.1 to the input data. Grown networks are used to "perceptually augment" the input data. This "perceptually augmented" data is fed to the same LSA algorithm as Experiment 3 and the performance metrics are compared.

For all experiments considerable time and effort is expended devising not only the experimental design but also the actual metrics that provide the empirical evidence for the relevant theory.

5.1. Evaluation of Knowledge Architecture

5.1.1. Introduction

The hypothesis underpinning the proposed evaluation experiments is that an artificial mechanism could diagnose simple sentences as being correct or incorrect and if these diagnoses correlate well with the independent evaluation of a human marker, the artificial mechanism has a good model of the sense of the sentence. This is very much within the Turing (1964) school of AI evaluation, using a functional evaluation of competence.

All sentences are given in response to a single simple question. A binary response is being forced upon the decision procedure which determines the truth status of that sentence, something that is enforced upon every teacher when they have to mark exam scripts. As many students sit the exam many possible responses or utterances are made to a single question. A tutor's independent evaluation of the truth status of the utterances is seen as a good model of the correctness of the responses (if in fact such an objective evaluation is possible). Note, when we talk of the truth status of the utterance, we do not mean is this statement is true, but is a statement that correctly answers the question being asked. We therefore have a full statistically significant set of empirical data and an ideal control against which the system may be evaluated.

5.1.2. Metrics of Evaluation

The metrics under which the performance of the knowledge schema may be evaluated can be divided into two broad types: glass box metrics and black box metrics. Black box metrics are implementation independent and therefore can be used to compare entirely different systems operating under the same application domain. Glass box metrics are implementation dependant and are therefore useful for plotting performance variations or identifying operational inefficiencies within a particular implementation only. These will be considered in turn.

5.1.2.1. Black Box Metrics

5.1.2.1.1. Empirical Measures of Success

Taking heed of Flanagan's work (1994), specifically, and in general the work on evaluating NLP systems discussed in Chapter 2, in order to evaluate a knowledge architecture operating

under this domain, implementation independent statistics are necessary that evaluate and characterise the performance of any particular system.

Several distinct measures of success can be employed. Apart from a general measure of 'goodness': correlation, four independent error metrics can be employed to best characterise the behaviour of the system. All these metrics are Bayesian statistics. Bayesian statistics are necessary to adjust for the correct /incorrect ratio of answers in any particular sample, and so allow aggregation of the metrics over the many questions within a particular sample. As implementation independent evaluations they concur with the Sparck-Jones & Galliers (1993) definition of a black box evaluation mechanism.

In order to define the respective error metrics, a mathematical language is necessary in which to phrase the performance of the various aspects of the system.

Each sentence Σ^+ is an finite ordered set of strings formed from a population of strings Σ. The set of all possible sentences U is the superset of all possible Σ^+'s. A particular sample of responses to a specific question A_i is a subset of U and comprises individual sentences, thus:

$$Ai = \{\sigma_{1i}, \sigma_{2i}\sigma_{3i}, \sigma_{4i}, \sigma_{5i}, \ldots \sigma ni\}$$

where

$$Ai \subset U$$

and so

$$\forall \sigma \subset U$$

A knowledge base K comprises the knowledge necessary to mark the questions Q $=\{q_1,q_2,q_3,\ldots,q_m\}$, and so is defined:

$$K = \{k_1, k_2 k_3, k_4, k_5, \ldots km\}$$

However it is worth pointing out that in the above implementation, due to the node sharing nature of the hierarchical network , k_i is a non discrete subset of K. In other words k_1 to k_m are non-orthogonal.

There are thus m questions and n students in any run of a system, with mxn sentences to evaluate.

Functions which evaluate a sentence's truth value do so under a particular knowledge set k_i, and do so within the Boolean set {0,1}. Thus an evaluative function e can be characterised:

$$e : A, K \rightarrow \{0,1\}$$

There are three such evaluative functions of relevance here: t , $\hat{t}$ and s. t is the actual truth state of a sentence under a particular question i.e. if the first student's answer to question 1 was correct then, $t(\sigma_1, k_1) = 1$ and if incorrect $t(\sigma_1, k_1) = 0$. For obvious practical and philosophical reasons the state of t is inaccessible. Instead the estimator function $\hat{t}$ is used, this is in fact the tutor's assessment of the truth-value, and the fact that it is represented mathematically with an estimator function reflects the real world issue of marker variability mentioned above. The s function models the automated assessment system's evaluation, so $s(\sigma_1, k_1) = 1$ if the automated assessment system judges the first student's response to question 1 correct, etc.

The following truth table characterises the performance of both systems (automated and human) for a particular question.

Answer to Question q1	Tutor $\hat{t}$	Automated s.
σ_{11}	0	0
σ_{12}	1	0
σ_{13}	1	1
σ_{14}	1	1
σ_{15}	0	1
...	...	...
σ_{1m}	1	1

Figure 5-1 Sample Tutor-Automated Performance Table

Now let us consider some specific measures of success:

5.1.2.1.1.1. Correlation

This is a simple statistical measure of similarity, but constitutes a good simple single measure of the performance of the system. This measure produces a figure between -1 and 1, where 1 is perfectly correlated and –1 is perfectly anti-correlated.

$$correl = \frac{n\sum_i t(i)s(i) - \sum_i t(i)\sum_i s(i)}{\sqrt{\left[n\sum_i t(i)^2 - \left(\sum_i t(i)\right)^2\right]\left[n\sum s(i)^2 - \left(\sum s(i)\right)^2\right]}}$$

5.1.2.1.1.2. False Positive

The false positive error is a measure of the chance that a system's assessment that a question is correct is itself incorrect. This is in fact the Bayesian probability that a question is false given that the system has evaluated it true.

$$FP(\sigma_i) = p(\hat{t}(\sigma_i) = 0 | s(\sigma_i) = 1)$$

This metric may be derived empirically from a data set of answers A_i.

$$p(\hat{t}(\sigma_i) = 0 | s(\sigma_i) = 1) = \frac{\left| \bigcup_{\sigma \in A_i} \begin{cases} \sigma, & t(\sigma,k) = 0 \text{ and } s(\sigma,k) = 1 \\ \varnothing, & \text{otherwise} \end{cases} \right|}{\left| \bigcup_{\sigma \in A_i} \begin{cases} \sigma, & s(\sigma,k) = 1 \\ \varnothing, & \text{otherwise} \end{cases} \right|}$$

5.1.2.1.1.3. False Negative

The false negative error is a measure of the chance that a system's assessment that a question is incorrect is itself incorrect. This is in fact the Bayesian probability that a question is correct given that the system has evaluated it false.

$$FN(\sigma_i) = p(\hat{t}(\sigma_i) = 1 | s(\sigma_i) = 0)$$

This metric may be derived empirically from a data set of answers A_i.

$$p(\hat{t}(\sigma_i) = 1 | s(\sigma_i) = 0) = \frac{\left| \bigcup_{\sigma \in A_i} \begin{cases} \sigma, & t(\sigma,k) = 1 \text{ and } s(\sigma,k) = 0 \\ \varnothing, & \text{otherwise} \end{cases} \right|}{\left| \bigcup_{\sigma \in A_i} \begin{cases} \sigma, & s(\sigma,k) = 0 \\ \varnothing, & \text{otherwise} \end{cases} \right|}$$

5.1.2.1.1.4. Correct is Correct Estimator

The correct estimator error is a measure of the chance that the system will evaluate a question correct if it is indeed correct. This is in fact the Bayesian probability that the system evaluates a question true given that it is true.

$$CE(\sigma_i) = p(s(\sigma_i) = 1 | \hat{t}(\sigma_i) = 1)$$

This metric may be derived empirically from a data set of answers A_i.

$$p(s(\sigma_i) = 1 | \hat{t}(\sigma_i) = 1) = \frac{\left| \bigcup_{\sigma \in A_i} \begin{cases} \sigma, & \hat{t}(\sigma,k) = 1 \text{ and } s(\sigma,k) = 1 \\ \varnothing, & \text{otherwise} \end{cases} \right|}{\left| \bigcup_{\sigma \in A_i} \begin{cases} \sigma, & \hat{t}(\sigma,k) = 1 \\ \varnothing, & \text{otherwise} \end{cases} \right|}$$

5.1.2.1.1.5. Incorrect is Incorrect Estimator

The correct estimator error is a measure of the chance that the systems will evaluate a question incorrect if it is indeed incorrect. This is in fact the Bayesian probability that the system evaluates a question false given that it is false.

$$IE(\sigma_i) = p(s(\sigma_i) = 0 | \hat{t}(\sigma_i) = 0)$$

This metric may be derived empirically from a data set of answers A_i.

$$p(s(\sigma_i)=0|\hat{t}(\sigma_i)=0)=\frac{\left|\bigcup_{\sigma\in A_i}\begin{cases}\sigma, & \hat{t}(\sigma,k)=0 \text{ and } s(\sigma,k)=0\\ \varnothing, & \text{otherwise}\end{cases}\right|}{\left|\bigcup_{\sigma\in A_i}\begin{cases}\sigma, & \hat{t}(\sigma,k)=0\\ \varnothing, & \text{otherwise}\end{cases}\right|}$$

5.1.2.1.2. Example

The following is some simple sample data upon which the calculations will be performed in order to demonstrate the various statistical measures described above.

5.1.2.1.2.1. Data

Answer to Question q1	**Tutor $\hat{t}$**	**Automated *s*.**
σ_{11}	1	0
σ_{12}	1	0
σ_{13}	1	0
σ_{14}	0	1
σ_{15}	1	1
σ_{16}	0	0
σ_{17}	0	1
σ_{18}	0	1
σ_{19}	1	0
σ_{110}	1	1
σ_{111}	1	1
σ_{112}	1	1
σ_{113}	0	1
σ_{114}	0	0
σ_{115}	1	0

Figure 5-2 Sample Question Results

5.1.2.1.2.2. Error Measures

In order to illuminate the calculation process, here are the defined error measure which can be derived from the above test data.

Error Measure	Value
$\hat{t}(\sigma,k) = 0$	6
$\hat{t}(\sigma,k) = 1$	9
$s(\sigma,k) = 0$	7
$s(\sigma,k) = 1$	8
$\hat{t}(\sigma,k) = 1$ *and* $s(\sigma,k) = 0$	5
$\hat{t}(\sigma,k) = 0$ *and* $s(\sigma,k) = 1$	4
$\hat{t}(\sigma,k) = 1$ *and* $s(\sigma,k) = 1$	4
$\hat{t}(\sigma,k) = 0$ *and* $s(\sigma,k) = 0$	2
Correlation	**-0.30**
FP()	**4÷8=0.50**
FN()	**5÷7=0.71**
CE()	**4÷9=0.44**
IE()	**2÷6=0.67**

Figure 5-3 Error Measures Derived from Sample

The five black box metrics to be used have already been outlined. That discussion gave a broad definition as well as a mathematical formulation, which could be used to derive the values from a given data set. A brief 'practical' interpretation is given now, of each of these metrics, to show how they would be used and what the implications of each of the values have.

The correlation metric gives a general measure of similarity of the performance of the two systems. Both correct and incorrect responses are taken into account. A value of 1 implies perfect correlation i.e. the two systems are operating identically. A value of 0 implies no correlation, as would be produced by two independent systems operating entirely randomly.

A value of -1 implies one system is operating as the inverse of another. The automated system would have to disagree with every human response to obtain this score.

The false negative and false positive metrics take the point of view of the user who is looking at the output of the automated assessment system (who has no knowledge of the true truth values of the inputs) and who wishes to assign confidence levels to the output the system is producing. A false negative value of 0 means if the system deems a sentence incorrect the user can have 100% confidence in its assessment. A false negative value of 1 means if the system deems a sentence incorrect, it can be assumed the system is in error and the sentence is truly correct. Conversely a false positive value of 0 means if the system deems a sentence correct the user can have 100% confidence in its assessment. A false positive value of 1 means if the system deems a sentence correct, it can be assumed the system is in error and the sentence is truly incorrect.

The correct/incorrect estimator metrics take the point of view of a user passing a data set in to the system (who has full knowledge of the true truth value of the input statements) and who wishes for some measure of how the system will respond to them. A correct estimator value of 1 means a correct statement fed to the system will be deemed correct. A correct estimator value of 0 means a correct statement fed to the system will be deemed incorrect. Conversely an incorrect estimator value of 1 means an incorrect statement fed to the system will be deemed incorrect. An incorrect estimator value of 0 means an incorrect statement fed to the system will be deemed correct.

A further black box metric, the absolute mark, was also recorded to give some measure of the distribution of correct and incorrect responses, which relates in turn to the difficulty of the questions. This aspect is not reflected in any of the above metrics, intentionally in order to allow potential comparison between distinct data sets; a variable that would otherwise systematically bias a set of results upon the distribution of responses within a specific data set.

5.1.2.2. Glass Box Metrics

The glass box metrics concern the implementation of the system and as such relate to many of the facets of the system discussed in Chapter 3. Note as the glass box metrics relate to the knowledge base itself, and as will be explained the same knowledge base was used on both experiments, there is therefore only one set of glass box metrics.

Net complexity is a measure of the sophistication of the knowledge base necessary to capture the form and variability of potential correct statements. The number of nodes necessary was seen as a good reflection of this property. This measure could be further enhanced by examining the number of links; the number of links to number of nodes ratio and the relative distribution of compound, abstract and evidence nodes. These however were not used in this case.

Knowledge reuse is a measure of the extent to which components of the knowledge base for a particular question are also used in other questions. Given the hierarchical tree type structure of the representation of a question, where the trunk of the tree corresponds to a single question, knowledge sharing can be envisaged as a tangling of the branches. Experimentally this is measured as the proportion of nodes for a particular question which are also part of the knowledge structure for other questions.

The difficulty that the tutor has in producing a good model of the semantics of a question is crudely reflected by time taken to create the model. Note this is only a very approximate measure as not all questions perform to the same standard. The acceptable standard is determined by the tutor on creation of the knowledge base and this itself can only be assessed during creation if the data set has been pre-marked. It does however give a general indicator of the effort required by the user which can be assessed in context, once measures of the performance of the model are available.

The final glass box metric under which the system will be assessed is the computational time taken to mark a single question across the entire data set. In practice this is taken as the average time across both experiments and their respective data sets. As an absolute figure the value is of little use as it is entirely dependant upon the processor upon which the system was run. (In this case the was an Intel 468 66MHz DX2 processor). However as a relative term it gives a sophisticated measure of the complexity of the network which takes into account the utilisation of the various components of the network which is in turn dependant upon the statistical distribution of the sentence types. To clarify this point: sentence resolution within this system is functionally a parallel process. However, this is being emulated here by a serial processor. Processing time will therefore correlate strongly with the complexity of the network; something that would not necessarily be true if a parallel process was implemented. Further, the tree modelling a particular question may comprise several branches some considerably more complex than others. In practice perhaps only one response utilises one of these more expensive branches. A processing time measure would

reflect this under-utilisation, which is apparent through the disproportional statistical distribution of node activation across sentences, an aspect that simpler measures of net complexity (such as node count) miss.

5.1.3. Experiment 1 - Retrospective Experiment

5.1.3.1. Design

The first experiment is to investigate the expressiveness and sophistication of the knowledge representation formalism; simply is it possible to capture the form and variety of correct responses to a question, and is the proposed implementation of the decision procedure capable of making the distinction between correct and incorrect statements? All questions are pre-marked by the tutor, and these marks can be correlated with system performance providing constant feedback to the user. A procedure of iterative knowledge base development is proposed, which utilises the feedback provided to direct the development of the knowledge base. In the analysis of the results a qualitative analysis of the idiosyncratic complexity of the individual questions shall be refrained from. Instead, attention shall be focused on the objective quantitative analysis of performance of the overall system under the various applied metrics.

5.1.3.2. Procedure

1. The tutor set 20 questions of a general programming nature which in the tutor's opinion could be answered satisfactorily with a single sentence reply.
2. The questions were presented over the network to the students and the responses were concatenated into a central database file.
3. Replies were hand marked by the tutor and the marks recorded.
4. Using the tutor's own intuition a knowledge base is constructed which attempts to capture the sense and the variety of language which was used to express a valid reply. At this stage the interfaced Roget's thesaurus can be used to help anticipate the variety of language that could be used.
5. Each student reply is presented to the knowledge base one at a time and the automatic marker's decision is recorded.
6. For each student reply, the tutor's mark is compared with the knowledge base decision. The two are correlated with each other to give a measure of similarity, and exceptions recorded.

7. The knowledge base may then be edited by the tutor in order to better capture the exceptions and so increase the correlation rates. Return to step 5 until acceptable model of the human marker's performance achieved.

For specific details of the process used to collect of the student responses see Section 4.1.2.1. For specific details of the process the tutor used to define the knowledge base see Sections 4.1.2.2 and 4.4.

For examples of the Questions, Answers and Sample knowledge bases see Appendices A-C.

5.1.3.3. Results

5.1.3.3.1. Absolute Mark

Question Number	Tutor Mark	Automated Mark
q1	0.75	0.63
q2	0.79	0.56
q3	0.97	0.97
q4	0.95	0.92
q5	0.52	0.33
q6	0.97	0.89
q7	0.89	0.86
q8	0.84	0.63
q9	0.54	0.7
q10	0.05	0.05
q11	0.86	0.51
q12	0.54	0.43
q13	0.43	0.38
q14	0.37	0.41
q15	0.40	0
q16	0.24	0.4
q17	0.48	0.67
q18	0.46	0.43
q19	0.41	0.38
q20	0.02	0.02
Average	0.55	0.48

Figure 5-4 Averages Experiment 1

The above table shows that the set questions were of medium difficulty as evidenced by the fact that the average score was around the 55% mark. The automated marker consistently marked fewer responses correct than the human marker, which would hint towards the conclusion that the main cause of error is omission, in other words the inability of the network to fully generalise. However, more detailed analysis is necessary before this may be ascertained for definite. There seems to be a wide distribution of question difficulties with 97% of students getting the easiest question correct and only 2% of students getting the most difficult question correct. On average 55% of students got each question correct.

5.1.3.3.2. Correlation

Question Number	Correlation %
q1	95
q2	80
q3	98
q4	76
q5	79
q6	82
q7	72
q8	90
q9	82
q10	84
q11	73
q12	93
q13	96
q14	84
q15	82
q16	93
q17	88
q18	73
q19	91
q20	66
Average	83.85

Figure 5-5 Correlation Measure Experiment 1

Using this technique a good model (average correlation 85%) could be produced for each question in less than 5 minutes. It is the authors subjective opinion that with slightly longer construction time correlation rates approaching 100% could be achieved in almost all cases. In order to approach these higher correlation rates, many of the *fringe* answers need to be incorporated into the knowledge base. Interestingly many of the problems in incorporating the last few answers were attributable to spurious decisions on the part of the marker rather than genuine difficulties in extending the knowledge base.

The knowledge base captured over 72% of all questions (the minimum), with performance topping out at the 98% level. Note that there is no noticeable correlation between questions that are objectively hard (i.e. few students get the correct answer) and questions that the knowledge base has difficulty modelling (i.e. question getting low correlation rates). This would seem to imply that it is not necessarily difficult to model *difficult* questions.

5.1.3.3.3. Error Analysis

Question Number	False Positive	False Negative	Correct Estimator	Incorrect Estimator
q1	0.00	0.15	0.89	1.00
q2	0.00	0.22	0.79	1.00
q3	0.00	0.09	0.92	1.00
q4	0.00	0.02	0.69	1.00
q5	0.00	0.07	0.75	1.00
q6	0.06	0.12	0.98	0.89
q7	0.00	0.30	0.79	1.00
q8	0.00	0.04	0.67	1.00
q9	0.00	0.14	0.94	1.00
q10	0.00	0.16	0.33	1.00
q11	0.00	0.21	0.93	1.00
q12	0.00	0.08	0.88	1.00
q13	0.00	0.06	0.78	1.00
q14	0.00	0.04	0.71	1.00
q15	0.00	0.03	0.77	1.00
q16	0.00	0.09	0.90	1.00
q17	0.00	0.10	0.85	1.00
q18	0.00	0.11	0.86	1.00
q19	0.00	0.16	0.75	1.00
q20	0.33	0.00	1.00	0.98
Average	0.02	0.11	0.81	0.99

Figure 5-6 Error Analysis Experiment 1

On a fine grained analysis of the error types it is apparent that there is an extremely low level of false positive errors. This means that if the system deems a sentence correct it almost certainly is correct. That is, there are few instances where the network incorrectly attributes 'correct' to an incorrect answer, in fact just two questions (question 6 and question 20). The

system's conclusion that a particular question is incorrect is to be less trusted; a false negative score of 0.12, with a relatively diverse distribution of scores over the questions.

The positive estimator metric gives an error around the 75% mark showing that the system concurs with about three quarters of correct answers. The incorrect estimator rate is very high 99%, again reflecting the fact that the errors of over generalisation are rare.

To illustrate the distinction between the two families of error metrics, False Positive and Incorrect Estimator, for example, it is worth paying particular attention to the results of question 20. For question 20 only two students supplied the correct answer. The automated system identified these two, but also falsely identified one other. Both FP and IE metrics reflect the notion of over generalisation, for FP a high value indicates over-generalisation and for IE a low value indicates over generalisation. The metrics however report this error from different perspectives: the estimator metric from the students' perspective, the false estimators from the system's perspective. From the system's perspective three false reports were produced, but a significant number of these were incorrect. The false negative metric therefore records a high error, 33%. However, from the students' perspective 62 students got the question wrong, and 61 of these students got the result they deserved, that is approximately 98%.

Note for this unusual question (20) there was not a single instance of under generalisation the FN and CE metrics therefore both returned their extreme values: 0% and 100% respectively.

5.1.3.4. Conclusion

Experiment 1 was essentially a test of the expressive sophistication of the knowledge architecture. That is, was it possible to construct a structure which could discriminate between right and wrong answers? The above results seem to confirm that this is possible, to a reasonably high degree of accuracy considering the limited construction times. However, an obvious danger when constructing these knowledge bases is that we are arbitrarily discriminating between right and wrong answers, whereas we hope we are synthesising a knowledge-base which truly embodies generalised elements of the problem domain. Further, this experiment does not prove the system is useable as in normal scenarios we would not have the answers pre-hand marked.

The most interesting result is that the system proves extremely reliable in its evaluation that a particular question is correct, less so that it is incorrect. This has obvious implications for any real-world application of the software. There is an implicit safety mechanism in the application of a system with low false/positive but high false/negative errors if we can assume that students will partially verify the conclusions of the system. That is, if students note that a question they believe correct has been marked wrong, it is in their interests to chase up the matter. A hypothetical system that has high false-positive but low false-negative errors is less secure, in as much as, it is not in the students interest to chase up a question they believe wrong but has been marked correct. As an interesting psychological aside, it has been noted at The Nottingham Trent University, where automated assessment for areas other than NLP has been in use for some time, that students are (probably quite sensibly) far more diligent in their validation of marking if led to believe that the script has been automatically marked.

5.1.4. Experiment 2 - Blind Experiment

5.1.4.1. Design

There is no obvious method for determining the arbitrariness of a constructed knowledge base at face value. However to get some measure of the generality of the knowledge base we could apply constructed bases against new unseen data and compare correlation rates. This is what has been done here. Knowledge bases constructed in Experiment 1 were applied to new data (actually the responses of the following year's students, again see Section 4.1.2.1). This data set was then hand marked and coverage rates compared.

5.1.4.2. Results

5.1.4.2.1. Average Mark

Question Number	Tutor Mark	Automated Mark
q1	0.80	0.43
q2	0.63	0.33
q3	0.98	0.69
q4	0.96	0.98
q5	0.41	0.18
q6	0.71	0.63
q7	0.94	0.71
q8	0.76	0.61
q9	0.51	0.20
q10	0.12	0.14
q11	0.90	0.63
q12	0.39	0.31
q13	0.39	0.22
q14	0.37	0.25
q15	0.33	0.27
q16	0.26	0.18
q17	0.31	0.37
q18	0.45	0.29
q19	0.52	0.35
q20	0.06	0.02
Average	0.51	0.37

Figure 5-7 Averages Experiment 2

The general distribution of correct and incorrect answers share many features of the previous experiment. Note, however, that although the average mark the students obtained is marginally greater for this set of students, the automated marker gave a lower score. This would seem to indicate that the coverage is not as good as the previous data set.

5.1.4.2.2. Correlation

Question Number	Correlation %
q1	52
q2	68
q3	72
q4	89
q5	48
q6	74
q7	66
q8	69
q9	41
q10	52
q11	73
q12	68
q13	54
q14	63
q15	82
q16	46
q17	82
q18	71
q19	59
q20	53
Average	64.1

Figure 5-8 Correlation Measure Experiment 2

When the knowledge base created in the previous phase was applied to a blind set of data an average coverage rate of 65% was obtained. It is to be remembered that this is a metric on which a random decision procedure would score 0%. Considering the present limitations of the system this is considered very encouraging. This data in itself gives no indication of the possible source of increased error.

5.1.4.2.3. Error Analysis

Question Number	False Positive	False Negative	Correct Estimator	Incorrect Estimator
q1	0.00	0.25	0.72	1.00
q2	0.00	0.17	0.52	1.00
q3	0.00	0.11	0.82	1.00
q4	0.03	0.28	0.74	0.97
q5	0.00	0.14	0.73	1.00
q6	0.00	0.17	0.26	0.82
q7	0.11	0.12	0.63	1.00
q8	0.00	0.29	0.50	1.00
q9	0.09	0.21	0.52	0.84
q10	0.06	0.24	0.67	0.97
q11	0.00	0.53	0.72	1.00
q12	0.00	0.26	0.69	1.00
q13	0.00	0.26	0.64	1.00
q14	0.00	0.18	0.58	1.00
q15	0.00	0.13	0.63	1.00
q16	0.00	0.32	0.69	1.00
q17	0.00	0.29	0.61	1.00
q18	0.09	0.11	0.60	0.84
q19	0.16	0.22	0.59	0.70
q20	0.00	0.34	0.71	1.00
Average	0.03	0.23	0.63	0.96

Figure 5-9 Error Analysis Experiment 2

Again, the false positive metric is notable for its extremely low error rate. This implies that even though the knowledge base is being applied to new unseen data, over generalisation has still not occurred. The false negative error rate has obviously increased, simply about 10%

more of the questions deemed incorrect are now really correct. Aspects of the knowledge base are therefore not generalising to the new data set.

The correct rate has dropped by about 10% in line with the false negative rate. The incorrect estimator rate has dropped by only a few percent following the trend implied by the still low false positive rate.

5.1.5. Glass Box Metrics - Knowledge Base Construction

Question Number	Net Complexity (number of nodes)	Knowledge Reuse (% node sharing)	User Time (mins - approx.)	Marking Time (seconds)
q1	41	70	8	0.31
q2	29	24	5	0.11
q3	38	68	7	0.22
q4	25	49	2	0.34
q5	12	82	3	0.27
q6	10	94	6	0.09
q7	44	68	4	0.25
q8	49	29	3	0.35
q9	28	74	8	0.17
q10	22	30	5	0.19
q11	19	82	2	0.18
q12	39	73	3	0.29
q13	19	86	4	0.32
q14	26	42	3	0.20
q15	43	55	5	0.30
q16	16	75	1	0.14
q17	26	58	2	0.11
q18	36	91	8	0.26
q19	40	69	2	0.29
q20	49	64	3	0.32
Average	30.55	64.15	4.2	0.24

Figure 5-10 Glass Box Metrics

The networks necessary to mark these scripts are not overly complex, all questions being under 50 nodes in size. There is good re-use of resources with on average 64% of a question's nodes composition being shared by other questions. All knowledge bases were created in less that 10 minutes with the average being 4.2 minutes. Note this is an arbitrary measure, in a sense, as the tutor himself decides when to stop refining the network and so completely determines the creation time. The marking processing time is useful as a relative measure only, in that it gives a strong indication of the computational complexity of the different networks.

5.1.6. Summary

	Correlation	False Positive	False Negative	Correct Estimator	Incorrect Estimator
Retrospective	85	0.02	0.11	0.81	0.99
Blind	65	0.03	0.23	0.63	0.96

Figure 5-11 Experiment 1 & 2 Result Summary

A knowledge construct which provides an approx. 85% coverage will typically be about 30 nodes in size. The time needed to mark is negligible: a single question answered by 64 students takes approximately 1 second to mark. At this stage the time required to both construct and mark a set of student sentences is slightly greater than that required by the tutor to do so by hand. However it is anticipated that long term benefits (in terms of man hours) would accrue from the re-use of questions over different students. Probably the best mode of use would be the random selection of questions from a stable pre-configured bank of questions and knowledge bases.

The knowledge constructs generated necessarily embody information for both the KR and NLP processes in the one tree-like structure. A distinction, if one is to be made, is one of degree rather than absolute. Knowledge pertinent to language processing, to do with the status and category of a word, is held low down in the tree structure. Knowledge of a higher level which embodies facts etc. is held high up in the tree. As such it is expected that the low

level nodes will be reusable across questions especially within the same domain. A level of cross domain reusability is provided through the thesaurus.

The automatic marker systematically marked fewer answers correct than the tutor. It is also noticeable that very rarely does the system mark an incorrect response correct. In a real application we can be reasonably certain (in our tests 98%) that a reply marked correct will indeed be correct. We can legitimately focus future efforts on capturing those unexpected correct replies.

When constructing knowledge bases our major problem is anticipating the variety of language which can be used to reply to a question. We have partially overcome that here by insisting on two runs of the system. The first on a set of prototype information to help generate these anticipations and a second true run. The next logical step in reducing the time required to generate the knowledge base and to improve coverage rates is to enhance the use of secondary knowledge sources and/or implement learning algorithms.

5.1.6.1. Problems - Sources of Error

As noted, errors within the systems are apparently of an under generalisation nature. There are a number of possible sources of this.

5.1.6.1.1. Spelling Errors

It is the nature of this data, which is produced under stressful exam type conditions with an imposed time limit, that noise will be introduced, manifestly in the form of spelling errors. Individual evidence nodes within the network only become active if an appropriate evidence word is discovered within the input string. Spelling errors mean that many strings that are intended to imply the meaning of a particular evidence node simply do not match any of the attached evidence words. At present a limited number of expected spelling errors is added to the lists to counter this problem, but this is a far from ideal solution.

5.1.6.1.2. Knowledge Base Sophistication

This is a somewhat nebulous point, but basically alludes to the fact that in some cases the knowledge base is not sufficiently sophisticated enough to capture all possible manners in which a particular answer may be expressed. This may cover such things as: synonym groups that are not wide enough in their membership; disambiguation mechanisms that have ill-defined contexts; missing idiomatic phrases; or insufficient fact mappings to composite nodes. The best way to improve this is to draw in secondary knowledge sources to help both to better

anticipate the variations of expression (e.g. thesauri) and to help compose the composite sense of the answers to be modelled (e.g. domain specific source books and encyclopaedias).

5.1.6.1.3. Syntax

No syntactic component is present within the current system. From the results given above the lack of a syntactic component does not appear to be contributing significantly to the overall error. This can be reasoned as follows. Syntactic information provides mainly clues towards word role within sentence context. The present system takes no notice of syntax (hence role), in effect all words are interpreted in all roles. Such a strategy will cause words to be interpreted within the wrong role and hence will give rise to incorrect statements being interpreted correctly. These are errors of over generalisation, from which, patently, the system does not suffer.

The reason role interpretation plays such a little part in the systems error is probably due to the strong contextual constraints implied by the question answering paradigm, and the semantic constraints of the component items.

"Pascal is a high level language"

Is a sensible combination of the appropriate terms *"Pascal"* *"high-level language"*.

"A high level language is Pascal"

Is almost without meaning, and unlikely to be presented by a student in answer to the question "What sort of language is Pascal?"

Further, if a student knows the correct terms that are expected within a correct answer, it is more than likely he/she will get them in the correct relationship with each other.

It must also be remembered that a full syntactic parse is virtually impossible to perform reliably on such data given the inherent noise within the sentences.

5.1.6.1.4. Fuzzification

At present the network uses a Boolean activation passing scheme. In that all questions are to be eventually marked either entirely correct or entirely incorrect this is convenient as it

leaves no room for ambiguity in the interpretation of the result. However fuzzification could be useful in enhancing the robustness of the processing mechanism in at least two ways.

Firstly, by accepting degrees of activation it is possible to model degrees of set membership. For synonyms in particular, this introduces a level of flexibility in the definition of synonym sets and could be the key to introducing a degree of metaphor into the system.

Secondly, fuzzification is a useful tool for resolving sense ambiguities in the manner outlined in the chapter on knowledge architecture. However, due to the limited scope of the semantic variation in such domain specific tests, this is not deemed an excessive contributory factor.

5.1.6.1.5. Logic

On the surface, logic interpretation would seem vital towards a functioning system for how else will you distinguish *"Pascal is a high level language"* from *"Pascal is not a high level language"*. Statistically, however the lack of such processing would seem to contribute little to the overall error. At some point however it will be necessary to introduce some form of logical processing to cope adequately with questions such as: *"Define NON-RESIDENT" in COBOL terms by completing the sentence 'NON-RESIDENT segments of COBOL code are...'?"* in particular (see Chapter 2).

Pure logic processing must obviously presume a fully developed syntactic process, for it is impossible to sensibly apply logical operators unless the roles and relationship of the key semantic components can be identified. This is clearly impossible.

If a feasible strategy of incorporating logical type operators within the input strings, an alternative more robust system must be developed, using token proximity for logical binding perhaps.

5.1.6.1.6. Anaphora

Due to the small scope of the answers supplied (usually a single sentence) and the fact that the given sentence is constrained heavily by the preceding sentence, the incidence of anaphora is very low. In cases where it does appear the constraint and limited scope make it easy to resolve.

5.1.6.2. Quantitative Analysis of Source of Error

In order to identify areas for further development it is imperative that a quantitative analysis of the contribution that the various aspects makes towards the overall error is made. This will maximise the likelihood that improvements made to this sector of the processing will result in a better global performance of the system. This was done as follows:

An error of under generalisation means that a given question node has not been activated. All errors that lead to the non-activation of a question node can be classified into one of two types. Those where the appropriate evidence nodes have not been activated; either the evidence nodes do not exist or the evidence and evidence strings do not match. These are errors of evidence. And those where although the correct evidence nodes have been activated, there are not enough paths that can lead to the activation of the correct node; these are errors of insufficient or inappropriate net complexity.

All of the above sources of error fall into one of these two types. Anaphora and spelling errors are errors of evidence; logic, fuzzification, syntax and knowledge generation are errors of net complexity.

In order to give more information about the error types a few simple tests are possible.

Firstly we can compare the percentage of evidence node activation within the two data sets. This is done as follows:

1. Analyse the tree for each question, by propagating a test signal downwards from the question node.
2. Identify all activated evidence nodes for this question, and use this to define the evidence set for this question (note these sets are non-exclusive between questions due to the knowledge sharing strategy employed).
3. Run all questions from this data set across the network and record the percentage of evidence nodes from each questions set that are activated by each answer.
4. Compare average percentage activation across the Blind and Retrospective data sets.

This produces the following results:

	Blind	Retrospective
Average evidence set activation	20.3%	15.4%

Figure 5-12 Average Evidence Set Activation

Clearly the blind set has a greater percentage of its evidence set activated for each question. There are two possible explanations for this. Either the evidence nodes for concepts used to express answers in the newer retrospective data set do not exist within the network. Or the evidence strings attached to the network do not match the precise formulation of the words to be found within the input strings, the most likely source for this being spelling errors.

The first explanation is considered unlikely as the 64 answers within the first data set necessitate a wide lexical coverage, it is unlikely that the few missing items will account for the relative 25% difference in set activation. Further the domain specific nature of the questions means that that the lexical variety allowed within many of the questions is actually quite narrow.

Conversely, when it is considered that the knowledge base contained many evidence strings to specifically cater for the spelling mistakes occurring in the first data set, yet contained non that covered those original mistakes occurring in the second data set, this seems a viable explanation of the difference in set averages.

In summary the above data implies that the difference in evidence node activation is attributable to either the second data set requiring new evidence nodes to correctly model the sense or to the second data set requiring a more robust matching mechanism to match input strings against the existing evidence strings. For the reasons stated above the second cause is deemed the most contributory, although of course both have an input. In essence the contribution of the two factors is determined by the increase in diversity to be found in the second set. If the second set exhibits a greater semantic diversity then the first factor is most important; if the second set exhibits greater spelling diversity then the second factor is most important.

Note this does not prove that knowledge base sophistication does not contribute, it will of course and will probably be the main source of error. It does however prove that errors of evidence contribute significantly to overall error.

In an attempt to quantify the increased spelling diversity in the second data set a further experiment is possible. If the evidence strings attached to evidence nodes are divided into strings that reflect the true form of a lexical item and those that reflect spelling variants upon that item, the two sets may be distinguished. If all spelling variants are *switched off* it would be possible to run the data sets over the knowledge base, re-marking the questions, and evaluate the contribution that spelling adjustments make on the two distinct sets.

	Retrospective	**Blind**
Correlation with spelling correction turned off	72.4%	61.2%

Figure 5-13 Correlation with Spelling Correction Off

The first data set performed much lower with the *hard-coded* spelling correction turned off, roughly a drop of 11%. The second data set also performed less well, but significantly less so, only 3%. This implies that the spelling corrections hard-coded within the knowledge base reflect more of the spelling errors to be found in the first data set than the second. In other words diversity of spelling between the first and second data sets is quite high. Spelling therefore must contribute in part to the overall performance differences between the first and second data sets.

As a final caveat all the error sources identified above contribute in part to the errors in sentence processing, and certainly for full interpretative systems errors of logic and semantic comprehensive must be the most significant. However in the context of the task of automated assessment attempted here, the above data seems to imply that errors of spelling, if not necessarily the most important, are certainly significant enough to warrant further investigation.

5.1.7. Conclusions

From the results of the retrospective experiment (85%) it is clear that it is possible to generate knowledge bases which model well the semantic coherence between correct answers. The knowledge base creation model of the retrospective experiment is however unworkable in the real world as it presupposes the answers are hand marked (which would make the automated system redundant).

The knowledge base creation model from the blind experiment is however workable in practice. It simply uses a model generated from a previous data set. The results are obviously not as good (65%) as the retrospective test, but are encouraging in that it seems that the majority of the information embodied within the knowledge base is still valid for the new data set. The lower correlation rates are explained by an increase in the variability in content and structure of the new data set.

From an analysis of the sources of data it was deemed that logic, fuzzification, anaphora and syntax are, at this stage, either contributing insufficiently to the error or are two complex to resolve. Knowledge base sophistication and spelling errors, on the other hand, appear to contribute significantly to the overall performance of the system.

Automated spelling correction is an interesting problem, and although no perfect algorithm yet exists for its resolution, it is already very heavily represented and researched in the literature. Further it is somewhat tangential to the main thrust of this thesis.

The issue of knowledge base sophistication and especially the research of methods of automatically inducing knowledge base models are believed to be a more natural extension of the work presented so far. These, learning algorithms if successful will either improve the robustness of the matches, or better capture the variability of language use. Such learning algorithms will also address the problem that, at present, there is considerable overhead and expert knowledge implied in the knowledge acquisition phase. More specifically, as has been described earlier in this thesis, the aim will be to induce network structures from the statistical properties of the student text, and that these structures will provide a perceptual augmentation function which lead to increased performance of an automated sentence judgement process.

It is these areas that shall be addressed in the final section of this thesis.

5.2. Algorithm Evaluation

5.2.1. Introduction

The nature of the networks that are produced by learning algorithms differs considerably to that of hand-crafted networks. The algorithmically produced networks are incapable of discriminating between correct and incorrect student responses on their own. This is because the algorithms used are unsupervised and therefore have no knowledge of the correctness of specific responses. This raises the thorny issue of how to evaluate the effectiveness of the produced networks.

The approach that has been taken is to pursue the perceptual information-enriching metaphor alluded to in both the previous section 5.1 and Chapter 3. What is required is a decision procedure that is capable of automating the sentence-judgement operation. This decision procedure obviously needs input in order to make its judgement. Ideally the procedure needs to be able to take this input in both a raw atomic form (for this application the list of words occurring in the sentence is a good approximation to this) and a perceptually enhanced information source (which will be provided by the activation passing network). On the assumption that the perceptually enhanced information will contain new and "useful" information, the hypothesis is that the decision procedure will perform better using the perceptually augmented information source containing information at different granularities.

The decision procedure chosen to perform this function is Latent Semantic Analysis. In the next section the necessary background is presented to understand how LSA is to be applied.

5.2.2. Latent Semantic Analysis

5.2.2.1. Introduction

Latent Semantic Analysis (henceforth LSA) is an extension of another text analysis technique, Latent Semantic Indexing (LSI). Both are rooted in the complex and powerful mathematical technique of Singular Value Decomposition (SVD).

LSA is a novel technique that has come to the forefront over the past few years. The application for which it is best known is the automated assessment of student essays (Landauer et al (1997)). It would therefore be impossible to properly conclude this thesis without an investigation into its effectiveness within this specific problem domain, i.e. the

automated assessment of single sentence/paragraph responses. The fact that it satisfies the criteria defined above, that is to automate the decision procedure for sentence judgement and to accept and compare input data at different perceptual granularities, is an added bonus.

5.2.2.2. Theory

Latent Semantic Analysis is summarised by Landauer et al (1997) as a

> *"corpus-based statistical method for inducing and representing aspects of the meaning of words and passages reflected in their usage".*

It is basically a method deriving an N-dimensional vector representation for both "words" and "documents". The point of this being that words and documents can then be compared and constructed using standard techniques from linear algebra. LSA is unusual in that the N-dimensional vector representation of both words and documents takes place within the same hyper-dimensional space. This means that rather than just words being compared against words and documents being compared against documents, words can be compared against documents.

Let us start by looking at the mathematical technique that underpins LSA, that is Singular Value Decomposition.

Weisstein (1998) defines SVD as:

An expansion of a real M x N matrix by orthogonal outer products, according to:

$$A = \sum_{k=1}^{K} s_k \mathbf{u}_k \mathbf{v}_k^T$$

where $s_1 \geq s_2 \geq \ldots \geq 0$

$$K \equiv \min\{M, N\}$$

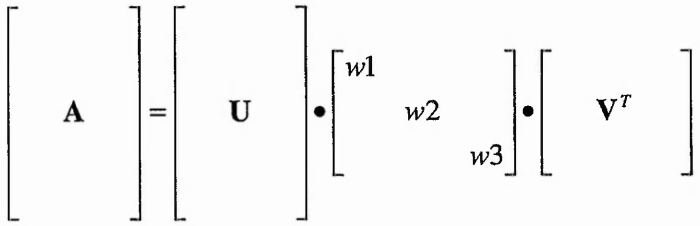
and

$$\mathbf{u}_k^T \mathbf{v}_{k'} = \mathbf{v}_k^T \mathbf{u}_{k'} = \delta_{kk'}$$

Here δ_{ij} is the Kronecker delta and A^T is the matrix transpose.

What does this mean? Basically, any MxN matrix **A** whose number of rows M is greater than its number of columns N, can be written as a product of an MxN column-orthogonal matrix **U**, an NxN diagonal matrix **W** with positive or zero elements (the singular values) and the transpose of an NxN orthogonal matrix **V**.

The following diagram should make this clearer:

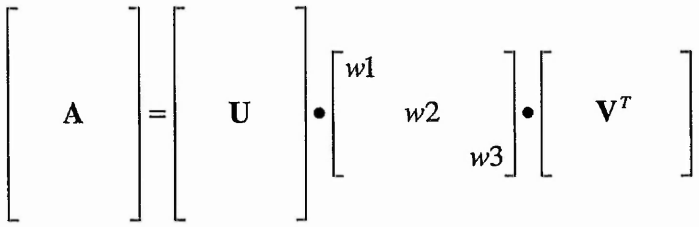


There are several useful applications of this technique. The most common as quoted by Press et al (1996) is as a method for solving least-squares problems. As such it can be used to optimally solve both overdetermined and underdetermined sets of linear equations. It can also be used to construct an orthonormal basis and to approximate matrices. It is in this manner that the technique is applied to Latent Semantic Analysis.

Consider the following. Construct a matrix **A** as a document term matrix where each row represents individual word types and each column represents individual documents/passages etc. Perform SVD upon this matrix, which produces **U**, **V** and **W** matrices. Under this interpretation:

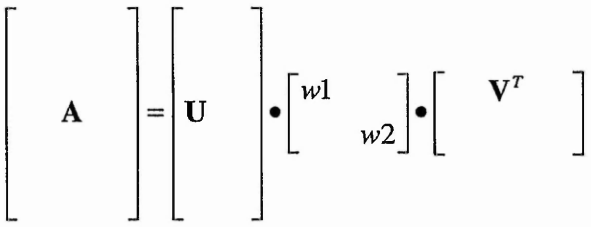
> U – is an orthonormal basis which spans the same subspace as the column vectors in A.
>
> W – is a diagonal matrix, the elements of which essentially gives a weighted importance to their respective basis vectors in U.
>
> V^T – is a matrix of row vectors, which when interpreted in the weighed subspace of UW can reconstruct the original column vectors of A. (Note row vectors of V^T are column vectors of V.)

An important quality of the decomposition is that if a particular diagonal element of W is small, i.e. $w \approx 0$, then this term may be removed from the W matrix, which in turn means that the appropriate column terms from U and V can also be removed. So if:



and w3=0

then



Where w3=0 then the equality is preserved, where w3≈0 then the above is an approximation. This is the application of approximating matrices as discussed above.

5.2.2.3. Application

In all applications of applying SVD to document analysis (this means LSA and LSI) the starting data is a document term matrix. Commonly, the rows of this matrix will correspond to terms, which in the simplest case could be simple instances of word type, or could be a more abstract notion of word group. The columns of this matrix will correspond to documents. Depending on the application this could be single sentences, paragraphs or entire essays.

Two principle properties of the above decomposition mechanism are utilised in these applications:

5.2.2.3.1. The reduction of document and term to common dimensionality.

Where every document was originally an M-dimensional vector and every term was an N-dimensional vector, after decomposition and then the removal of low w values, it is possible to represent both terms and documents and terms as K-dimensional vectors. This means that

not only can terms be compared directly against documents, but by simple algebraic operations psuedo-documents can be constructed from a collection of terms.

5.2.2.3.2. The reduction of dimensionality.

By removing small w terms the original A matrix is slowly being approximated to. This is because we are trying to span the same subspace with a set of basis vectors of lower dimensionality. There will inevitably be points in the original A subspace that are unreachable by the decomposed approximation. SVD assures us, if we remove the smallest w values first, then the area in A, that is now unreachable, is of the smallest volume. In terms of the application to document analysis there is an implicit assumption or expectation here. That is, it is the semantically coherent attributes that are being preserved in the reduced basis vectors and that it is noise and non-useful information that is being discarded in the now unreachable subspaces of A.

5.2.2.3.3. Text Searching – LSI

LSI is an effective way of finding best matching documents from a given user query. According to Dumais (1991) Latent Semantic Indexing is characterised as:

> *"The LSI approach partially overcomes the problem of variability in human word choice by automatically organising objects into a "semantic" structure more appropriate for information retrieval. This is done by modelling the implicit higher-order structure in the association of terms with objects."*

This "semantic" structure is the reduced, decomposed orthogonal basis discussed above. Once the mechanics of SVD are understood the implementation is trivial:

1. Construct an "indexing" document-term matrix, using documents available.
2. Decompose this matrix using SVD.
3. Reduce the dimensionality to K by removing w terms, smallest first. Documents and terms can now be represented by a K-dimensional vector.
4. An incoming query is specified by a user as a series of terms (ie a sentence, several words in a row), possibly a single term. These terms can be *combined* to produce psuedo-document which represents that query.
5. Best matching documents can be found by *comparing* the pseudo document to actual documents and finding closest matches.

6. New documents can be added later by "folding" them into the matrix. This is a technique that shall not be discussed here.

There are two terms in the above description that were left deliberately ambiguous; these are "combined" and "compared". These processes shall now be considered in greater detail.

5.2.2.3.3.1. Combining terms into a psuedo document

Given that a K-dimensional vector can represent documents and terms, a pseudo-document, which is derived from a set of terms, is constructed as the average of the vectors for all its constituent terms.

$$\mathbf{p} = \frac{\sum_i^n \mathbf{t}_i}{n}$$

5.2.2.3.3.2. Comparing documents and terms.

There are various mechanisms for comparing two documents (or terms). In fact virtually all of the methods discussed in the section on syntactic clustering would be appropriate. Specifically all the similarity measures discussed in the section on clustering measures are appropriate. But within the literature Foltz et al (1998), Landauer et al (1997), Landauer et al (1998a) Landauer et al (1998b), it is the cosine measure that seems to be the most commonly used.

5.2.2.3.4. Text Evaluation - LSA

Now that the groundwork has been laid, it should be fairly easy to see how these techniques can be applied to automated assessment. The most frequently cited application (Landauer et al 1997) is the automated assessment of essays. Typically the described procedure is as follows

1. Construct a term-document matrix from all the student essays available using essays as individual documents and individual word types as term entities (optionally apply a stop list first, and optionally apply term-weighting)
2. Normalise the columns.
3. Decompose the matrix using SVD
4. Reduce to K-dimensions.
5. Generate a "correct" target essay.

6. From the words in the target essay generate its pseudo-document; this is simply another K-dimensional vector.
7. Using a few marked essays "calibrate" the marking zone
8. For new unmarked essays – compute the pseudo-document
9. Compare the new essay's vector to the target vector and use the empirically derived calibration to assign a mark.

Landauer et al (1997) have reported considerable success with these techniques. In studies in which short student essays on scientific subjects were analysed, they reported that the correlation of approx 80% between LSA analysis and tutor marking, which proved as accurate as the inter tutor correlation figures.

5.2.2.4. The Issue of Word Order

It is interesting to note that this technique takes no account of word order. The Landauer et al (1997) paper, specifically titled "How Well Can Passage Meaning be Derived without Using Word Order?" concludes:

> *"This paper presents new evidence that a great deal of information about the meaning of passages may be carried independently of word order"*

This contrasts interestingly with Lou & Foxley's (1993) investigations where they also consider a semantic analysis system that does not use a strict syntactic model. In fact they conclude:

> *"The requirement for strict grammatical correctness would form a barrier to success."*

Obviously this is not to say that word order is irrelevant. Clearly the introduction word order into input information would add a new dimension to the input data and would help clarify and disambiguate sentences that would not be fully analysable without it. But given the difficulties that previous researchers have had in introducing this element to LSA and fuzzy logic respectively, it is interesting to note what is possible without it.

5.2.2.5. The Issue of Multiple Senses

It is worth pointing out that the LSA system as it stands is incapable of handling polysemy. As Landau et al(1998b) remark:

"A dynamic contextual disambiguation process can be mimicked in LSA, but the acquisition and representation of multiple meanings cannot"

This is another way of saying that it may be possible to use the output of LSA to disambiguate identified polysemous words but the modelling of polysemy within the model itself it not possible.

5.2.3. Experiment 3 – Simple LSA, The Control

LSA has been identified as the ideal mechanism for evaluating the usefulness of the algorithmically generated networks. In order to perform this evaluation of "effectiveness" a control is needed. The control is to be produced by analysing the performance of LSA when applied to the atomic presentation of data. A simple document-term matrix, where each document corresponds to a single student's response to a question and each term corresponds to a uniquely occurring string, is the basis of the atomic presentation of data.

This experiment has a further justification in that it will be interesting to see how this alternative implementation of an automated assessment procedure performs on the target application of the automated assessment of single sentence student responses.

5.2.3.1. Method

The procedure will closely resemble the technique for automated essay marking described above. The data to be used is the aggregate of the data sets from the two previous experiments, which is approx 120 students answering 20 questions on computer science related questions, all answers between 1 and 3 sentences in length.

1. Construct for each question a term-document matrix from all the student responses available, using individual paragraphs (the entirety of the response for that question) as individual documents and individual word types as term entities. A simple stop list was applied (the, to, which, for, when etc)
2. Normalise the columns.
3. Decompose the matrix using SVD
4. Reduce to K-dimensions, the value of K was derived empirically, i.e. the best performing, in practice this proved to be a dimension of about 20.
5. A small set of pre-marked answers were selected, in this instance 10 were selected.

6. From this set, all correctly marked responses were summed and averaged. This was used as the correct target.
7. Using, from the small sample set, the incorrectly marked responses as well as the correctly marked responses, a similarity threshold was identified. This was the least mean squared separator between all correct and all incorrect responses when using the cosine measure of similarity with reference to the identified target vector.
8. For new unmarked responses – compute the pseudo-document
9. Compare the unmarked responses vector to the target vector. In the cosine measure of similarity falls within the threshold identified in stage (6) then the response is deemed correct. If outside the threshold it is deemed incorrect.

5.2.3.2. Results

The effectiveness of the above procedure was analysed using the standard set of black-box measures researched and defined earlier.

5.2.3.2.1. Averages

Question Number	Mean Human	Mean Simple-LSA
q1	0.77	0.63
q2	0.72	0.56
q3	0.97	0.97
q4	0.95	0.92
q5	0.47	0.33
q6	0.85	0.89
q7	0.91	0.86
q8	0.80	0.63
q9	0.53	0.7
q10	0.08	0.05
q11	0.88	0.51
q12	0.47	0.43
q13	0.41	0.38
q14	0.37	0.41
q15	0.37	0.04
q16	0.25	0.4
q17	0.40	0.67
q18	0.46	0.43
q19	0.46	0.38
q20	0.04	0.02
Average	0.55	0.48

Figure 5-14 Averages Experiment 3

As with the network based automatic marker it is notable that the average score for LSA is slightly lower than the tutor's, perhaps suggesting that it misses more than it incorrectly attributes a correct score. On simple inspection there appears to be a fair distribution of

scores with the LSA marker scoring within the same general range as the human ascribed score, excepting question 15.

5.2.3.2.2. Correlation

Question Number	Correlation %
q1	0.15
q2	0.2
q3	0.22
q4	0.09
q5	0.08
q6	0.6
q7	0.47
q8	0.14
q9	0.47
q10	0.28
q11	0.28
q12	0.93
q13	0.79
q14	0.64
q15	0.46
q16	0.24
q17	0.64
q18	0.6
q19	0.35
q20	0
Average	0.36

Figure 5-15 Correlation Measure Experiment 3

From the correlation figures it is obvious that the LSA system performs considerably worse than both the blind (Experiment 1) and retrospective experiments (Experiment 2) from the activation passing networks system. However a correlation of .37 is still significantly above the zero correlation that we would expect if marks were being assigned randomly. Also it must be remembered that this an automated marking system that is largely un-supervised

requiring little input on the part of the tutor, just the provision of a few marked responses in order to calibrate the system. Contrast this with the laborious process of generating activation passing networks by hand.

5.2.3.2.3. Error Analysis

Question Number	False Positive	False Negative	Correct Estimator	Incorrect Estimator
q1	0.18	0.67	0.82	0.33
q2	0.16	0.67	0.73	0.5
q3	0.04	0	1	0
q4	0	1	0.89	0
q5	0.47	0.44	0.71	0.36
q6	0	0	1	1
q7	0.05	0.57	0.83	0.75
q8	0.16	0.67	0.91	0.2
q9	0.18	0.36	0.78	0.7
q10	0.75	0.04	0.5	0.88
q11	0.12	0.5	0.96	0.25
q12	0.06	0	1	0.91
q13	0.13	0.08	0.93	0.86
q14	0	0.32	0.6	1
q15	0.38	0.17	0.83	0.63
q16	0.5	0.21	0.29	0.9
q17	0.21	0.14	0.85	0.8
q18	0.31	0.08	0.92	0.69
q19	0.38	0.27	0.67	0.69
q20	0.96	0	1	0
Average	0.24	0.29	0.77	0.55

Figure 5-16 Error Analysis Experiment 3

In general the in depth error metrics demonstrate the tendency that should be expected, that is false positives and negatives should be below 0.5 and estimators should be above 0.5. However, it is notable that the extremes of the previous section are not reached. Specifically

the extremely low false positive metric that seemed to characterise the performance of the hand crafted networks is not present.

5.2.3.3. Conclusions

In general the results are low, certainly lower than the figures obtained by the two previous experiments. Also, it is significantly lower than the results that have been published in previous LSA experiments, where tutor correlation rates of 0.80 were reported. This is speculation, but a probable reason for this is the inherently lower dimensionality of the problem domain. The LSA experimental results that have been published come from essay studies. The resulting document-term matrix, before dimensional reduction, is far larger.

Despite this, they are still considerably better than that which would be expected by chance. It must not be forgotten that the automatic assessment model is generated with far less tutor intervention. Only a handful of pre-marked questions are required in order for the Simple-LSA system to extrapolate to a global model of correctness.

5.2.4. Experiment 4 – Perceptually Augmented LSA – The Algorithm Evaluation

Experiment 4 uses the same procedure as the last experiment but on a different input matrix. The input matrix is however calculated from the same input data. The documents for this matrix are, as before, the individual student responses for each question. The term elements, however, now correspond to individual nodes within a network that is generated from the student responses, using the network growth algorithms outlined in the previous chapter.

It is intended as an evaluation of usefulness of the information added by learning algorithms, and therefore compares the "*quality*" of raw atomic data to that produced by the perceptual augmentation process that the algorithmically produced networks supply. We use the term Perceptually Augmented LSA (PA-LSA) to distinguish this system from the system from the previous experiment.

The input data is again the same as for Experiment 3.

5.2.4.1. Method

1. Each student response is presented to the clustering/compositioning algorithm one at a time.

2. The network is allowed to grow in the manner determined by the configuration of the various heuristics.
3. A blank document-term matrix is generated which has as many term elements, as there are nodes within the network.
4. A single response is then presented to the network where each of the atomic words occurring leads to the activation of the appropriate "primitive" node in the network.
5. Activation is allowed to propagate up the network thus activating the appropriate clustered and composite nodes.
6. The activation of each node is then transcribed onto its respective cell in the document term matrix.
7. Singular value decomposition, dimensional reduction etc. performed as in the previous experiments.

5.2.4.1.1. Algorithm Configuration

A compositioning linkage heuristic was used which was a 2 window right-left adjacency
A clustering heuristic was used which was based on the cosine metric of similarity and required a similarity measure of 0.65 or greater, but also used node count as a alpha-cut to approximate to a test for statistical significance.

5.2.4.2. Results

Again the results are generated by comparing a tutor's evaluation of correctness against the PA-LSA system using the black-box metrics.

5.2.4.2.1. Averages

Question Number	Mean Human	Mean Perceptually Augmented LSA
q1	0.77	0.72
q2	0.72	0.65
q3	0.97	0.88
q4	0.95	0.81
q5	0.47	0.38
q6	0.85	0.81
q7	0.91	0.83
q8	0.80	0.79
q9	0.53	0.6
q10	0.08	0.02
q11	0.88	0.79
q12	0.47	0.6
q13	0.41	0.39
q14	0.37	0.37
q15	0.37	0.46
q16	0.25	0.3
q17	0.40	0.41
q18	0.46	0.48
q19	0.46	0.51
q20	0.04	1
Average	0.53	0.56

Figure 5-17 Averages Experiment 4

Interestingly this is the first experiment that has been run for which the automatic marker has attributed a higher average mark to student responses than the tutor did.

5.2.4.2.2. Correlation

Question Number	Correlation
q1	0.5
q2	0.7
q3	0.7
q4	0.46
q5	0.36
q6	0.7
q7	0.48
q8	0.68
q9	0.62
q10	0.57
q11	0.35
q12	0.88
q13	0.75
q14	0.86
q15	0.62
q16	0.53
q17	0.71
q18	0.65
q19	0.7
q20	0
Average	0.55

Figure 5-18 Correlation Measures Experiment 4

The correlation figure is again still less than the results from the activation passing network experiment, but significantly above the correlation figure is results from the third experiment. Specifically, it represents an improvement of 48% over the non perceptually augmented input matrix.

5.2.4.2.3. Error Analysis

Question Number	False Positive	False Negative	Correct Estimator	Incorrect Estimator
q1	0.13	0.38	0.87	0.63
q2	0	0.41	0.82	1
q3	0.02	0	1	0.5
q4	0.75	0.0	0.85	1
q5	0.25	0.38	0.55	0.8
q6	0	0.5	0.97	1
q7	0.08	0.33	0.86	0.43
q8	0.07	0.14	0.98	0.6
q9	0.21	0.16	0.88	0.72
q10	0	0.03	0.33	1
q11	0.08	0.22	0.85	0.56
q12	0.11	0	1	0.86
q13	0.19	0.06	0.93	0.83
q14	0.09	0.05	0.91	0.95
q15	0.28	0.12	0.84	0.79
q16	0.32	0.09	0.73	0.83
q17	0.16	0.13	0.87	0.85
q18	0.2	0.15	0.83	0.82
q19	0.25	0.06	0.92	0.78
q20	0.98	0	1	0
Average	0.2	0.15	0.81	0.71

Figure 5-19 Error Analysis Experiment 4

Again the characteristic low false positive metric from the activation passing networks experiment is not present. The figures do however represent a considerable improvement over the last experiment.

5.2.4.3. A Qualitative and Quantative Analysis of the Grown Networks

It was noted in Section 3.2.3 that the anticipated form and structure of the automatically grown networks would be considerably different to the form and structure of the hand-crafted networks. We are now in a position to be more specific as to what these differences are.

Firstly there is the issue of size. The average HCN was 30.55 nodes in size. Not surprisingly the average AGN was much larger than this, an average of 248.40 nodes. To put this figure in context, the average starting size of the network (i.e. the number of primitive nodes) is approximately 74 nodes. What this means that if the words are aggregated over each question, then a stop list is applied and then primitive spelling correction applied, we end up with about 74 unique nodes to represent the different word types in the input strings. This growth represents a three-fold increase.

The second most obvious difference is the topology of the network. All the HCN are singly rooted hierarchical structures, where the node, which is the tree's root, is the node that is intended to represent, by its activation state, the truth-state of the question. The AGNs are multiply rooted hierarchical structures. Those nodes that are higher up the tree simply represent a higher-level perception of the data that is available in the input string. There are many terminal roots to the tree, on average 57.

Some further comments can be made on the constitution of the AGNs. It has already been mentioned that the average AGN consisted of 248 nodes whilst 74 for of those were primitive nodes. This means, on average, 174 nodes were added to the initial network and these must necessarily be either composite or clustered nodes. In actual fact the average distribution was 112 composite nodes compared to 62 clustered nodes. These results can therefore be summarised as:

Node Type	*Percentage Constitution*
Primitive	29%
Composite	45%
Clustered	26%

Figure 5-20 AGN Constituents

It must be remembered that the specific constitution of AGN network is heavily determined by the precise heuristics used in the algorithmic configuration. A different set of heuristics could give an entirely different shape, structure and size to the network. In this case the heuristics used were determined empirically, in that they were those that led to a better system performance (see Section 5.2.4.1.1 for precise configuration).

On the whole specific comment on the idiosyncratic content of a trained network has been avoided, instead a focus has been kept on the proven utilitarian benefit of the net in performing the task of automated assessment and the black box metrics that characterise its functional performance. In essence this is much the same approach that is taken by the experimenter who applies a neural net to a specified task and who cares little for the specific weight configurations that give rise to the best performance, but cares more for the training and net topology configurations that allow them to efficiently and repeatably reach the specific weight configurations that give rise to optimum performance. However, as the nodes created by learning algorithms do maintain a local and transparent mapping onto the strings from the problem domain, the process is perhaps not as "black-box" as the typical neural network-training algorithm. It is perhaps worth a surface inspection of the net creation process to reassure ourselves that the grown nets are indeed modelling sensible and coherent semantic properties. The nets themselves are however very large complex and heavily interconnected; it is difficult to establish an informed opinion from a surface inspection. It is easier to look at the program logs in order to get an insight as to what is going on. Taking the

networks trained on the first question, for purposes of illustration, and considering the clustering process first we note that the program logs[15] report:

```
Node [closer] is like [close] by=0.772348
Node [close] is like [closest] by=0.828198
```

These are obviously terms that are highly correlated semantically, being merely mild morphological variations on one another. Interestingly, within the handcrafted networks this same set can be found within the evidence list for the close node, the only difference being "closely" is a member of this set, yet does not seem to have been induced as a member of this set by the algorithm on pure statistical grounds. (In fact on closer inspection it is noted that "closely" only occurs once in the entire data set, hence would not survive the *alpha-cut* implemented within the clustering algorithm). Similarly:

```
Node [assembler] is like [assembly] by=0.686844
```

have been identified as similar.

The core phrase within the answer to question one, which constitutes an acceptable answer, is "machine-code", which can variously be phrased as "machine-language", "assembly-code" and "assembly-language". The handcrafted network models this potential interchange and combination of phrases explicitly. To parallel this we find that within the algorithmically grown network:

```
Node [assembler] is like [machine] by=0.739442
Node [assembly] is like [machine] by=0.768909
```

also:

```
Node [code] is like [language] by=0.711512
```

The other major cluster to be found in the handcrafted network is the "near or use" cluster. Within the statistically induced network the following are found which closely parallel this cluster:

---

[15] For purposes of clarity the following logs do not reflect the recursed clusters that are to be found in the nodes, but simply the core similarities between words that are identified from the corpus and upon which the recursed node hierarchies are based.

```
Node [close] is like [directly] by=0.726252
Node [close] is like [like] by=0.775058
Node [close] is like [uses] by=0.760639
Node [closest] is like [uses] by=0.685125
```

Not all of the derived associations are as semantically coherent. The following exemplify some of the more incoherent clusters found:

```
Node [assembler] is like [code] by=0.651757
Node [assembly] is like [close] by=0.841306
Node [assembly] is like [level] by=0.650669
Node [machine] is like [close] by=0.661293
Node [like] is like [code] by=0.691036
```

Some (but by no means all) of these incoherent clusters can be explained away by the following reasoning. When there is a compositioning heuristic defined which is anything greater than simple adjacency, and there do exist composites that by definition are close to one another, it stands to reason that the composite terms will share, or have similar, context histories. There are therefore instances where composites will be miss-classified as clusters.

This probably embodies a fundamental antagonism within the definitions of the heuristics, that is: the wider the window in the compositioning heuristic (which is necessary to capture composites that are more than simple adjacent words) the more imprecise the data is for inducing potential clustering. Empirically, the right-left compositioning heuristic of two seemed to optimally trade-off these antagonistic constraints.

Turning our attention now to the identified composites we find that at a simple lexical level:

```
composite= assembler && code
composite= assembler && machine
composite= assembly && code
composite= assembly && language
```

also

```
composite= machine && code
composite= machine && language
composite= machines && code
composite= machines && language
```

which again parallels the core "machine-code" phrase to be found in the hand crafted network.

Some further identified simple composites, which are worthy of note, are:

```
composite= instruction && set
composite= low && level
composite= operating && system
composite= processor && level
composite= reserved && words
```

More importantly however are the composites of clusters that were identified. Again the focus is on the core phrase "machine-code", and the critical composite being:

```
composite= (assembler ,assembly ,machine, machines) && (code,
language)
```

Note that although there is no precise[16] test for statistical significance of composites (or indeed clusters) in the outlined algorithms, it is worth noting that composites of clusters do have *better* statistical grounds for creation than their associated composites of simple lexemes. This is for the simple reason that the context histories for clusters are aggregates over the cluster's constituent parts. The statistical sample upon which the composite is based is therefore, simply, greater.

The above does not constitute a complete examination of all identified associations for the question under investigation, it is simply a subset intended to exemplify some of the recognisable, semantically coherent structures that are identified by the algorithms. As pointed out there are some identified structures, which although having a sound statistical basis for creation, as identified by the algorithms, on inspection do not appear to model anything of particular usefulness. This should be of no great concern. Firstly, the empirical utility of the grown structures has been proven which is justification enough. Secondly, the experimental design and especially the application of LSA are specifically intended to remove

---

[16] The crude approximation to a test for statistical significance is the implementation of the alpha-cut in the clustering and compositioning heuristics.

incoherent dimensions from the resulting multidimensional space. And where each node is represented by a dimension, this means the removal of incoherent nodes.

On surface inspection the nets trained on answers from the other questions exhibit similar qualities. That is sets of clusters can be found that seem to correspond to either morphological variants or synonymous groupings and sets of composites can be found which correspond to frequently used and/or idiomatic phrases. Generally, there are a further set of composites which composite the largest clustered groups. There are relatively few instances of clustered composites.

5.2.4.4. Conclusions

	False Positive	False Negative	Correct Estimator	Incorrect Estimator	Correlation
Simple LSA	0.24	0.29	0.77	0.55	0.36
Perceptually Augmented LSA	0.2	0.15	0.81	0.71	0.55

Figure 5-21 Results Summary

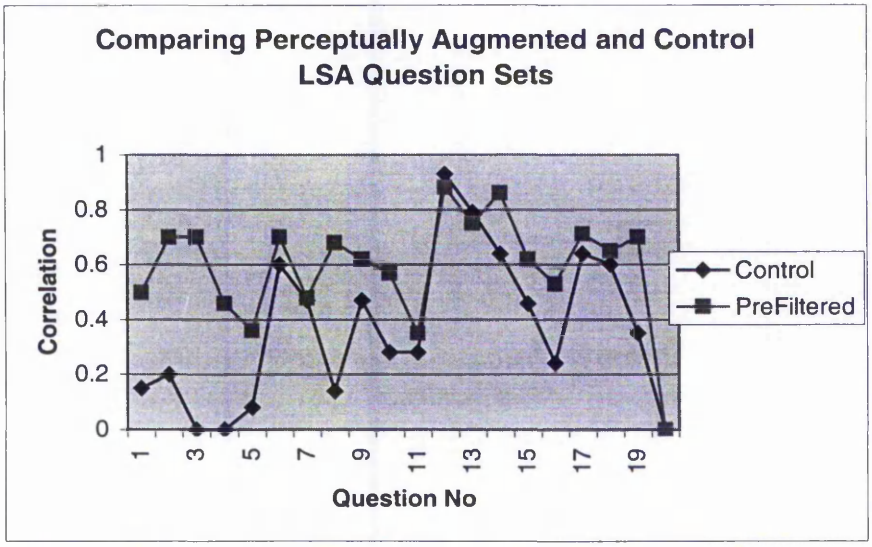


Figure 5-22 Graph to Compare Perceptually Augmented and Raw LSA Performance

The principle conclusion to be drawn is that the networks produced from the compositioning and clustering algorithms do seem to form an effective perceptual enhancer when considering an automated assessment application that utilises latent semantic analysis.

It was speculated that relatively low correlation measures of Experiment 3 was attributable to inherently lower dimensionality of the starting matrix (certainly as compared to essay evaluation applications). This is not a hypothesis that can be easily investigated directly, however the results from this experiment do seem to support it, for the document-term matrix that was generated for the perceptual network represents vectors of a significantly higher dimensionality. It is highly probable that a considerable number of these added dimensions proved useless, and consequently would have been discarded into the subspace that is unreachable by the orthogonal basis vectors that are produced after dimensional reduction. However the empirical evidence seems to suggest that a significant number of them were preserved within the basis vectors and that these dimensions served to aid the successful discrimination of correct and incorrect pseudo document vectors.

The hypothesis that activation passing networks can be "grown" by identifying statistical regularities within a problem domain, and that these networks can serve to perceptually augment incoming data providing higher-order derived descriptions of the same information, appears to be supported. The hypothesis that these higher-order descriptions can be utilised to "reasonable effect" within the application domain of automated assessment has been supported empirically by the above experiments.

These results do give rise to some further questions. The most important of which is: Given that LSA can be seen as a "perceptual re-alignment" of a particular feature space, what is it that the network growth algorithms are capable of identifying that the LSA process is not?

From the descriptions given of the data structures and algorithms in Chapter 3 the answers to this question may be quite obvious. However, this may be an appropriate point to re-visit some of these issues and to at least speculate what is going on behind the scenes.

First, let us examine the nature of LSA's perceptual realignment. Note initially that the simple document term matrix is insensitive to word order. It is however sensitive to the number of terms appearing in a document. After the application of SVD the input space has been realigned, to a set of ordered, maximally coherent basis vectors. These vectors are in

fact a form of clustering upon the initial input. After the dimensional reduction we have in a sense chosen to retain an arbitrary number of the best clusters.

Compare this to the perceptual refinement of the network growth algorithms. The first thing to note is that the whole notion of composition relates to word order. Whether it is windowed composition or simple adjacency, the fact that a composite node has been identified means that a contextual regularity has been identified and this context relates to the relative word order of its items. On the counter side, note that the network is relatively insensitive to number of terms per document[17], as it is a simple Boolean activation network. The term "relatively insensitive" is used because there is a small exclusion clause to this statement. Where a particular term occurs twice, and each time in a different context, and this context maps onto a specific composite node, then there is an implicit record of the term occurring twice.

The second thing to note is that the clustering that occurs within the network is significantly different to both the implicit clustering that happens after the dimensional reduction of an SVD produced matrix, and the recursive clustering that occurs within both the syntactic clustering applications discussed earlier in this chapter. Both of these methods of clustering are essentially best match first recursive procedures that reduce to an arbitrary final cluster size. The network clustering is different; clustering criteria is dynamic and local. Clustering does not occur on a best match first basis, there are absolute criteria to identify potential clusters. Also clustering does not reduce to an arbitrary cluster size, clustering simply stops when the local absolute criteria at each node are satisfied.

The third essential difference is the recursive complexity due to the two differing node constructs. We have seen that although LSA has an analogous process to clustering it has no analogy to the representation of composition. Considering, a cluster of composites would simply be impossible.

It is for these reasons that the perceptually augmented LSA process performs significantly better than the LSA process fed with simple words. Quite simply new, higher order, "useful" information is presented to the process, which the LSA process itself is incapable of deriving.

---

[17] Note this does not matter over much for single sentence answers as the only terms that are likely to be repeated are common words which are likely to be removed by the stop list anyway.

6. Conclusions

The subject matter for this thesis has been the development of a natural language processing framework for application in the field of automated assessment. This has been approached in the following chronological order (this is not necessarily the same as the order that it appears in this thesis):

1. Analyse the requirements of natural language processing in the context of automated assessment and develop a custom knowledge schema to specifically address these issues.
2. Formulate an evaluative strategy within which the performance of the activation passing networks can be empirically investigated
3. Perform two experiments. The first to demonstrate that it is possible to construct a network that is capable of effectively discriminating between correct and incorrect student responses. The second that demonstrates this networks ability to generalise to new unseen data.
4. Develop novel algorithms for growing "perceptual" networks consisting of both compositioning and clustering node-types.
5. Develop and evaluative strategy for the empirical investigation of these "perceptual" networks.
6. Perform a further two experiments the first of which uses an automated evaluative procedure to discriminate between correct and incorrect responses but uses as its input a raw-atomic description of the input feature space. The second of which uses a perceptually enhanced description of this same input.

It is interesting to note that for each of the two main stages (the development of the network schema and the development of the algorithms) as much investigative and development effort was required to define and implement the evaluative mechanisms as was required to do the primary implementation of the novel representational structure and algorithms respectively.

The main conclusions to be derived from each of these stages will be briefly summarised here (along with the references to the refereed publications in which this work has been published).

Firstly, the development of the custom knowledge representation network for application to automated assessment resulted in the definition of knowledge schema that strongly resembles a localist connectionist network, and consequently has combined representational and computational properties. (Allott et al 1994a)

An evaluative framework was defined, which drew heavily on the published academic works stressing the importance of detailed evaluation metrics for natural language processing systems. The aim was to produce a robust set of measures that not only gave a good global indicator of the system's performance, but whose fine grained metrics would point towards specific causes of problems within the system. (Allott et al 1994c)

Experiment 1 proved successful in that it was possible to produce a network within the custom knowledge schema that could effectively discriminate between correct and incorrect responses such that the results correlated 85% with tutor marked responses. Experiment 2, which was a measure of the systems ability to generalise to new unseen data, proved successful producing a tutor correlation of 65%. (Allott et al 1994b)

To address the principal problem of the human effort required to generate these networks, algorithms were developed which would, by identifying statistical regularities within the input data, generate complex networks consisting of composite and clustering nodes. These networks preserved the activation passing ability of the handcrafted networks and therefore have consequent computational properties in their own right. (Allott et al 1995, 1997a, 1997b, 1997c 1997d)

These networks are produced by unsupervised learning algorithms and therefore are not capable of discriminating between correct and incorrect responses on their own; they simply produce a higher order description of the input data. Consequently evaluation of these networks is a complex issue. To address this problem the technique of Latent Semantic Analysis was introduced. This is a mathematical technique that is capable of automating the correct/incorrect decision procedure with a little tutor input to calibrate the system. Further it can be fed with differing input data granularities.

Experiment 3 automated the decision procedure using a simple document term matrix as input data to the LSA process, where each term mapped to a simple word type. This met with a fair degree of success correlation (37%) with tutor marks. Although this is substantially

lower than the figures from Experiment 1 and Experiment 2 it is a technique that requires far less human input.

Experiment 4 used the same LSA process but started with differing input data. The input was a document term matrix of higher dimensionality where each term mapped onto the activation of a unique node on a network grown from the student data using the compositioning and clustering algorithms. This resulted in a tutor correlation of 55%, an improvement of 48% over the basic input matrix. This proves the effectiveness of perceptual networks in contributing significantly useful information to the automated assessment application.

6.1. Connectionism and Symbolism in Symbiosis

Both Hughes (1993) and Finches (1992) PhD theses concluded with some comments on the implication their work has for some of the biological theories of language development and processing. Some similar conclusions are possible within this thesis.

Specifically, in Allott et al (1997c) a high order analysis of the implementation of the compositioning algorithms was made, specifically those aspects that pertain to the processing of temporal data. The main point of this paper was to show that the algorithm and data structure combined could be analysed as collaborating symbolic and connectionist modules. Also when analysed as such there are certain aspects of its performance that correlate well with what is known of human cognitive behaviour. It is appropriate to repeat some of these comments here.

The architecture described in summary consists of a symbiotic connectionist and symbolic process. The connectionist process both provides a permanent store for the associations found between units and the perceptual framework for the overall process (i.e. identified units within data). The symbolic process (the saliency module) provides a type of working memory for our network. Let us consider each of these processes in greater detail.

A symbolic process by definition operates on symbols. The question of "what do these symbols represent?" is usually defined prior to the instigation of the process. However in the outlined architecture we circumvent the need to do this. We define only the lowest levels symbols - those at atomic level. The interaction between the connectionist and symbolic processes serves to identify new symbols that are hopefully more appropriate for the task in hand.

The connectionist layer represents the relationship between identified clusters/composites within the problem domain. This could well map out the hierarchical description of the problem domain, or by the use of excitatory and inhibitory links describe a causal link between nodes. However in the outlined design the exact nature of the relationship is embodied in the symbolic layer. By extracting this information from the network itself we allow for specialisation of the network.

There are certain aspects of the outlined model which correlate well with what has been observed of our own human problem solving behaviour. We comment on the similarities in idle speculation only and do not consider the similarities to constitute any form of proof of the validity of the proposed model.

6.1.1. Memory Types

Psychologists have for some time maintained distinctions between Short Term Memory (STM) and Long Term Memory (LTM) (Atkinson et al 1977). The three major distinctions being: duration, capacity and coding (described by Wicklegren 1991). Within the two modules discussed here there are similar distinction to be made. Items instantiated within the symbolic layer have a short life span (the length of the current input pattern) whilst the connectionist links are permanent. The coding of the nodes within the connectionist layer is contextual; all that is known about that node is inherent within the links attached to it. The tokens used within the symbolic layer are arbitrary representations. However, the capacity of the symbolic layer does not seem to be limited in the same way that STM seems to be. This could be due to their differing implementations (see 6.1.2.)

Note, also, in overall function the symbolic layer is similar to Klatzky's (1980) description of STM as a "mental workbench".

6.1.2. Symbolic Whilst Learning

It seems a feature of learning that, when presented with learning a new task or skill, the processing tends to be symbolic and procedural in the initial phases, less so in the latter phases. For example, consider learning to type, to drive a car or learning to read. Again we see similarities with the symbiotic design. Consider the clustering problem discussed above. In the initial phases the identification of new features is performed entirely within the symbolic layer. Once identified the apparatus necessary to *perceive* the feature is

incorporated within the nodes and links of the connectionist layer, so this is where the processing now takes place.

6.1.3. Speed and Implementation

The architectural implementation of the modern computer is in most cases a serial, symbolic, Von Neumann process. Whereas there is no doubt that the brain is made up from nerves, which can operate in parallel.

People seem to have two modes of operation, to quote Norman (1986) "one rapid, efficient, subconscious, the other slow, serial and conscious". The proposed architecture also seems to perform in two modes: a fast serial symbolic process, a slow connectionist process. A distinction between processes is preserved, although the performance ratios disagree with one another. It seems possible, however, that the juxtaposition of performance ratios is attributable to the differing implementations. Certainly the connectionist network discussed above is in reality a serial emulation of a connectionist process and so it would be reasonable to expect a performance drop.

The issue of emulating a serial process within a connectionist architecture is more complex. Clark (1987) has speculated for some time that "the human mind might effectively simulate a serial, symbol processing Von Neumann architecture." And it is interesting to note that in his book (Clark 1990) he notes two shortfalls of connectionist networks in explaining human capacity:

1) To be able to perform serial reasoning in which the ordering of operations is vital.
2) To be able to utilise a control structure in order to specify salient micro features for inductive generalisation.

This is interesting as this is precisely the type of functionality being satisfied here by the symbolic layer.

Rumelhart and Smolensky (1986) have also pondered on the human capacity to engage in conscious, symbolic reasoning, and Touretsky (1990) has contributed to the debate with his proof that neural nets can be used as Turing Machines. All we can do here is to leave the open ended question: would a neural implementation of a symbolic process have a limited

capacity and be relatively slow to process? If so, this would fall in line with the arguments presented above.

6.2. Further Work

The number of ways that the current work could be extended is varied. Some of the best contenders are considered below.

6.2.1. Improvements to the LSA procedure

There are a number of ways that improvements to the LSA procedure may be explored, especially in relation in to how the LSA procedure interfaces with the clustering and compositioning algorithms.

6.2.1.1. Neural Net Decision Procedure

Latent semantic analysis maps both terms and documents into a N-dimensional space. Within this spatial interpretation the correctness judgment is usually implemented as a simple linear algebra operation (usually dot product, sometimes simple Euclidean distance). This assumption of linearity may be incorrect. It is possible that a better estimate of a sentence's correctness would be the definition of some non-linear subspace of the N-dimensional space in which all documents and terms are represented. A neural network is a way of inducing the optimum non-linear subspace that best distinguished between correct and incorrect responses. The procedure would be quite simple:

1. Construct a 3-layer net with N input nodes, arbitrary hidden nodes and single output node.
2. With marked responses train the net with the documents input vectors (where size of the input vector determines N) and output vector trained to {0,1} - the correctness judgement.
3. Reconfigure net with different number hidden nodes until adequate trade-off is reached between over-fitting and accuracy.
4. Now compare the neural net judgement to that of the simple dot product measure.

This technique is not practical for applying to real data as it would require too much human input (i.e. all documents pre-marked) but it would be an interesting experiment to test the linearity assumption.

6.2.1.2. Term Weighting

As was mentioned in Section 5.2.4.3 there is no precise test for statistical significance in the network growth algorithms, but more importantly no marking-up or strengthening of nodes or links to reflect the importance of a node in terms of the frequency that it appears or the weight of the corroborating evidence that supports a nodes instantiation. If this were implemented we would have the ideal grounds upon which we could implement term weighting on the document/term matrix that is presented to the LSA procedure. Term weighting has been shown by Landauer et al (1997b) to significantly improve the performance of the LSA procedure.

6.2.1.3. Dynamic Dimensional Reduction

Within the LSA process, the decision of the number of dimensions to reduce the document term/matrix to is a fairly ad hoc procedure. To illustrate just how ad hoc this procedure is, consider the following:

Imagine that a handcrafted network is used as the perceptual augmentation process on the LSA procedure for a simple question. It should be fairly apparent (and indeed Experiment 1 and 2 prove) that one of these nodes (i.e. input terms on the document term matrix) would be statistically significant in predicting a specific answers "correctness". This will simply be the node that is intended to model the answers "truth" value. However when the LSA procedure is applied this document term matrix will be reduced to an arbitrary number of dimensions, of which this "effective" dimension will just be a part. The LSA implemented correctness judgement will simply be a similarity measure of vectors within this dimensional space. The noise to signal ratio is high: as has been shown there is a single dimension which is effective at predicting "correctness", but this could be one of twenty un-weighted dimensions. A sentence could be judged incorrect even if the effective dimension is in alignment, but two other arbitrary dimensions are giving contrary indications.

A possible way to avoid this problem is to implement a procedure that is capable of estimating a dimensions contribution to or ability to estimate the sentences truth state. This procedure would necessarily be a supervised procedure requiring knowledge of the mark that an answer has been given. But if implemented this information could be fed back in order to dynamically and knowledgably reduce the matrix to an optimally effective set of dimensions. (Note the neural net decision procedure from Section 6.2.1.1 is a procedure capable of making such judgements.)

6.2.2. *Clustering/Compositioning Algorithm Enhancements*

There is virtually no end to the ways in which the core algorithms can be enhanced. Not only is it possible to make adjustments and optimisations to the algorithms themselves, but the various configurable heuristics can be tuned in various dimensions. Within this thesis a simple consistent line of argument has been taken to prove that the "grown" networks are effective within this problem domain, however there is plenty of scope for further investigation in identifying the precise configurations that lead to the most effective networks.

6.2.3. *Network Visualisation Tools*

Unlike the networks hand generated by the tutor the networks generated by the algorithms are extremely dense and complicated. Specifically, rather than a single rooted hierarchy; most of the algorithmically produced networks are many rooted and somewhat tangled in nature. Further, the networks do not necessarily form bottom heavy hierarchies, for the simple reason that the number of parents of a single child node is in no way restricted. It is quite possible for the sum of the parents of a discrete subset of child nodes to outnumber the number of child nodes themselves. In fact rather conforming to the restrictive tree type hierarchy, the AGN's conform to the more general description of a directed a-cyclic graph, and conceivably even to a directed cyclic graph. Consequently, in their raw form they are virtually impossible to present in 2D. The obvious constraint that links do not cross on the 2D representation in order for the graph to be readable is impossible to satisfy. However, if some useful visualisation tools could be produced, or maybe simple structured queries devised that would expose the network in fragments, they may deliver some useful insights that would aid in the refinement of the algorithms.

6.2.4. *Non Discretisation of the Activation Network*

An obvious enhancement would be to relax the constraint that all activations be Boolean. This would make the algorithms applicable to a much wider range of problem domains, though possibly not bring that many benefits to this particular automated assessment application. The algorithms would need redesigning if this were done, primarily it is the issue of composition and the identification of true instances of the occurrence of a composite that would prove problematic. Associated with this point and Section 6.2.1.2 above, is the ability to mark-up or strengthen the composite and clustered nodes that have a stronger statistical grounds for creation.

6.2.5. Parallel Implementation

A major limitation of the algorithms as they stand is that the algorithms are inherently parallel in nature. Implementing them on a serial machine obviously limits the dimensionality of the problem domain due to the consequent combinatorial explosion. Implementing the algorithms on a truly parallel machine would open up interesting new possibilities in applying the algorithms to new applications.

6.2.6. Application of Algorithms to Different Domains.

Probably the most interesting piece of research is identifying the applicability of the core algorithms to different domains. Obvious contenders within the natural language processing field are grapheme analysis, phoneme analysis and syntactic analysis. Each of course would need the development of an evaluative framework or application that would be capable of accepting input at the perceptually enhanced description. Another potential application is within the generic domain of data mining. In fact some progress has already been made in investigating this route (Allott et al 1997d).

6.3. Concluding Remarks

Automated Assessment is an attractive application to study both because of its obvious benefits in the real world and as a formal framework in which to analyse the performance of natural language understanding systems.

However, despite the limited successes that have been reported both here and in the latent semantic analysis work, we should be sceptical of any of the approaches in their present form being applicable on large scale in the field. Consider for a minute the cynical teacher who marks his/her student reports by first looking at the thickness of the report and then at the neatness of the handwriting. The teacher is by no means analysing the content, but they are making judgements on indicators, which in their experience correlate sufficiently well with content. This is quite possibly what is being uncovered by techniques such as LSA, i.e. it is identifying and making judgement on features which happened to correlate with the quality of the work. This is not a criticism of the approach as this form of empirical, Turing Test type approach is precisely what is needed when dealing with such difficult to define problems as understanding. Nor does it invalidate the results, as the metrics outlined will identify any major statistical irregularity in a given student result base. No, the problem is entirely practical. Given any marking scheme, if the mechanics of the operation are public knowledge, is it possible to satisfy the mechanics of the evaluative mechanism without satisfying the "proper" criteria for correctness? Unfortunately this is probably the case. Take LSA: a target

document is defined as a vector in N-dimensional space. If the vectors for individual terms are known it would be fairly easy to approximate to the target vector with just a few well chosen terms. Although improbable, the resourcefulness of students, especially when it comes to finding ways of avoiding doing work, should not be underestimated. Note the critical distinction here is: although the error metrics developed will check statistically for refutations of the given model of correctness within a particular student base, in order to be genuinely applicable in the real world the model really needs to be able to stand up to deliberate attempts to produce counter-examples.

Despite this, automated assessment remains a useful framework within which natural language theories may be developed and tested. It is within this framework that two pieces of work have been completed. Firstly the development of a network schema with combined representational and computational properties. Secondly, the development of algorithms for producing such networks. Both have met with limited success within this domain. What remains to be seen is whether the core technologies may have any practical application within any other domain. This is an avenue of investigation (Allot et al 1997d) that has already started by analysing the algorithms within the more generic application of data mining.

7. References

Abdullah, M.Z. & Foxley, E. (1991). *Automatic Program Assessment System.* Proceedings of the IFIP Conference on Software Quality.

Allen, J. (1995), *Natural Language Understanding*, Benjamin Cummings, California.

Allott N, Halstead P, Fazackerley P (1995), *'A Clustering Algorithm to Produce Context Rich Networks'* Neural Networks and their Applications, ed Taylor J.G., p265-269., John Wiley & Sons, Chichester.

Allott N, Halstead P, Fazackerley P (1994a), *"Knowledge for Language"*, Postgraduate Workshop AISB94, Leeds.

Allott N, Halstead P, Fazackerley P (1994b), *"A Knowledge Driven Aid to the Automated Assessment of Free Text"*, AISB Quarterly Journal, Vol 88, p19-24

Allott N, Halstead P, Fazackerley P (1994c), *"Automated Assessment: Evaluating a Knowledge Architecture for Natural Language Processing"* Applications and Innovation in Expert Systems II, eds Milne, R. and Montgomery, A., p319-332..

Allott N, Halstead P, Fazackerley P (1997a), *"Connectionist Pattern Matching for Information Extraction",* Natural Language Processing for Business Use, AI and Soft Computing Series.

Allott N, Halstead P, Fazackerley P (1997b), *"Sequence Clustering Using Time Delay Networks"* in Artificial Neural Nets and Genetic Algorithms, Eds. Smith, G. et al Springer, New York, p454-459.

Allott N, Halstead P, Fazackerley P (1997c), *"Connectionism and Symbolism in Symbiosis"*, in Artificial Neural Nets and Genetic Algorithms, Eds. Smith, G. et al Springer, New York, p570 –576.

Allott N, Halstead P, Fazackerley P (1997d), *"Composition, Clustering and Predictor Pruning in Hierarchical Networks"* Neural Networks for Data Mining, AI and Soft Computing Series, p181-207.

ALPAC (1966). *Languages and Machines: Computer in Translation and Linguistics.* Report of the Automatic Language Processing Advisory Committee. Division of Behavioural Sciences. National Academy of Sciences. National Research Council Publication. Washington DC.

Anderson J. R. (1983) *The Architecture of Cognition*, Cambridge: Harvard University Press.

Atkinson, R.C., and Shiffrin, (1977) R.M. *'Human Memory: A Proposed System and its Control Processes'* In Human Memory: Basic Processes, Bower, G.H.(ed) New York : Acedemic Press.

Bahl, L.R., Jelinek, F. & Mercer, R.L. (1983). *A Maximum Likelihood Approach to Speech Recognition.* IEEE Transactions of Pattern Analysis and Machine Intelligence, PAMI-5, p179-190.

Balkan, L. et al (1994). *Test Suites for Natural Language Processing.* ELSNET pp17-24, Edingburgh.

Bartlett, F.C. (1932). *Remembering*, Cambridge University Press: Cambridge.

Benford, S., Burke, E. & Foxley, E. (1992). *Course Developers Guide to CEILIDH System.* LTR Report. Department of Computer Science, Nottingham University.

Bishop, C.M. (1995), *Neural Networks for Pattern Recognition*, Oxford: Oxford University Press.

Boisen, S. & Bates, M. (1992) *A Practical Methodology for the Evaluation of Spoken Language Systems.* In Proceedings of the Third Conference on Applied Natural Language Processing. p162-169. Trento, Italy.

Bolinger, D. (1965). *The Atomisation of Meaning*, Language, 44, p555-573.

Boole, G. (1854), *An investigation of the Laws of Thought*, Dover, New York.

Brachman, R.J. (1983). *What IS-A Is and Isn't: An Analysis of Taxonomic Links in Semantic Networks.* IEEE Computer 16(10) p30-36.

Brachman, R.J. (1985). *I Lied About the Trees. Or Defaults and Definitions in Knowledge Representation.* AI Magazine 6(3) p60-93.

Brown P, Lee H and Spohrer, (1983) *'Bayesian Adaptation in Speech Recognition'*, Proceeding of the ICCASP, Boston, 761-764.

Brugman, C. & Lakoff, G. (1988). *Cognitive Topology and Lexical Networks. In: Lexical Ambiguity Resolution*: Perspectives from Psycholinguistics, Neuropsychology and Artificial Intelligence, ed Cottrell, G.W. et al. San Mateo, CA: Morgan Kaufman Publishers.

Brugman, C. (1981*). The Story of Over: Polysemy, Semantics and the Structure of the Lexicon.* New York: Garland Press.

Bryson, A. E. and Ho, Y. (1975). *Applied Optimal Control.* John Wiley, N.Y.

Carbonell, J.G. (1970) *AI in CAI: An Artificial Intelligence Approach to Computer Aided Instruction.* IEEE Transactions on Man-Machine Systems 11(4) p190-202.

Carbonell,J.G., Hayes,P.J., (1987) *Natural Language Understanding*, Encyclopaedia of Artificial Intelligence, ed. Shapiro.

Chase C. I. (1978*) Measurement for Educational Evaluation*, Addison Wessley.

Church K. W. (1988) *Stochastic parts program and noun phrase parser for unrestricted text*, Proc. 2nd Conf. on Applied Natural Language Processing, 136 – 143.

Clark, A. (1990) *'Microcognition'*. MIT Press, Cambridge, 1990.

Clark, A. (1987) *'Connectionism and Cognitive Science.'* In J. Hallam and C. Mellish eds., Advances in Artificial Intelligence, pp3-15, Chichester: Wiley, 1987

Collier R (1994). *'An Historical Overview of Natural Language Processing Systems that Learn'*. Artificial Intelligence Review, Vol 8, p17-54

Collins A.M. and Loftus E.F. (1975) *"A Spreading Activation Theory of Semantic Processing"*, Psychological Review, 85, 407-28.

Collins Concise Dictionary Plus (1989). Collins: London, England.

Computer Science Department (1989). Model Answers for 1989 Examinations. The Nottingham Trent University.

Dahl, V (1995), *Understanding and Translating Language - Challenges of the 90's*, AICOM, p71-77, Vol 8, No 2.

DARPA proceedings (1993) Speech and Natural Language Workshop.

Du M W, Chang S C (1994), *'An Approach to Designing Very Fast Approximate String Matching Algorithms'* IEEE Transactions on Knowledge and Data Engineering, Vol 6, No 4, 620-633.

Dumais, S. T. (1991), "*Improving the retrieval of information from external sources.*" Behavior Research Methods, Instruments and Computers, 23(2), 229-236.

EAGLES (1994) Evaluation Working Group. Interim Report. Obtainable from the centre for Language Technology, University of Copenhagen.

Earley, J. (1970). "An Efficient Context-free Parsing Algorithm". Communications of the ACM 12, 2:94-102.

Elisseeff, A., and Paugam-Moisy, H. (1997), "*Size of multilayer networks for exact learning: analytic approach,*" in Mozer, M.C., Jordan, M.I., and Petsche, T., (eds.) Advances in Neural Information Processing Systems 9, Cambrideg, MA: The MIT Press, pp.162-168.

Elman, J.L. (1990), Finding Structure in Time, Cognitive Science, 14, p 179-211.

Erman, D.L., et al. "*The HEARSAY-II Speech Understanding System: Integrating Knowledge to Resolve Uncertainty,*" Blackboard Systems, 31--86 (1988).

Filmore, C.J., Kay, P. & Connor, M.C. (1987), *Regularity and Idiomacity in Grammar:* The Case of Let Alone, Internal Report, Department of Linguistics, University of California, Berkley.

Finch, S.P. (1993) *Finding Structure in Language*. PhD Thesis, Edinburgh University.

Finch, S.P. & Chater, N. (1992a). *Bootstrapping Syntactic Categories using Statistical Methods*. In W. Daelemands & d. Powers (Eds) Background and Experiments in Machine Learning of Natural Language. ITK Tilburg, NLK 229-236.

Finch, S.P. & Chater, N. (1992b) *Bootstrapping Syntactic Categories*. Proceedings of the Fouteenth Annual Conference of the Cognitive Science Society of America. Bloomington, Indiana. 820-825.

Finch, S.P. & Chater, N. (1992c) *Unsupervised Methods for Finding Linguistic Categories.* Proceedings of the International Joint Conference on Neural Networks, Brighton, UK.

Flanagan, K. (1994). *Error Classification in MT Evaluation.* In Technology Partnerships for Crossing the Language Barrier: Proceedings of the First Conference of the Association for Machine Translation in the Americas. Columbia: MD.

Foxley, E. & Gwei, G.M. (1989). *Synonymy and the Contextual Disambiguation of Words* 2(2) p111-134

Foltz, P. W., Kintsch, W.,& Landauer, T. K. (1998). *The measurement of textual Coherence with Latent Semantic Analysis.* Discourse Processes, 25, 285-307.

Frege, G. (1879) *Begriffsschrift*, translated in *From Frege to Godel: A Source Book in Mathematical Logic*, Harvard University Press, Cambridge, 1967 ed. van Heijenoort (1967) 1-82.

Garagliano, R. (1996) [Online] http://www.dur.ac.uk/~dcs0www3/lnle/system.html

Gasser, M., Lee, C. (1992), *Networks that Learn about Phonological Feature Persistence*, in Connectionist Natural Language Processing, ed Sharkey, N., p349-362. Intellect Books, Oxford.

Godel, K. (1967). *On Formally Undecidable Propositions.* Basic Books: New York.

Gronlund N. E. (1981) *Measurement and Evaluation in Teaching*, NewYork: Macmillan.

Gwei, G.M. & Foxley, E. (1987). *A Flexible Synonym Interface with Application examples in CAL and Help Environments.* The Computer Journal, 6(30), p551-557

Gwei, G.M. (1987). *New Models of Natural Language for Consultative Computing.* PhD Thesis, Department of Computer Science, Nottingham University.

Harman, D. (1995) *The First Text Retrival Conference* (TREC1). National Institute of Standards and Technology Special Publication 500-207. Gaithburg: MD.

Harris, C. (1992), *Connectionism and Cognitive Linguistics*, in Connectionist Natural Language Processing, ed Sharkey, N., p349-362. Intellect Books, Oxford.

Hayes, P.J. (1979). *The Logic of Frames.* In Frame Conceptions and Text Understanding. ed Metzing, D. Walter de Gruyter and Co: Berlin.

Hebb, D. (1949) *The Organisation of Behaviour*. Wiley and Sons: New York.

Heckerman, D. & Horvitz E. (1995) *Inferring Informational Goals from Free-Text Queries.* Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence. 230-238, Morgan Kaufman, San Fransisco.

Hinton, G. (1984). *Parallel Computations for a Controlling Arm*. Journal of Motor Behaviour, 16, 171-194.

Hofstadter, D. (1979*). Godel, Escher, Bach: An Eternal Golden Braid.* Penguin: London, England.

Hornik, K., Stinchcombe, M. and White, H. (1989), "*Multilayer feedforward networks are universal approximators,*" Neural Networks, 2, 359-366.

Hornik, K. (1993), "*Some new results on neural network approximation,*" Neural Networks, 6, 1069-1072.

Hughes, G.E. & Cresswell, M.J. (1968). An Introduction to Modal Logic. Methuen: London.

Hughes, J. (1992*) Automatically Acquiring a Classification of Words*. PhD Thesis, Leeds University.

Hughes, J. (1992). *Automatic Word Classification*. In Proceedings of the 19th International Conference for Literary and Linguistic Computing. ChristChurch Oxford, April 1992.

Hughes, J., Atwell, E. (1993*). Acquiring and Evaluating a Classification of Words.* Proceedings of IEE Grammatical Inference Colloquium. University of Esses. Colchester. 1993.

Hughes, J. & Atwell, E. (1994) *A Methodological Approach to Word Class Formation using Automatic Evalutation*. Proceedings AISB 1994.

ISO/IEC 9126 (1985*). Information Technology* - Software Product Evaluation - Quality Characteristics and Guidelines for their Use. December.

Jackendoff, R. (1983). *Semantics and Cognition*. Cambridge, MA: MIT Press.

Jackson, E., Appelt, D., Bear, J., Moore, R., and Podlozny, A. (1991), *A Template Matcher for Robust NL Interpretation,* Speech and Natural Language Workshop, DARPA.

Jelinek F, Mercer R and Bahl (1983), *'Continuous Speech Recognition: Statistical Methods'*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol PAM-5.

Johnson_Laird, P.N., Herrmann, D.J., and Chaffin, R. (1984), *"Only Connections: A Critique of Semantic Networks"*, Psychological Bulletin vol 96, p292-315.

Jordan, M.I. (1986), *Attractor Dynamics and Parallelism in a Connectionist Sequential Machine*, Proceedings of the Eighth Conference of the Cognitive Science Society, p531-546.

Kant, I. (1963, originally published 1787). *Critique of Pure Reason*, Smith, N.K (translation). Macmillan: London.

Katz, J.J & Fodor, J.A. (1963*), The Structure of semantic Theory, Language,* vol 56, p 251-299.

Keenan F (1992), *'Large Vocabulary Syntactic Analysis for Text Recognition'*, PhD Thesis, The Nottingham Trent University.

King, M. (1996*). Evaluating Natural Language Processing Systems.* Communications of the ACM. Vol 36, No1.

Klatzky, R.L. (1980) *'Human Memory: Structure and Processes*. Freeman, San Francisco.

Kluender, R. (1989), *Semantic Barriers to Extraction*, Conference on the Psycho-linguistics Island Constraints, university of Ottawa.

Kohonen T., Hinton G., Taylor J., Baum E. et al (1997) *Panel Discussion at the ICNN97 on Connectionist Learning.* OnLine: http://cns-web.bu.edu/inns/index.htm

Kosko, B. (1990), *Fuzziness vs. Probability*. International Journal of Gen. Systems 17(2-3), 211-240.

Kukich K (1992), *'Techniques for Automatically Correcting Words in Text'*, ACM Computing Surveys, Vol 24, No 4, 377-439.

Kuno, S. (1987), *Functional Syntax: Anaphora, Discourse and Emapthy*. Chicago: University of Chicago Press.

Landauer, T. K., & Dumais, S. T. (1997a*). A solution to Plato's problem*: The Latent Semantic Analysis theory of the acquisition, induction, and representation of knowledge. Psychological Review, 104, 211-240.

Landauer, T. K., Laham, D., Rehder, B., & Schreiner, M. E., (1997b). *How well can passage meaning be derived without using word order? A comparison of Latent Semantic Analysis and humans*. In M. G. Shafto & P. Langley (Eds.), Proceedings of the 19th annual meeting of the Cognitive Science Society (pp. 412-417). Mawhwah, NJ: Erlbaum.

Landauer, T. K., Foltz, P. W., & Laham, D. (1998). *Introduction to Latent Semantic Analysis.* Discourse Processes, 25, 259-284.

Landauer, T. K., Laham, D., & Foltz, P. W., (1998*). Learning human-like knowledge by Singular Value Decomposition: A progress report.* In M. I. Jordan, M. J. Kearns & S. A. Solla (Eds.), Advances in Neural Information Processing Systems 10,(pp. 45-51). Cambridge: MIT Press.

Lehmann, S. (1993). *Test Suites for Natural Language Processing*. Internal Report Linguistic Research Engineering (LRE), European Commision.

Lewis, C.L. (1912), *Implication and the Algebra of Logic*, reprinted in J.D. Goheen & J.L. Motherhead eds, Collected Papers of Clarence Irving Lewis, Stanford Universtiy Press, Stanford.

Lou B, Foxley E (1994a), *'A Simple Text Automatic Marking System'*. The AISB Conference of Computational Linguistics for Speech and Handwriting Recognition. Leeds.

Lou, B, Foxley, E (1993), *Semantic Understanding and Fuzzy Techniques,* Technical Report of the Computer Science Department. Nottingham University.

Lou, B, Foxley, E (1994b) *A Multi Subject Intelligent Student Assessment System*. Technical Report of the Computer Science Department. Nottingham University.

Lou, B, Foxley, E (1995) *Semantic Understanding for an Automatic Marking System Using Fuzzy Techniques*. The 5th Scandanavian Conference on AI (SCAI95) p436-440. IOS Press.

Lucas, S.M. & Dampner, R.I. (1992). *Syntactic Neural Networks*. in Connectionist Natural Language Processing, ed Sharkey, N., p56-82. Intellect Books, Oxford

Marshall S. M.. (1986) "*An Intelligent Marking Assistant: An Application of Artificial Intelligence in Teaching*", Higher Education Research and Development, p201-211.

Marzal A, Vidal E, (1995) '*Computation of Normalised Edit Distances*', IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol 15, No 9, 926-931.

McCarthy, J. (1980). *Circumscription - A Form of Non Monotonic Reasoning*. Artificial Intelligence 13(1,2) p27-39.

McClelland, J.L & Rumelhart, D.E. (1986) *The Appeal of in Parallel Distributed Processing*, in Parallel Distributed Processing: Explorations in the Microstructure of Cognition, ed Rumelhart& McClelland, Vol 1, Cambridge, MA:MIT Press.

McClelland, J.L. & Rumelhart, D.E. (1981). *An Interactive Activation Model of Context Effects in Letter Perception*: Part I. An account of basic findings. Psychological Review, 88, 375-407.

McClelland, J.L. (1981). *Retrieving General and Specific Information form Stored Knowledge of Specifics*. Proceedings of the Third Annual Conference of the Cognitive Science Society, Hillsdale, NJ: Lawrence Erlbaum.

McCulloch, W. & Pitts, W. (1943) *A Logical Calculus of the Ideas Immanent in Nervous Activity*. Bulletin of Mathematical BioPhysics.

McDermott, D. & Doyle, J. (1980). *Non-Monotonic Logic* I. Artificial Intelligence 13(1,2), p41-72.

McDermott, D. (1987). *A Critique of Pure Reason*. Journal of Computational Intelligence.

Miller, G.A. (1985*). WordNet: A Dictionary Browser*. in Information in Data, Proceedings of the First Conference of the UW Centre for the New Oxford Dictionary. University of Waterloo: Waterloo, Canada.

Miller G.A & Teibel A.T. (1991), *A Proposal for Lexical Disambiguation*, Proceedings of the Speech and Natural Language Processing Workshop, DARPA.

Miller, G.A. & Johnson-Laird, P.N. (1976) *Language and Perception*, MA: Harvard University Press.

Miller, G.A. (1978) *Semantic Relations Among Words*, in Linguistic Theory and Psychological Reality, ed Halle, M. et al, Cambridge MA: MIT Press.

Minsky M. (1975) *A Framework for Representing Knowledge in The Psychology of Computer Vision*, editor Winson P., NewYork: McGraw Hill.

Minsky, M. & Papert, S. (1969) *Perceptrons*. MIT Press: Cambridge.

Moore R.C. & Dowding J. (1991), *Efficient Bottom Up Parsing*, Proceedings of the Speech and Natural Language Processing Workshop, DARPA.

Moore, R.C. (1985). *A Formal Theory of Knowledge and Action.* in Formal Theories of the Common Sense World, Ablex Publishing.

MUC-3. (1991). *Proceedings of the third Message Understanding Conference*. Morgan Kaufman: San Mateo.

Nelson C.H., (1970) *Measurement and Evaluation in the Classroom*, New York: Macmillan.

Norman, D. (1986) *'Reflections on Cognition and Parallel Distributed Processing'* In McClelland and Rumelhart (ed) Parallel Distributed Processing, vol2, p110-146. MIT Press, Cambridge.

Pavelin, C. (1988). *Logic in Knowledge Representation.* in Approaches to Knowledge Representation, ed Ringland, G.A, Duce, D.A. John Wiley: New York.

Pinker, S. (1994), *The Language Instinct*, Penguin, London, England,.

Press, W.H. Teukolsky, S.A., Flannery, B.P. & Vetterling, W.T. (1996), *Numerical Recipes in C*, The Art of Scientific Computing, Cambridge University Press, Cambridge.

Quillian M.R. (1966), "*Semantic Memory*" unpublished PhD dissertation, Carnegie Institute of Technology:Pitsburg.

Quillian R.(1968), "*Semantic Memory*" in Semantic Information Processing, editor Minsky M., Cambridge: MIT,.

Randal, D. (1988) *Semantic Networks, in Approaches to Knowledge Representation*, ed Ringland, G.A, Duce, D.A. John Wiley: New York.

Reiter, R. (1985). *On Reasoning by default. in Readings in Knowledge Representation*, ed Brachman, R.J. & Levesque, H.J.. Morgan Kaufman: Los Altos, CA.

Ringland, G. (1988) *Structured Object Representation: Schemata and Frames.* in Approaches to Knowledge Representation, ed Ringland, G.A, Duce, D.A. John Wiley: New York.

Riseman E and Hanson A (1974), *'A Contextual Postprocessing System for Error Correction Using Binary N-Grams'*, IEEE Transactions on Computers, Vol C-23, 490-493.

Rosenblatt, F. (1962) *Principles of Neurodynamics*. Spartan Books: New York.

Roget, P.M. (1987) Rogets Thesaurus, Longman, Reading, UK. First Edition 1852.

Rose, T.G. (1993). *Large Vocabulary Semantic Analysis for Text Recognition.* Unpublished PhD Thesis. Department of Computer Science, The Nottingham Trent University.

Rumelhart D.E. and McClelland J. (1986) *Parallel Distributed Processing,* Cambridge: MIT.

Rumelhart, D.E., Hinton G.E., & Williams, R.J. (1986) *Learning Internal Representations by Error Propagation*, in Parallel Distributed Processing: Explorations in the Microstructure of Cognition, ed Rumelhart& McClelland, Vol 1, p319-362, Cambridge, MA:MIT Press.

Rumelhart, D., Smolensky, P. McClelland, J & Hinton, G. (1986) *' Schemata and Sequential Thought Processes in PDP Models'* Parallel Distributed Processing, vol2, p110-146. MIT Press, Cambridge.

Russell, B. (1945*). A History of Western Philosophy*. Simon and Schuster: New York.

Sampson G. (1987) *Evidence against the "Grammatical"/"Ungrammatical" distinction*, in Corpus Linguistics and Beyond: Proceedings of the 7th International Conference on English Language Research on Computerised Corpora, Ed W. Meijs, Amsterdam, Rodopi, 219-226

Schank R.C. and Colby K.M. (1973) *Computer Models of Thought and Language*, San Francisco: Freeman, 1973.

Schank, R., (1975). *Conceptual Information Processing*. Amsterdam: North Holland.

Schank, R.C. & Abelson, R.P. (1977). *Scripts, Plans, Goals and Understanding*. Lawrence Erlbaum Associates: NewYork.

Schank, R.C. and Colby, K.M. (1973) *Computer Models of Thought and Language*. San Fransisco: Freeman.

Schubert, L. (1991) *"Semantic Nets are in the Eye of the Beholder", Principles of Semantic Networks,* (editor J.F.Sowa), Morgan Kaufman, San Mateo, CA.

Schubert, L.K. (1976). *Extending the Expressive Power of Semantic Networks*. Artificial Intelligence 7(2). p163-198.

Sejnowski, T.J. & Rosenberg, C.R. (1987), *Parallel Networks that Learn to Pronounce English Text*, Complex Systems, 1, p145-16,

Servan-Schreiber, D., Cleeremans, A. & McClelland, J.L. (1989) *Learning Sequential Structure in Simple Recurrent Networks*, In Advances in Neural Information Processing Systems 1, ed Touretzky, D.S, p643-642.

Shannon, C.E., (1951) *Prediction and Entropy in Printed English*, Bell System Technical Journal, vol.30, p50-64.

Shapiro, S.C. (1971). *A Net Structure for Semantic Information Storage, Deduction and Retrieval*. In Proceedings of the Second IJCAI, p512-523. Morgan Kaufman: Los Altos, CA.

Sokal, R.R. and Sneath, P.H.A. (1963*). Principles of Numerical Taxonomy*. Freeman, San Francisco.

Sowa, J.F., (1984), *Conceptual Structures: Information Processing in Mind and Machine*, Addison-Wessley, Massachusetts.

Sowa, J.F. (1991), *Principles of Semantic Networks*. San Mateo, CA: Morgan Kaufman.

Sowa, J.F. and Way E.C. (1986) *Implementing a semantic interpreter using conceptual graphs*. IBM Journal of Research and Development, 30(1) p57-69.

Sowa, J.F. & Mineau, G.W. & Moulin, B. (1993) *Conceptual Graphs for Knowledge Representation*, Springer Verlag.

Sparck-Jones, K. & Galliers, J.R. (1993). *Evaluating Natural Language Processing Systems.* Technical Report 291, Computer Laboratory, Cambridge University.

Sparck-Jones, K. (1994). *Towards Better NLP Evaluation. Proceedings of the Human Language Technology Workshop.* ARPA, Morgan Kaufman: San Fransisco, p102-107.

Touretsky, D.S. (1990) *'BoltzCONS: Dynamic symbol Structures in a Connectionist Network'* Artificial Intelligence vol46, p5-46.

Turing, A. (1964). *Computing Machinery and Intelligence.* Reprinted in Minds and Machines, edited Ross Anderson, A. Englewood Cliffs N.J Prentice Hall.

Vidal E, Marzal A, Aibar P (1995), *'Fast Computation of Normalised Edit Distances',* IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol 17, No 9, 926-931.

Weischedel R., Ayuso, D., Bobrow, R., Boisen, S, Ingria, R. and Palmucci, J. (1991), *Partial Parsing*: A Report on Work in Progress, Proceedings of the Speech and Natural Language Processing Workshop, DARPA.

Weisstein E.W. (1998), *CRC Encyclopaedia of Mathematics*, CRC Press, London.

Weizenbaum, J. (1966) *ELIZA. Communications of the ACM.* Vol 9. P36-45.

Weizenbaum, J. (1976). *Computer Power and Human Reason - From Judgement to Calculation.* Freeman: California.

White, J.S., O'Connel, T. & O'Mara, F.E. (1994). *The ARPA MT Evaluation Methodologies: Evolution, Lessons and Further Approaches.* In Technology Partnerships for Crossing the Language Barrier: Proceedings of the First Conference of the Association for Machine Translation in the Americas. Columbia: MD.

Whitehead, A.F., & Russel, B. (1910), *Principa Mathematica*, 2nd edition, Cambridge University Press, Cambridge.

Wicklgren, W.A. (1991) *'The Long and Short of Memory'.* Psychological Bulletin, 80, 425-438.

Winograd T. (1972) *Understanding Natural Languages*, New York Academic.

Winston, P.H. (1975) *Learning structural descriptions from examples.* The Psychology of Computer Vision, ed. Patrick Henry Winston, McGraw Hill.

Woods, W.A. (1975). *What's in a Link: Foundations for Semantic Networks.* In Representation and Understanding: Studies in Cognitive Science, ed Bobrow, D.G.& Collins, A.M. p35-82. Acedemic Press: New York.

Zadeh, L.A., (1974*), Fuzzy Logic and its Application to Approximate Reasoning*, Information Processing 74, p591-594.

Zadeh, L.A., (1982), *Test-score Semantics for Natural Languages*, COLING 82, 425-430.

Zadeh, L.A. (1965*). Fuzzy Sets.* Information and Control. Vol 8 p338-353.

Zobel J, Dart P (1995), *'Finding Approximate Pattern Matches in Large Lexicons'*, Software Practice and Experience, Vol 25, No 3, 331-345.

AppendixA. Sample Questions and Answers

Questions

This test checks whether you have grasped the meaning of certain key oncepts presented in the course so far. The questions are not marked as you sit the test but will be recorded for your tutor to mark and will be returned to you shortly.

Question number 1
Define the term 'low-level language' by completing the sentence 'A low level language is ....'.

Question number 2
Define the term 'high level language' by completing the sentence 'A high level language is...'.

Question number 3
Is COBOL a high or low level language ? State which and explain the reasons for your choice.

Question number 4
Describe the SELECTION JSP structure item given that the other two are SEQUENCE (different types of items following each other in a known order each of which must be processed in a different way by different sections of code in the program) and ITERATION (many items, each of the same type and thus processed by exactly the same piece of code in the program).

Question number 5
"One to One correspondence is ....." : complete this sentence.

Question number 6
Define the term SEQUENTIAL in Jackson Structured Programming (JSP) by completing the sentence "Sequential items are....".

Question number 7
What is an iteration item in terms of Jackson Structured Programming ?

Question number 8

JSP's SELECTION type applies to data/operations that are ....

Question number 9

Leaf items exist on JSP program structure diagrams. Explain what these are be completing the sentence "Leaf items are ....".

Question number 10

List the extra 3 schematic titles resulting from a JSP selection item ALPHA that has three options BETA, GAMMA and DELTA. The schematic logic item ALPHA-SEL would be followed by these three matching 'ALPHA-xxxxx' items. List the three in the correct order.

Question number 11

Read-ahead is a programming technique that allows the program to...what?

Question number 12

A PROCESS item on a program structure diagram infers several processes, list these processes in the correct order of appearance.

Question number 13

Define 'RESIDENT' in COBOL terms by completing the sentence 'A SEGMENT of COBOL code is resident if....'.

Question number 14

Define 'NON-RESIDENT' in COBOL terms by completing the sentence 'NON-RESIDENT segments of COBOL code are.....'.

Question number 15

Subroutines in COBOL are not PERFORM'ed paragraphs or SECTIONs of the main program. Define them by completing the sentence 'Subroutines in COBOL are....'.

Question number 16

Variables inside a COBOL subroutine are unlike variables inside subroutines/procedures in other languages, why?

Question number 17

A flow schema is a diagrammatic representation of what ?

Question number 18

In a flow schema, complex conditions containing sub-conditions OR'ed with each other must be considered as separate sub-conditions. Explain why this is so by completing the sentence "OR'ed expressions must be broken down into their components because ....."

Question number 19

Define the term 'condition list' by providing the remainder of the sentence 'A condition list is...'.

Question number 20

Four ways to reduce the number of test paths resulting from flow schema analysis are often quoted. One relies on eliminating a zero iteration path because of count initialization, two others rely on paths not existing because of nested iterations sharing the same conditions. What does the fourth way of reducing the number of test paths rely on ?

Answers

Question1

Student 1: ONE ones WHICH ALLOWS LOW LEVEL CALLS AND THE USE USE OF REGISTERS AND INTERUPTS

Student 2: A LANGUAGE THAT OPERATES AT PROCESSOR LEVEL

Student 3: a language that translates into machine code at a ratio 1:1

Student 4: one that uses commands at cpu and register level, with non-english commands

Student 5: a language which translates instructions in to code at a ratio of 1:1

Student 6: one in which ONE instruction translates to ONE machine code instruction

Student 7: AN ASSEMBLY LANGUAGE

Student 8: A low level language is asembler.

Student 9: machine orientated rather then user orientated

Student 10: a machine and assemlby code orientated langauge.

Student 11: a language that is easily used by the pc, eg assembler or machine code

Student 12: .... a language which falls closest to machine code type commands.

Student 13: is a one that is closly related to the machine eg Assembler

Student 14: a language which is closely linked to the machines OS & architechture.

Student 15: ASSEMBLY CODE

Student 16:

Student 17: A language written directly to register addresses. i.e Assembler languages.

Student 18: a language that operates directly on the cpu, e.g assembler, machine code

Student 19: A LANGUAGE THAT IS EXTREMELY CLOSE TO THE ACTUAL MACHINE INSTRUCTION.

Student 20: code closest to the machines architecture.

Student 21: a language which relates closely to the computers own language (binary)

Student 22: a language whose syntax is close to the machines instruction set

Student 23: a language that is close to the operating system.

Student 24: one that contains instructions understood by the machine

Student 25: a language that is closer to the computers object code, eg assembler.

Student 26:

Student 27: one which communicates more directly with the machine

Student 28: closer to the actual machine code, such as Assembler for example!!

Student 29: CLOSER TO THE LANGUAGE OF THE OPERATING SYSTEM

Student 30: A low level language is a language using assembly code type commands

Student 31: a machine orientated language that does not require a compiler/translater

Student 32: machine dependant,or machine code, or assembly language

Student 33: close to the machine code as possible and

Student 34: A low level language is a machine set of instruction.

Student 35: a language that talks closely to the computers architecture.

Student 36: one that relates more to what the computer understands rather than humans.

Student 37: a language based on the processors instruction set, eg assembly language.

Student 38: is related closer to the level of machine languages,less user orientated.

Student 39: A low level language is a language like Machine code

Student 40: a language which is closest to real machine code.

Student 41: a set of instructions used to handle files.

Student 42: where instructions are at a low level, close to machine code.

Student 43: on that resembles the computers architecture at a hardware level

Student 44: A low level language is written without the use of mnemonic commands

Student 45: A language thats bears a closely resembles the language the machines uses

Student 46: a language that uses commands native to the machine, for example, Assembler

Student 47: a programming language that uses assembly code (mnemonics) e.g INT 21H

Student 48: a language at a machine code level. eg assembler.

Student 49: one that is closest in syntax to the machines own language eg assembler

Student 50: one that is close to the actual machine language used by the computer

Student 51: a language similar to machine code.

Student 52: A low level lanugage is a language which is closest to the operating system

Student 53: which is like machine code

Student 54: a language that works from the operating system such as assembler

Student 55: an operating system language (assembler) converts into machine code.
Student 56: A LANGUAGE THAT IS, OR IS CLOSE TO THE MACHINES OWN CODE EG BINARY OR ASSEM
Student 57: IS A LANGUAGE THAT USE ASSEMBLY CODE
Student 58: one that uses mneumonics that the programmer and the machine understand
Student 59: 1 that is hard to understand but is faster as the comp can trslte it easier
Student 60: a language that resembles the machines actually machine-code instructions.
Student 61: A LANGUAGE THAT RELIES ON DIRECT CALLS TO THE COMPUTER ARCHITECTURE
Student 62: Machine Code. It is of more meaning to the computer than to a person.
Student 63: nearer to a computers understanding - nearer to machine code
Student 64: A LANGUAGE THAT OPERATES AT PROCESSOR LEVEL

Question2

Student 1: PASCAL, ALLOWS USE OF DATA STRUCTURES BUT NOT INTERRUPT HANDLING
Student 2: A HIGH LEVEL LANGUAGE IS ONE THAT HAS PRE-WRITTEN ROUTINES
Student 3: a langauage that translates in machine code with a ratio 1 : Many
Student 4: one that uses english type commands and has functions to handle cpu calls.
Student 5: a language containing functions relating to English, easier to understand.
Student 6: one in which ONE instruction translates to MANY machine code instrutions
Student 7: COBOL
Student 8: A high level language is cobol.
Student 9: user orientated - easier for the programmer to use.
Student 10: similar to the user's own language (in that its command works are verbs).
Student 11: a language that requires translating into a low level set of code
Student 12: .... a language which is similar to English.
Student 13: one that is in easy to under stand in an english type form. eg COBOL
Student 14: one which is verbose in its instructions, i.e ADD 1 TO G GIVING X
Student 15: A LANGUAGE WHICH IS SIMILIAR TO WRITTEN ENGLISH I.E. PASCAL
Student 16: njk
Student 17: programmer coded language such as COBOL, PASCAL, BASIC etc.
Student 18: a language that converts understandable instructions to machine code
Student 19: CLOSE TO THE HUMAN LANGUAGE, EASY TO READ BY A HUMAN.
Student 20: code nearer to the English language and mathematic expressions.
Student 21: a language which is easily readable by the user no knowledge of cpu needed
Student 22: a language with statements unlike the machines instructions (english like)

Student 23: a language made up of 'English' like statements to make it easier to use.
Student 24: one that needs to be translated to machine (low-level) language
Student 25: a language that is more like standard English, words rather than codes.
Student 26: one that uses english keywords eg read goto etc rather than JMP.
Student 27: one which is interpreted by the machine rather than directly communicating
Student 28: related to english type sentances or words that require further compilation
Student 29: A LANGUAGE THAT REQUIRES COMPILING BEFORE THE O.S. CAN RUN IT
Student 30: a language such as cobol which gives a set of high level commands to use
Student 31: a user orientated language that needs to be compiled to obj code before run
Student 32: a language which is compiled or interpreted before it can be executed
Student 33: where source code is converted into opcode
Student 34: A language which is easily reconized by human and machine.
Student 35: a language which is close to English, eg Pascal.
Student 36: one which relates more to the Human language rather than the machine lang.
Student 37: a verbose language, using words that are english(natural) language
Student 38: more user orientated and user friendly easier to use and programme
Student 39: A high level language is
Student 40: a language that is similar to english, and thus furthest away from M/Code.
Student 41:
Student 42: a language where an instruction represents many machine code instructions
Student 43: one that does not resemble computer architecture
Student 44: one which uses more abstract commands while must be compiled or translated
Student 45: A language that has a structure tailored towards the user and not the comp.
Student 46: a language that uses english words to carry out specific functions
Student 47: a language that has to be converted into machine code before processing.
Student 48: is constructed with easy to read english type statements.
Student 49: one that allows commands that are closer to a human language eg COBOL
Student 50: close to the english language in readablity and grammar
Student 51: similar to english
Student 52: used to create applications to be used on a computer.
Student 53: which is like english,one command has many machine code commands
Student 54: when the language is structured towards a user (PASCAL)
Student 55: one that is highly structured to do certain things when it is run.
Student 56: A LANGUAGE THAT IS ENGLISH LIKE AND EASY TO UNDERSTAND EG PASCAL,COBOL

Student 57: LANGUAGE THAT HAS TO BE CONVERTED TO MACHINE CODE BEFORE PROCESSING

Student 58: one that has to be tokenised by the machine, but is more like english text

Student 59: easy to follow as it uses nearer english comms but must be translated to mc

Student 60: a language that resemble spoken english unlike low lev ones. easier to use

Student 61: A LANGUAGE THAT TRANSLATES 'ENGLISH' WORDS INTO MACHINE CODE

Student 62: of more meaning to a human by using syntax which can easily be understood.

Student 63: one which is near to written english - cobol, pascal

Student 64: A LANGUAGE THAT CAN HAVE OTHER ROUTINES DESIGNED IN IT

AppendixB. Sample Master Document

Full report on answers (set number 1) in file "Allhnd".
Student: "ANGUS LORIMER".

Question file "P2wk8" (last edited at 11:01 25.11.92).
Title : Prog 2 - wk8 test. 20 questions asked.

Test occurred at 13:53 26.11.92 and took 36 mins,39 secs.

Questions

1 : answer is : Mark : 0
"ONE ones WHICH ALLOWS LOW LEVEL CALLS AND THE USE USE OF REGISTERS AND INTERUPTS"

2 : answer is : Mark : 0
"PASCAL, ALLOWS USE OF DATA STRUCTURES BUT NOT INTERRUPT HANDLING"

3 : answer is : Mark : 0
"HIGH,DOES NOT ALLOW LOW LEVEL CALLS OR VIEWING AND USE OF REGISTERS"

4 : answer is : Mark : 0
"SELECTION COULD BE AN 'IF' STRUCTURE IF NOT THIS THEN DO THE OTHER.ITERATIO
N COLULD WHILE NOT EOF DO.I.E. KEEP LOOPING AND PROCESS ITEMS UNTIL END OF
FILE IS REACHED."

5 : answer is : Mark : 0
"WHERE THERE IS A LINK BETWEEN TWO ITEMS OR FILES. E.E. MERGING WILL INVOLVE
A ONO TO ONE CORRESPONDENCE WHERE THERE IS TRANSACTION FILE AND A MASTER F
ILE THERE WILL BE A ONO TO CORRESPONDENCE BETWEEN THE DATA IN BOTH FILES."

6 : answer is : Mark : 0
"ITEMS WHICH FOLLOW ONE ANOTHER IN A GIVEN ORDER"

7 : answer is : Mark : 0
"A LOOP STRUCTURE,WHILE NOT EOF DO.DO PROCCESES UNTIL EOF"

8 : answer is : Mark : 0

"A SELECTION COULD BE AN IF THEN ELSE STRUCTURE. A SELECTION IS ONE WHEN EIT
HER ONE LOT OF INSTUCTIONS ARE CARRIED OUT OR ANOTHER SET OF INSTRUCTIONS A
RE CARRIED OUT.E.G. IF S=1 DO START ELSE DO END. THIS IS A SELECTION STRUCT
URE WHERE START IS PROCESSED IF S =1 ELSE END IS PROCESSED."

9 : answer is : Mark : 0

"THESE ARE ITEMS WHICH ARE USED TO SHOW PROCESSES LIKE OPEN FILES,CLOSE FILE
S ETC THEY FOLLOW A PARTICULAR ORDER AND ARE USED TO SHOW THE ABOVE."

10 : answer is : Mark : 0

"ALPHA-BETA,ALPHA-GAMMA,ALPHA-DELTA"

11 : answer is : Mark : 0

"THIS ALLOWS THE PROGRAM TO READ THE FIRST THE RECORD BEFORE ANY PROCESSING
IS PERFORMED ON THE FIRST RECORD. IT IS USED SO THAT EOF IS NOT REACHED BEF
ORE ALL THE RECOPRDS IN A FILE HAVE BEEN READ. e.G. READ AHEAD THEN DO PROC
ESSING ETC. IN COBOL THE READ AHEAD IS USED FOR THIS PARTICULAR REASON AS O
THERWISE THE EOF FLAG WOULD BE SET BEFORE THE LAST RECORD HAD BEEN PROCESSE
D."

12 : answer is : Mark : 0

"ITERATION,SELECTION"

13 : answer is : Mark : 0

"IT IS PRESENT IN MEMORY/BUFFER.IF IT IS NOT RESIDENT IT HAS TO BROUGHT IN."

14 : answer is : Mark : 0

"WHICH ARE NOT RESIDENT IN THE BUFFER/MEMORY,I.E.DATA/CODE NOT AVAILABLE"

15 : answer is : Mark : 0

"CALLS TO OTHER PARTS OF THE PROGRAM.I.E.IF EOF GO TO END-PARA."

16 : answer is : Mark : 0

"BECAUSE THEY ARE GLOBAL VARIABLES AND NOT ARE NOT JUST WITHIN A PROCEDURE"

17 : answer is : Mark : 0

"THE FLOW OF DATA AROUND A PROGRAM I.E. A PROGRAM TEST SCHEMA"

18 : answer is : Mark : 0

"OTHERWISE IT WILL INVOLVE ALOT OF UNNESSARY TESTS. IF THEY ARE ORED THEN TH

EN THE EXPRESSION AND THE TEST BECOME ALOT SIMPLER AND THE AMOUNT OF DIFFER

ENT PATHS REQUIRING TESTING BECOME LESS AND MORE CLEARER"

19 : answer is : Mark : 0

"IS A LIST OF CONDITIONS AND TESTS THAT HAVE BEEN TESTED OR WILL BE"

20 : answer is : Mark : 0

"IT RELIES ON EOF BEING TRUE THUS NOT HAVING TO TESTED."

0

Total score is 0% (0 out of 0 points).

Full report on answers (set number 2) in file "Allhnd".

Student: "NEILS AJ".

Student name has changed. Original name was: "Neils AJ".

Question file "p2wk8" (last edited at 11:01 25.11.92).

Title : Prog 2 - wk8 test. 20 questions asked.

Test occurred at 17:51 26.11.92 and took 39 mins,5 secs.

Answer data edited at 12:21 27.11.92.

Questions

1 : answer is : Mark : 0

"A LANGUAGE THAT OPERATES AT PROCESSOR LEVEL"

2 : answer is : Mark : 0

"A HIGH LEVEL LANGUAGE IS ONE THAT HAS PRE-WRITTEN ROUTINES"

3 : answer is : Mark : 0

"HIGH. YOU DON'T COMMUNICATE DIRECTLY WITH THE PROCESSOR / OS"

4 : answer is : Mark : 0

"MANY ITEMS EACH OF A DIFFERENT TYPE PROCESSED BY THE PROGRAM DEPENDANT ON

CERTAIN CONDITIONS BEING MET."

5 : answer is : Mark : 0

"WHERE A DATA ITEM FROM ONE FILE MAPS DIRECTLY TO ONE ITEM IN ANOTHER DATA

FILE."

6 : answer is : Mark : 0

"PROCESSED ONE AFTER THE OTHER IN A KNOWN ORDER"

7 : answer is : Mark : 0

"AN ITEM THAT IS REPEATED UNTIL A CONDITION BECOMES TRUE"

8 : answer is : Mark : 0

"DIFFERENT AND DEPENDANT ON CONDITIONS BEING MET"

9 : answer is : Mark : 0

" "

10 : answer is : Mark : 0

"ALPHA-BETA, ALPHA-GAMMA, ALPHA-DELTA"

11 : answer is : Mark : 0

"DETERMINE WHEN THE END OF A FILE IS REACHED BEFORE IT STARTS TO CARRY OUT

THE OPERATIONS ON THAT PIECE OF DATA"

12 : answer is : Mark : 0

"READFILE, DO WHATS TO BE DONE WITH IT, WRITE THE OUTPUT."

13 : answer is : Mark : 0

"IT IS BEING USED AT THE TIME"

14 : answer is : Mark : 0

"SEGMENTS THAT ARE MADE RESIDENT WHEN A CALL GOES TO THEM."

15 : answer is : Mark : 0

"PROCESS es THAT GET CALLED UPON FROM DIFFERENT AREAS OF THE MAIN PROGRAM"

16 : answer is : Mark : 0

"BECAUSE THE STATUS OR VALUE OF THAT VARIABLE IS RETAINED UNTIL THE NEXT AxS"

17 : answer is : Mark : 0

"THE DIFFERENT PATHS A PROGRAM CAN TAKE TO REACH THE END."

18 : answer is : Mark : 0

"EACH OR'ed CONDITION ALLOWS THE PROGRAM TO EXIT VIA A DIFFERENT ROUTE..."

19 : answer is : Mark : 0

"A LIST OF COND'S THAT MUST BE MET IN ORDER FOR THE PROG TO TAKE THAT PATH.."

20 : answer is : Mark : 0

"THE NUMBER OF DAYS LEFT UNTIL XMAS"

0

Total score is 0% (0 out of 0 points).

AppendixC. Sample Knowledge Base File

NextConcept: c_0

Name: q9
Type: BAS
Threshold: 1

Name: C_brach
Type: ABS
Threshold: 1
Association: q9

Name: C_notbelow
Type: ABS
Threshold: 2
Association: q9
Association: not

Name: not
Type: BAS
Threshold: 1
Evidence: not
Association: C_notbelow

Name: below
Type: BAS
Threshold: 1
Evidence: below
Association: C_notbelow

Name: branch
Type: BAS
Threshold: 1
Evidence: branch
Evidence: branches
Association: C_brach

Name: q8
Type: SAT
Threshold: 1

Name: loop
Type: BAS
Threshold: 1
Evidence: loops
Evidence: loop
Association: C_repeat

Name: C_repeat
Type: BAS
Threshold: 1
Association: q7

Name: control
Type: BAS
Threshold: 1
Evidence: control
Association: C_stop

Name: C_stop
Type: ABS
Threshold: 1
Association: q8
Association: q7

Name: q7
Type: SAT
Threshold: 1

Name: condition
Type: BAS
Threshold: 1
Evidence: conditional
Evidence: conditions
Evidence: condition

Association: q4
Association: q8
Association: C_stop
Association: q7

Name: repeat
Type: BAS
Threshold: 1
Evidence: repeadted
Evidence: repeats
Evidence: repeated
Evidence: repeat
Association: C_repeat
Association: q7

Name: process
Type: BAS
Threshold: 1
Evidence: processed
Evidence: process
Evidence: processes
Association: C_other

Name: turn
Type: BAS
Threshold: 1
Evidence: turn
Association: C_after

Name: serially
Type: BAS
Threshold: 1
Evidence: serial
Evidence: serially
Association: C_after

Name: order
Type: BAS
Threshold: 1
Evidence: order
Association: C_other

Name: item
Type: BAS
Threshold: 1
Evidence: items
Evidence: item
Association: C_brach
Association: C_after
Association: C_itemfile

Name: q6
Type: SAT
Threshold: 2

Name: C_other
Type: ABS
Threshold: 1
Association: q6

Name: C_after
Type: ABS
Threshold: 1
Association: q6

Name: after
Type: BAS
Threshold: 1
Evidence: ofter
Evidence: after
Association: C_after

Name: follow
Type: BAS
Threshold: 1
Evidence: follows
Evidence: following
Evidence: follow

Association: C_after

Name: another
Type: BAS
Threshold: 1
Evidence: another
Association: C_other

Name: other
Type: BAS
Threshold: 1
Evidence: other
Association: C_other

Name: human
Type: BAS
Threshold: 1
Evidence: human
Association: C_useown

Name: q2
Type: SAT
Threshold: 1

Name: english
Type: BAS
Threshold: 1
Evidence: english
Association: C_useown

Name: user
Type: BAS
Threshold: 1
Evidence: users
Evidence: user
Association: C_useown

Name: C_useown
Type: ABS
Threshold: 1
Association: q2

Name: assembly
Type: BAS
Threshold: 1
Evidence: assembler
Evidence: assemlby
Evidence: assembly
Association: C_langtype

Name: machine
Type: BAS
Threshold: 1
Evidence: machine
Evidence: machines
Association: C_langtype

Name: computer
Type: BAS
Threshold: 1
Evidence: computer
Evidence: computers
Association: C_langtype

Name: language
Type: BAS
Threshold: 1
Evidence: language
Evidence: langauge
Evidence: languages
Association: C_lang

Name: code
Type: BAS
Threshold: 1
Evidence: code
Association: C_lang

Name: instruction
Type: BAS
Threshold: 1
Evidence: instruction
Evidence: instructions
Association: C_lang

Name: C_langtype
Type: ABS
Threshold: 1
Association: C_lowlang

Name: C_lang
Type: ABS
Threshold: 1
Association: C_lowlang
Association: C_trans1:1

Name: C_lowlang
Type: ABS
Threshold: 2
Evidence: assembler
Association: C_langORarc

Name: C_langORarc
Type: ABS
Threshold: 1
Association: aa

Name: C_nearORuse
Type: ABS
Threshold: 1
Association: aa

Name: C_near
Type: ABS
Threshold: 1
Association: C_nearORuse

Name: C_use
Type: ABS
Threshold: 1
Association: C_nearORuse

Name: C_arc
Type: ABS
Threshold: 1
Association: C_langORarc

Name: register
Type: BAS
Threshold: 1
Evidence: register
Evidence: registers
Association: C_arc

Name: interupts
Type: BAS
Threshold: 1
Evidence: interupts
Evidence: interrupts
Association: C_arc

Name: cpu
Type: BAS
Threshold: 1
Evidence: cpu
Association: C_arc

Name: processor
Type: BAS
Threshold: 1
Evidence: processor
Evidence: processer
Evidence: processer
Evidence: proceser
Association: C_arc

Name: use
Type: BAS
Threshold: 1
Evidence: used
Evidence: using
Evidence: uses
Evidence: use
Association: C_use

Name: operate
Type: BAS
Threshold: 1
Evidence: operate
Evidence: operates
Evidence: uses
Evidence: use
Association: C_use

Name: based
Type: BAS
Threshold: 1
Evidence: base
Evidence: based
Evidence: bases
Association: C_use

Name: allow
Type: BAS
Threshold: 1
Evidence: allow
Evidence: allows
Association: C_use

Name: rely
Type: BAS
Threshold: 1
Evidence: rely
Evidence: relies
Evidence: relys
Association: C_use

Name: depends
Type: BAS
Threshold: 1
Evidence: depend
Evidence: depends
Evidence: dependant
Association: C_use

Name: close
Type: BAS
Threshold: 1
Evidence: closest
Evidence: closely
Evidence: closer
Evidence: close
Association: C_near

Name: similar
Type: BAS
Threshold: 1
Evidence: similar
Association: C_near

Name: at
Type: BAS
Threshold: 1
Evidence: at
Association: C_near

Name: like
Type: BAS
Threshold: 1
Evidence: like
Association: C_near

Name: near
Type: BAS

Threshold: 1
Evidence: near
Evidence: nearly
Evidence: nearer
Association: C_near

Name: type
Type: BAS
Threshold: 1
Evidence: type
Association: C_near

Name: orient
Type: BAS
Threshold: 1
Evidence: orientated
Evidence: oriented
Association: C_near

Name: resemble
Type: BAS
Threshold: 1
Evidence: resemble
Evidence: resembles
Association: C_near

Name: translate
Type: BAS
Threshold: 1
Evidence: translate
Evidence: translates
Association: C_convert

Name: convert
Type: BAS
Threshold: 1
Evidence: convert
Evidence: converts
Association: C_convert

Name: C_convert
Type: ABS
Threshold: 1
Association: C_trans1:1

Name: 1:1
Type: BAS
Threshold: 1
Evidence: 11
Evidence: 1
Evidence: one
Association: C_trans1:1

Name: aa
Type: ABS
Threshold: 2
Association: q1
Association: q3

Name: C_trans1:1
Type: ABS
Threshold: 3
Association: q1

Name: q1
Type: SAT
Threshold: 1

Name: q3
Type: SAT
Threshold: 1

Name: high
Type: BAS
Threshold: 1
Evidence: high
Association: q3

Name: q4
Type: SAT
Threshold: 1

Name: option
Type: BAS
Threshold: 1
Evidence: option
Association: q4

Name: choice
Type: BAS
Threshold: 1
Evidence: choice
Evidence: choices
Association: q4

Name: one
Type: BAS
Threshold: 1
Evidence: one
Association: q4

Name: if
Type: BAS
Threshold: 1
Evidence: if
Association: q4

Name: C_itemfile
Type: ABS
Threshold: 1
Association: q5

Name: file
Type: BAS
Threshold: 1
Evidence: file
Evidence: files
Association: C_itemfile

Name: C_map
Type: ABS
Threshold: 1
Association: q5

Name: map
Type: BAS
Threshold: 1
Evidence: map
Evidence: mapping
Evidence: maps
Association: C_map

Name: link
Type: BAS
Threshold: 1
Evidence: link
Association: C_map

Name: q5
Type: SAT
Threshold: 2

Name: match
Type: BAS
Threshold: 1
Evidence: match
Evidence: matches
Evidence: matching
Association: C_map

Name: compatible
Type: BAS
Threshold: 1
Evidence: compatible
Evidence: compatiable
Association: C_map

Name: relates
Type: BAS
Threshold: 1
Evidence: relate
Evidence: relates
Evidence: relationship
Association: C_map

Name: input
Type: BAS
Threshold: 1
Evidence: input
Association: C_itemfile

Name: correspond
Type: BAS
Threshold: 1
Evidence: corresponds
Evidence: correspond
Association: C_map

x

x

AppendixD. Publications

MPhil to PhD Transfer Publications

From the first part of this thesis where the knowledge schema itself was developed, justified and empirically investigated there have been four resulting publications:

- "Knowledge for Language", poster presented at the AISB94 Workshop. This paper gave an initial analysis of the problems arising from creating a knowledge schema specifically for natural language processing and outlined a general approach.
- "A Knowledge Driven Aid to the Automated Assessment of Free Text", the full paper based on the work above published in the AISBQ Journal. This paper gives gave a more formal specification of the schema under discussion.
- "Automated Assessment: Evaluating a Knowledge Architecture for Natural Language Processing" published in Applications and Innovation in Expert Systems II (1994). This paper develops the argument for using Automated Assessment and an Evaluative Mechanism.
- "Connectionist Pattern Matching for Information Extraction" further develops the knowledge schema and applies it to the problem of information extraction. This was published in the NLP stream of the AI and Soft Computing Series (1997).
-

PhD Publications

From the latter half of this thesis where the network growth algorithms have been developed and investigated there have been another four resulting publications:

- "A Clustering Algorithm to Produce Context Rich Networks" presented at the Advanced Decision Technologies Conference (1995) in the Neural Networks Stream and later published in the book: "Neural Networks and Their Applications". This paper presented an algorithm for generating network consisting of composite nodes using the statistical regularity found in corpora.
- "Sequence Clustering Using Time Delay Networks" given at the ICANNGA (1997) conference. Develops the work above concentrating on the representation and processing of time based information, and outlines an improved algorithm for the construction of such networks.

- "Connectionism and Symbolism in Symbiosis", given at the ICANNGA (1997) conference. Gives a more abstract analysis of the approach taken to the processing of time based data and considers the manner in which the distinct components interact.
- "Composition, Clustering and Predictor Pruning in Hierarchical Networks" published in the Data Mining stream of the AI and Soft Computing Series (1997). Builds upon the algorithm for identifying composite nodes, by adding an algorithm for identifying clustered nodes and outlining a strategy for tree pruning.

Full paper list

1. Allott N, Fazackerley P, Halstead P(1995),'A Clustering Algorithm to Produce Context Rich Networks' Neural Networks and their Applications, ed Taylor J.G., p265-269., John Wiley & Sons, Chichester.
2. Allott N, Fazackerley P, Halstead P(1994),"Knowledge for Language", Postgraduate Workshop AISB94, Leeds.
3. Allott N, Fazackerley P, Halstead P(1994), "A Knowledge Driven Aid to the Automated Assessment of Free Text", AISB Quarterly Journal, Vol 88, p19-24
4. Allott N, Fazackerley P, Halstead P(1994), "Automated Assessment: Evaluating a Knowledge Architecture for Natural Language Processing" Applications and Innovation in Expert Systems II, eds Milne, R. and Montgomery, A., p319-332..
5. Allott N, Fazackerley P, Halstead P(1997),"Connectionist Pattern Matching for Information Extraction", Natural Language Processing for Business Use, AI and Soft Computing Series.
6. Allott N, Fazackerley P, Halstead P (1997), "Sequence Clustering Using Time Delay Networks" accepted ICANNGA97 - International Conference on Artificial Neural Networks and Genetic Algorithms, University of East Anglia, in Artificial Neural Nets and Genetic Algorithms, Ed. Smith, G. Springer, New York
7. Allott N, Fazackerley P, Halstead P (1997), "Connectionism and Symbolism in Symbiosis", accepted ICANNGA97 - International Conference on Artificial Neural Networks and Genetic Algorithms, University of East Anglia. Also published in Artificial Neural Nets and Genetic Algorithms, Ed. Smith, G. Springer, New York

8. Allott N, Fazackerley P, Halstead P (1997), "Composition, Clustering and Predictor Pruning in Hierarchical Networks" Neural Networks for Data Mining, AI and Soft Computing Series, p181-207.

AppendixE.

APPENDIX E

A KNOWLEDGE DRIVEN AID TO THE AUTOMATED ASSESSMENT OF FREE TEXT

N. Allott, P. Fazackerley and P. Halstead

Computing Department,
The Nottingham Trent University,
Burton St,
Nottingham,
NG1 4BU.

ABSTRACT

This paper documents an attempt to produce an automated aid to the assessment and evaluation of student's free text responses to simple questions. The chosen solution involves a comparison of tutor's expectations against student's sentence content. To this end, a nodal based, message passing, hierarchical knowledge representation scheme has been developed in which to embody these expectations. Hand crafted knowledge bases have been applied to real student data. Using scripts that have been hand marked, through a series of stepwise refinements it proved possible to produce knowledge bases which correlated with human marked scripts upwards of 85%. By applying knowledge bases produced this way to unseen scripts correlation rates of 65% were achieved.

1 - INTRODUCTION

What is the need for automated marking? Firstly, speed: in today's climate of reducing staff/student ratios automated marking frees up staff time for more all important student contact time. Also marking turnaround is greatly reduced giving the students feedback quicker. Secondly, objectivity: by marking all scripts with a single machine, a standard is introduced where formerly marker variability (both between different markers and the same marker at different times) is notoriously high. Thirdly, focus: it enforces a stricter definition of the syllabus which must be advantageous to staff and student alike. Students are more aware of what they are expected to learn, and tutors aware of the knowledge they are supposed to impart.

The automated marking of multiple choice scripts using graphite sensitive machines is now widely used, even in many formal examinations. The question style however is limited in the type of knowledge it can be used to test and is somewhat at odds with the more traditional free text type questions [5],[6] and [7].

Marshall's Intelligent Marking Assistant [1] documents a system to aid in the marking of student's essay style questions. The broad outline is to break down the problem of

Published in: AISB Quarterly Journal (1994)
Vol 88, p19-24

assessment into a series of problems: in effect qualitative judgements made by a human on various aspects of the text. These low level judgements are combined by sets of rules into a quantitative evaluation of the text's worth, often with attached *canned* comments. There are now several commercial systems on the market which work along similar lines

We set ourselves the same broad aim as the above, that is removing the human bottleneck from the marking process. Our emphasis, however, is something of a compromise between multiple-choice and essay marking. We wish for an objective evaluation of the content of the student response but we wish these responses to be of the more traditional unconstrained free text form. For simplicity we have constrained ourselves initially to single sentence answers only.

Implicit in any solution to the proposed problem is that aspects of the language processing and knowledge representation tasks previously performed by the marker must be transplanted into the computer. It is our view that with the right knowledge architecture such processes may be embodied within the computer and can model the marking process to a statistically significant degree.

2 - ANALYSIS OF PROBLEM: KNOWLEDGE FOR LANGUAGE

To clarify the problem and to provide data on which various theories could be tested, a series of short questions were given to second year HNC students on a general programming nature. Some typical questions:

Define a low level language?

One to one correspondence is...?

Define the term sequential in Jackson structured programming?

Read-ahead is a programming technique that allows the program to ...?

Define non-resident in COBOL terms?

A sample of typical replies to the first of these questions (correct and incorrect):

A language that translates into machine code at a ratio 1:1

One that uses commands at the cpu and register level with non english commands

A language which falls closest to the machine code type commands

A language that operates directly on the cpu eg. assembler and machine code

An assembly language.

Machine orientated rather than human orientated.

A language that is easily used by the pc.

A language that is closely related to the machine OS.

Published in: AISB Quarterly Journal (1994)
Vol 88, p19-24

APPENDIX E

A careful study of the form and quality of our incoming data uncovered several features of language that our knowledge architecture must be capable of handling. These features are:

2.1 - Synonymy

The representation of alternatives is essential not only on the word level (i.e. synonyms: many words mean the same thing) but at the clause level also. To reformulate our problem in linguistic terminology, we are attempting to produce a mapping from the surface structure to deep structure. In this context our architecture must be able to model a *many-to-one* relationship.

2.2 - Polysemy/Homonymy

Some words have multiple meanings. The knowledge architecture must not only be able to represent this multiplicity but also provide a mechanism for disambiguation. In the context of a surface to deep mapping it is a *one-to-many* relationship.

2.3 - Idioms

At many levels natural language resists a compositional analysis, where the implication is the whole is the sum of its parts. An idiom is an example of this where the meaning of a phrase can not be built up from the meaning of the component words.

2.4 - Reasoning

Our knowledge architecture must have the capacity to reason. To reiterate our aim we wish to reason from the evidence of student's sentence whether it belongs to the set of sentences that mean the same as the tutor's definition of a correct answer. We must make an implementation decision whether this reasoning capacity is to be embodied in an external module which is applied to the knowledge or whether it become an implicit aspect of the architecture itself.

2.5 - Intuitive interaction

Although later research may produce more sophisticated methods for building our knowledge bases for the sake of simplicity the K-Base for initial systems will necessarily be hand edited. It is essential therefore that it be possible to represent the knowledge base in a clear intuitive manner, preferably graphically.

3 - OVERRIDING DESIGN PROBLEMS

In order to produce a system that both encourages data exchange from marker to marking system and that can reasonably model the marking process certain design issues need to be resolved before an implementation can proceeded. These are:

3.1 - Segmentation of KR and NLP processes

Published in: AISB Quarterly Journal (1994)
Vol 88, p19-24

APPENDIX E

Knowledge Representation and Natural Language Processing are traditionally thought of as two distinct but closely associated disciplines. In this application they prove difficult to separate. Not only is the natural script is being used to build and test the knowledge structures, but the language processing, itself, is a knowledge intensive process.

In fact here the two have been embodied in the same simple but powerful underlying tree-like architecture. The distinction, if one is to be made, is one of degree rather than absolute. Knowledge pertinent to language processing, to do with the status and category of a word, is held low down in the tree structure. Knowledge of a higher level which embodies facts etc. is held high up in the tree.

3.2- Contribution of Syntax

To illuminate our problem look at the following example. All sentences below have similar meanings however the key concept *love* occurs as noun, verb, adverb and adjective.

Romeo loved Juliet

Romeo felt great love for Juliet

Romeo felt lovingly towards Juliet

Romeo had a loving feeling for Juliet

It is widely assumed that syntactic parse is a necessary precursor to any semantic analysis that is to take place. It is not deemed necessary in this case for the following reasons. Firstly, we are dealing with sentences in a highly constrained context, in that they all answer the same question. Any meaning to be usefully extracted from the grammatical status of a word may be safely inferred from context. Secondly, grammatical analysis is an extremely difficult problem in its own right. Finally, real world, untidy data is being handled; replies are ungrammatical and badly spelt - a reliable parse would be impossible.

When we are attempting to crudely match the meaning of one sentence against another where the context of the sentence is heavily constrained, syntax and the issues it raises seem best ignored, with little apparent effect on the eventual performance of the system.

3.3 - Understanding vs Evidential Reasoning

It is our aim simply to discriminate between correct and incorrect responses to a question. To achieve this true *understanding* is not, we believe, necessary. In this NLP problem where the context of the responses is highly constrained, it is possible to generate a strong expectation of what will be submitted. The underlying strategy then is to explicitly set down these expectations within the representational scheme offered by the chosen knowledge architecture. The submitted student sentence is then proffered as a set of evidence that either confirms or refutes this expectation.

The strategy then serves to focus information on the pertinent information only. It carries with it, however, the disadvantage of being unable to perceive what it can not expect.

3.4 - Relation - Object Distinction

Published in: AISB Quarterly Journal (1994)
Vol 88, p19-24

APPENDIX E

Careful consideration must be taken of the respective epistemological status of the knowledge architecture and the knowledge base. (Here I use the term knowledge base to mean the information that is entered following the rules prescribed by the architecture). Common within many knowledge formalisms is an implicit Relation - Object distinction at the architectural level. This is undesirable for our purposes for the following reasons.

Firstly, it is difficult to express knowledge about relationships in the same way knowledge that we express knowledge about objects. Similarly complications arise if we try to represent knowledge about facts. With an implicit object relation distinction a fact becomes a relation between n-objects. To then represent knowledge about facts we either give our (relation,object,object,..) the status of an object or introduce a new entity type a fact and enhance the syntax (rules for combing entities) to allow facts into the (relation,object,object,..) construct. Either way it becomes the structure becomes untidy. Thirdly we may wish to make a type/token distinction between relationships as we almost certainly do with objects. By removing the distinction we acquire a greater consistency.

By such a move nothing need by lost, should it become essential to make such a distinction it is always possible to make this at the knowledge base level as opposed to the architectural.

4 - ALTERNATIVE TECHNIQUES

From the various literature on natural language processing ([2][10] among others) knowledge formalisms applicable to our task were identified. The subject area is enormous, we therefore restricted our considerations to the major different *breeds*. Logics are a popular approach. In their favour they are highly formalised and have powerful well established reasoning mechanisms. However they are hard to understand and difficult to visualise by the layperson. Frames [9] are easier to visualise and are suited for the representation of high level knowledge. But as with the various forms of semantic nets [5] [3], their many sub-types are frequently inconsistent, and do not possess the formal qualities of logics which make it easy to spot the principal components and their status (slot, filler, node, arc, event etc.) from within a sentence. Further, these slot and filler techniques do not have well established reasoning techniques. The work of Collins and Loftus [8] on spreading activation and work from the connectionism field as a whole [11] have strongly influenced our solution to this reasoning problem.

5 - KNOWLEDGE ARCHITECTURE

The knowledge architecture opted for is similar in many ways to a semantic net, but has borrowed heavily from all the above. An emphasis has been put on simplicity, as the user has to interact directly with it. The fundamental distinction between semantic nets and the chosen architecture is that links are not labelled. A relationship between two objects is instead represented by the two objects along with a third object, representing the relationship, all linking to a higher abstract node. This higher node serves to group the constituents together and provide an abstract representation of (or a shorthand for) the fact itself.

5.1 - Node Characteristics

Each node has the following characteristics:

- Name (identifier)
- Activation value
- Threshold value
- List of parents (to which activation passed)
- List of children(from whom activation sent)

Published in: AISB Quarterly Journal (1994)
Vol 88, p19-24

APPENDIX E

- **List of evidence(words which trigger activation of evidence nodes)**

5.2 - Node's Functional Capacities

Each unit is incorporated into a hierarchical tree type structure. Within nodes tend to operate in one of three capacities:

5.2.1 - Evidential

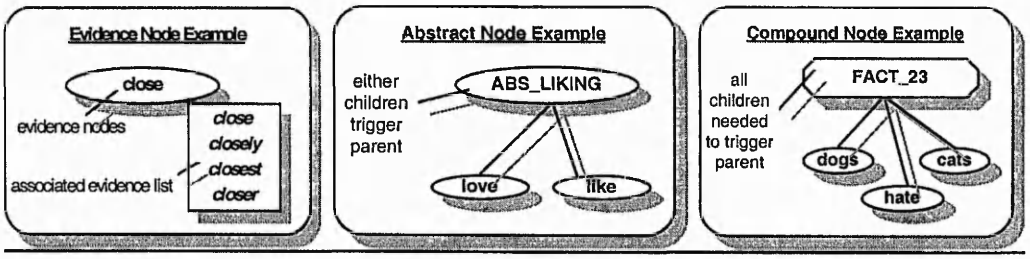
Evidential nodes are used to represent the low word level knowledge. Each has associated with it a list of words (evidence list) whose presence in the sentence will lead to the activation of the node. An evidential node is analogous to a lexeme.

5.2.2 - Abstract

An abstract node can be used to represent the abstract sense of several different nodes. It is then a disjunction of nodes and will become active if ANY of its children become active. In its simplest sense can be used to represent synonyms. (Logical OR).

5.2.3 - Compound

A compound node is used to represent the compound sense of several nodes. It is a conjunction of nodes and becomes active only if ALL of its children become active. A fact is represented with a compound type as a conjunction of abstract types. (Logical AND).



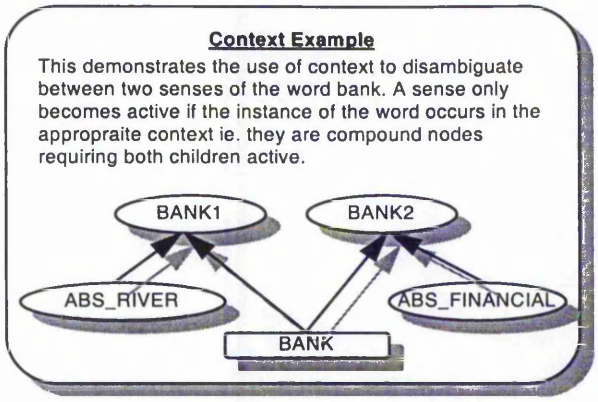
5.3 - Overall Structure

The knowledge is embodied in a nodal based, hierarchically structured message passing structure. The reasoning strategy has becomes implicit in the message passing aspect of the structure. So to judge a sentence each component word is presented to each evidence node. If the resulting flow of activation ends with the top level compound node becoming active the answer is deemed correct.

The abstract node and complex node are respectively responsible for representing synonyms and facts. A complex node can however also be used to tackle the problem of polysemy mentioned above.

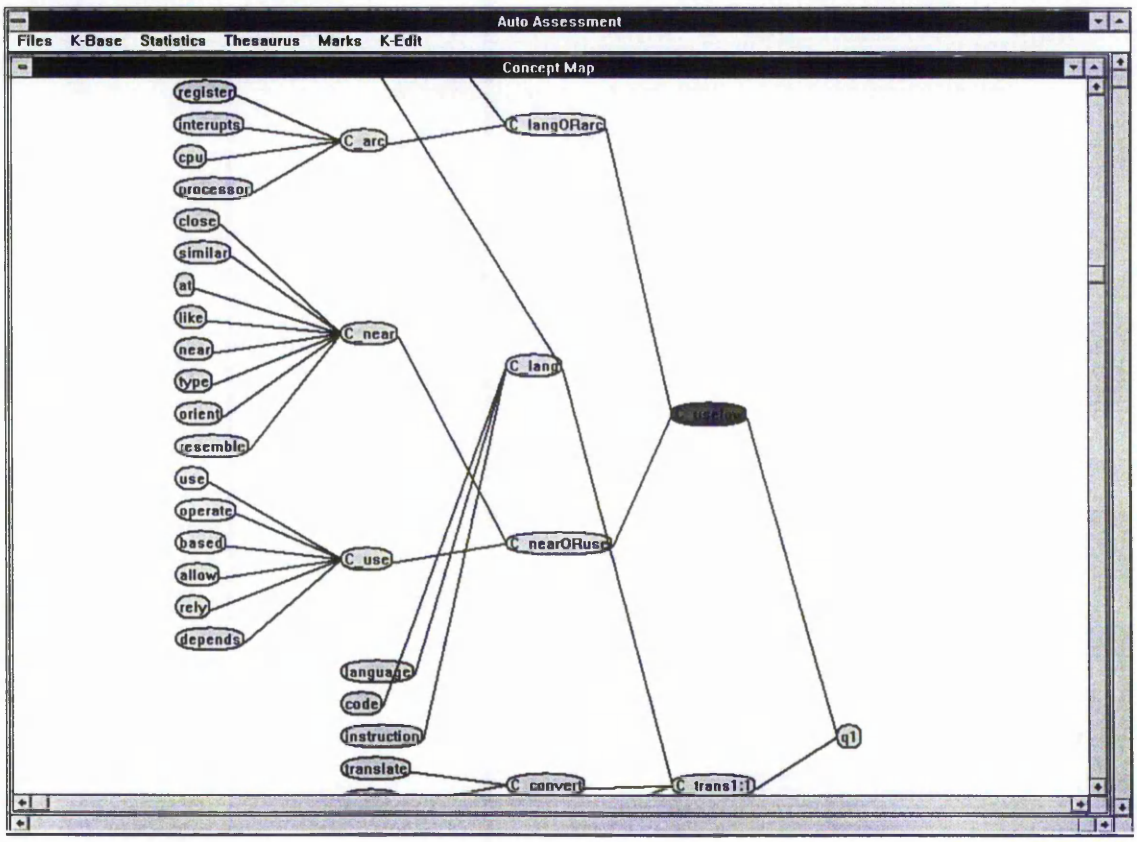
Published in: AISB Quarterly Journal (1994)
Vol 88, p19-24

APPENDIX E



Here the topology of the nodes and the required threshold levels allow us to insist on context for a node to become active. Precisely the same principles may be used to model idiomatic phrases.

5.4 - Sample K-Base



6 - TESTING

Published in: AISB Quarterly Journal (1994)
Vol 88, p19-24

APPENDIX E

In order to test the various theories put forward and to test the sophistication of the knowledge architecture a complete trial was performed. This involved generation of the knowledge base from scratch, then applying this to real student responses.

The generation of this knowledge base was facilitated by a fully integrated graphical editor. The knowledge base has all the qualities described earlier. Characteristics of individual nodes are edited with a double click of the mouse, and the overall network topology is edited using drag and drop techniques. A thesaurus has also been added to the system to help the user develop the knowledge base.

Two experiments were performed on the system:

6.1 - Experiment 1 - Retrospective

The first set of data (replies to the 20 short questions on programming) were initially hand marked. A knowledge base was then constructed which embodied all the qualities that the correct answers had in common. The answers were then applied to the network. Since we had the hand marked results also, we could correlate the automatic judgements against the manual and identify exceptions. With a series of stepwise refinements the knowledge base could be modified until a good model of the human marker's performance was reached.

Using this technique a good model (average correlation 85%) could be produced for each question in less than 5 minutes. With slightly longer construction time correlation rates approaching 100% could be achieved in almost all cases. In order to approach these higher correlation rates, many of the *fringe* answers need to be incorporated into K-base. Interestingly many of the problems in incorporating the last few answers were attributable to spurious decisions on the part of the marker rather than genuine difficulties in extending the K-base.

The first experiment was essentially a test of the expressive sophistication of the knowledge architecture. That is, was it possible to construct a structure which could discriminate between right and wrong answers? The above results seem to confirm that this is possible. However, an obvious danger when constructing these knowledge bases is that we are arbitrarily discriminating between right and wrong answers, whereas we hope we are synthesising a K-base which truly embodies generalised elements of the problem domain. Further, this experiment does not prove the system is usable as in normal scenarios we would not have the answers pre-hand marked.

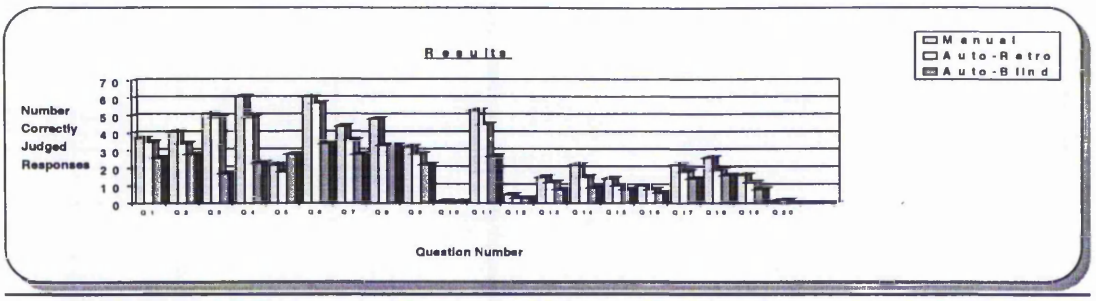
6.2 - Experiment 2 - Blind

There is no obvious method for determining the arbitrariness of a constructed knowledge base at face value. However to get some measure of the generality of the K-base we could apply constructed bases against new unseen data and compare correlation rates. This is what has been done here. Knowledge bases constructed in Experiment 1 were applied to new data (actually the responses of the following years students). This data set was then hand marked and coverage rates compared.

When the knowledge base created in the previous phase was applied to a blind set of data an average coverage rate of 65% was obtained. Considering the present limitations of the system this is considered very encouraging.

Published in: AISB Quarterly Journal (1994)
Vol 88, p19-24

APPENDIX E



6.3 - Comments

The automatic marker systematically marked fewer answers correct than the manual mark. It is also noticeable that very rarely does the system mark an incorrect response correct. In a real application we can be reasonably certain (in our tests 98%) that a reply marked correct will indeed be correct. We can legitimately focus future efforts on capturing those unexpected correct replies.

When constructing knowledge bases our major problem is anticipating the variety of language which can be used to reply to a question. We have partially overcome that here by insisting on two runs of system. One on a set of prototype information to help generate these anticipations and a second true run. The next logical step in reducing the time required to generate the knowledge base and to improve coverage rates is to introduce secondary knowledge sources.

7 - FURTHER DEVELOPMENT

The following have been identified as possible areas for further development:

7.1 - Running Parallel NLP Tasks Under the Same Architecture

Within the same underlying architecture it may be possible to implement many of the other natural language processing tasks, allowing techniques to be run in parallel: sharing the same information, and providing a mechanism for combing results.

- Syntactical analysis: a bottom up parse would be ideal for such an architecture. Further, contextual information is easily added (as in semantic processing) to augment its power. Variable node activation and mutually inhibitory connections would provide ambiguity resolution.
- Spelling correction and word recognition: by producing a similar architecture for the sub word level (i.e. dissolving words into hierarchically structured letter clusters) an information rich, context sensitive mechanism for word recovery may be provided. An algorithm for producing such sub-word structures has already been developed.
- Semantic analysis: more theoretical semantic methods such as componential analysis may also be implemented within the same architecture.

7.2 - Introducing Secondary Information Sources

In order to reduce the considerable overhead involved in generating the knowledge base and to improve the performance by utilising information about language variability secondary information sources need be integrated into the system. There are two immediate sources that have been identified:

Most immediately thesauri: these essentially give us synonyms i.e. words that mean the same thing. This is what abstract nodes within the knowledge base most frequently embody. A thesaurus has already been implemented within the GUI as an aid to the

marker. It is a relatively simple step to embed the information produced this way automatically into the network. WordNet has also been identified as a possible supplier of such information. It has a richer hierarchical description of word meanings than conventional thesauri

Associated work within the department has produced extensive word-lexeme maps. This is precisely what is needed to produce the evidence lists required by the evidence nodes. This will reduce the Knowledge Base construction phase by another order of magnitude.

7.3 - Enhancing the architecture

Finally, the present system is limited in that node activation and message passing is a Boolean all or nothing event. Fuzzy data would be better processed by introducing some variability to the activation and hence the decision making processes. Also message passing may be enhanced by allowing messages to be passed down and across the network. This would give stronger ambiguity resolution via a winner take all strategy implemented by mutually inhibitory connections between sibling nodes.

8.0 - Conclusions

Automatic marking is a problem that serves as an ideal testbed for integrated knowledge representation and natural language processing theories. This is because (a) the domain is highly constrained, and (b) we have a large sample of possible sentences which when processed may then be correlated against a human decision to give a good statistical measure of the systems reliability.

Our solution to the problem has required the development of combined NLP and KR architecture, which has borrowed heavily from existing techniques. Any success achieved is attributable largely to our refusal to consider the problems of grammar, justified we believe by the constrained nature of expected responses. Our results at this stage are very encouraging and seem to confirm many of the assumptions we have made in our approach to the problem.

We have identified areas for future work which can be summarised briefly as (a) improve the reasoning mechanism of the architecture and (b) reduce the time and effort required to produce the knowledge bases. Our meter of success (i.e. correlation rates) is in place.

REFERENCES

[1] Marshall S. M., "An Intelligent Marking Assistant: An Application of Artificial Intelligence in Teaching", Higher Education Research and Development, p201-211, 1986.
[2] Winograd T., Understanding Natural Languages, New York Academic, 1972.
[3] Anderson J. R., The Architecture of Cognition, Cambridge: Harvard University Press, 1983.
[4] Collins A.M. and Loftus E.F., "A Spreading Activation Theory of Semantic Processing", Psychological Review, 85, 407-28, 1975.
[5] Gronlund N. E., Measurement and Evaluation an Teaching, NewYork: Macmillan, 1981.
[6] Chase C. I., Measurement for Educational Evaluation, Addison Wessley, 1978.
[7] Nelson C.H., Measurement and Evaluation in the Classroom, New York: Macmillan, 1970.

Published in: AISB Quarterly Journal (1994)
Vol 88, p19-24

APPENDIX E

[8] Quillian R., "Semantic Memory" in Semantic Information Processing, editor Minsky M., Cambridge: MIT, 1968.
[9] Minsky M., "A Framework for Representing Knowledge" in The Psychology of Computer Vision, editor Winson P., NewYork: McGraw Hill, 1975.
[10] Schank R.C. and Colby K.M., Computer Models of Thought and Language, San Francisco: Freeman, 1973.
[11] Rumelhart D.E. and McClelland J., Parallel Distributed Processing, Cambridge: MIT, 1968.

Published in: AISB Quarterly Journal (1994)
Vol 88, p19-24

AppendixF.

APPENDIX F

Automated Assessment: Evaluating a Knowledge Architecture for Natural Language Processing

N. Allott, P. Fazackerley and P. Halstead
email: nma@uk.ac.ntu.doc

Computing Department,
The Nottingham Trent University,
Burton St,
Nottingham,
NG1 4BU.

ABSTRACT

This paper describes a novel knowledge architecture for the representation of semantic knowledge for natural language processing. This architecture is not only a powerful knowledge description language but offers integrated processing capabilities along either symbolic or connectionist paradigms. We attempt to evaluate the architecture by applying it to the real world problem of the automated assessment of student's free text responses to simple questions. Using scripts that have been hand marked, through a series of stepwise refinements it proved possible to produce knowledge bases under this architecture which correlated with human marked scripts upwards of 85%. By applying knowledge bases produced this way to unseen scripts correlation rates of 65% were achieved. To facilitate the rapid development of such knowledge bases we describe a method of interfacing to a version of Roget's thesaurus.

1 - INTRODUCTION

The automated assessment of multiple choice style questions and more recently the style and correctness computer programs are well researched fields that are now mature enough to be incorporated in numerous commercial packages. The assessment of free text style questions is a natural extension to this but notoriously difficult due to variability of natural language and its resistance to formal analysis. However, this problem of evaluating free text, we believe, forms the ideal testbed for integrated Natural Language Processing(NLP)/ Knowledge Representation(KR) theories.

To illustrate: by presenting a question to group of students we can produce a large sample of simple sentences in a highly constrained domain and of which a fair percentage will have identical meanings. If we then present these

Published in: APPLICATIONS AND INNOVATION IN EXPERT SYSTEMS II (1994)
Eds Milne, R. and Montgomery, A. p319-332

student replies to both human and automatic markers, and compare results, we have an elegant controlled experiment which gives an empirical measure of the sophistication of the knowledge base.

It is implicit in any attempt to construct an automatic marker that aspects of the language processing and knowledge representation tasks previously performed by the marker must be transplanted into the computer. In order to anticipate the quality and form of the language we require our KR/NLP modules to handle, a series of short questions were given to second year HNC and HND students on a general programming nature. Some typical questions used for this purpose:

Define a low level language?
One to one correspondence is...?
Define the term sequential in Jackson structured programming?
Read-ahead is a programming technique that allows the program to ...?
Define non-resident in COBOL terms?

A sample of typical replies to the first of these questions (correct and incorrect):

A language that translates into machine code at a ratio 1:1
One that uses commands at the cpu and register level with non english commands
A language which falls closest to the machine code type commands
A language that operates directly on the cpu eg. assembler and machine code
An assembly language.
Machine orientated rather than human orientated.
A language that is easily used by the pc.
A language that is closely related to the machine OS.

Our first step in finding a solution to the problem was a survey of the various literature on natural language processing ([1][13][14][2][10]) in order to identify knowledge formalisms applicable to our task. The subject area is enormous, we therefore restricted our considerations to the major different *breeds*. Logics are a popular approach. In their favour they are highly formalised and have powerful, well established reasoning mechanisms. However they are hard to understand and difficult to visualise by the layperson. As in initial versions it is anticipated that the tutor will have to interact directly with the knowledge base this is considered an important feature. Frames [9] are easier to visualise and are suited to the representation of high level knowledge. But as with the various forms of semantic nets [5] [3], their many sub-types are frequently inconsistent, and do not possess the formal qualities of logics which make it easy to spot the principal components and their status (slot, filler, node, arc, event etc.) from within a sentence. Further, these slot and filler techniques do not have well established reasoning techniques. The work of Collins and Loftus [8] on spreading activation and work from the connectionism field as a whole [11] have strongly influenced our solution to this reasoning problem.

Published in: APPLICATIONS AND INNOVATION IN EXPERT SYSTEMS II (1994)
Eds Milne, R. and Montgomery, A. p319-332

2 - DESIGN ISSUES

Before implementation could proceed, in order to produce a system that both encourages data exchange from marker to marking system and that can reasonably model the marking process certain design issues need to be resolved. These are:

2.1 - Segmentation of KR and NLP processes

Knowledge Representation and Natural Language Processing are traditionally thought of as two distinct but closely associated disciplines. In this application they prove difficult to separate. Not only is the natural script being used to build and test the knowledge structures, but the language processing, itself, is a knowledge intensive process.

2.2- Contribution of Syntax

To illuminate our problem look at the following example. All sentences below have similar meanings however the key concept *love* occurs as noun, verb, adverb and adjective.

Romeo loved Juliet
Romeo felt great love for Juliet
Romeo felt lovingly towards Juliet
Romeo had a loving feeling for Juliet

It is widely assumed that a syntactic parse is a necessary precursor to any semantic analysis that is to take place. It is not deemed necessary in this case for the following reasons. Firstly, we are dealing with sentences in a highly constrained context, in that they all answer the same question. Any meaning to be usefully extracted from the grammatical status of a word may be safely inferred from context. Secondly, grammatical analysis is an extremely difficult problem in its own right. Finally, real world, untidy data is being handled; replies are ungrammatical and badly spelt - a reliable parse would be impossible.

2.3 - Understanding vs. Evidential Reasoning

It is our aim simply to discriminate between correct and incorrect responses to a question. To achieve this true *understanding* is not, we believe, necessary. In this NLP problem where the context of the responses is highly constrained, it is possible to generate a strong expectation of what will be submitted. The underlying strategy then is to explicitly set down these expectations within the representational scheme offered by the chosen knowledge architecture. The submitted student sentence is then proffered as a set of evidence that either confirms or refutes this expectation.

The strategy then serves to focus information on the pertinent information only. It carries with it, however, the disadvantage of being unable to perceive what it can not expect.

Published in: APPLICATIONS AND INNOVATION IN EXPERT SYSTEMS II (1994)
Eds Milne, R. and Montgomery, A. p319-332

2.4 - Relation - Object Distinction

Careful consideration must be taken of the respective epistemological status of the knowledge architecture and the knowledge base. (Here I use the term knowledge base to mean the information that is entered following the rules prescribed by the architecture). Common within many knowledge formalisms is an implicit Relation - Object distinction at the architectural level. This is undesirable for our purposes for the following reasons.

Firstly, it is difficult to express knowledge about relationships in the same way that we express knowledge about objects. Similarly complications arise if we try to represent knowledge about facts. With an implicit object-relation distinction a fact becomes a relation between n-objects. To then represent knowledge about facts we either give our (relation,object,object,..) the status of an object, or introduce a new entity type a fact and enhance the syntax (rules for combing entities) to allow facts into the (relation,object,object,..) construct. Either way it becomes the structure becomes untidy. Thirdly we may wish to make a type/token distinction between relationships as we almost certainly do with objects. By removing the distinction we acquire a greater consistency.

By such a move nothing need by lost, should it become essential to make such a distinction it is always possible to make this at the knowledge base level as opposed to the architectural.

3 - KNOWLEDGE ARCHITECTURE

The knowledge architecture opted for is similar in many ways to a semantic net, but has borrowed heavily from all the mainstream KR schemes. As the user has to interact directly with it, an emphasis has been put on simplicity. The architecture is essentially a hierarchically structured activation passing network. Such an architecture was chosen because it seems to model well the aspects of natural language which seem most important.

As was outlined in the design issues section no distinction has been made between the KR and NLP levels. Also the architecture departs from the symbolic paradigm where a distinction is conventionally made between data representation and an engine used to process it. The architecture corresponds more closely the connectionist paradigm where the reasoning capacity is embodied within the activation passing aspect of the network itself.

The following have been identified as properties of an activation passing network which make it particularly appropriate for natural language processing (see figure 1).

3.1 - Abstraction: many words in certain contexts have similar meanings (i.e. synonyms). This abstract similarity may be adequately modelled by an interconnected network in which any ONE of a number of child nodes

Published in: APPLICATIONS AND INNOVATION IN EXPERT SYSTEMS II (1994)
Eds Milne, R. and Montgomery, A. p319-332

(synonymous words) may trigger the activation of a parent node (synonymous group/ abstract type).

3.2 - Composition: certain words have a composite sense which is distinct from the conjugate sense of its parts, for example idioms. (Not as much the whole is greater that the sum of its parts but the whole is different.) This phenomena may be modelled by a network where a single parent node requires ALL of its children nodes to become active before it itself activates.

3.3 - Decomposition: some words have a composite structure where its parts contribute different senses to the meaning of the word as a whole. For example, at the morphological level consider the word walk-ed: 'walk' tells us the action, 'ed' tells us it happened in the past. Or at the semantic level, the word husband, where we may infer that the entity referenced is (male), (human) and (married). Both these cases could be represented by a child node which passes its activation to several parent nodes.

3.4 - Parallelism: closely associated with decomposition is the concept of parallelism whereby the separate implications (modelled by spreading activation) of the separate items may be computed concurrently.

3.5 - Context: there are many words that have multiple meanings. In a particular case it is only possible to identify which meaning is applicable by looking at the context. This is the linguistic phenomena of polysemy or homonymy. We may model this in a network by a single child node linking to two parent nodes in parallel. Both of these parents will require input from a node representing the correct context also before it, itself, becomes active. Take the classic example of 'bank', in a financial context it has a completely different meaning to a water/countryside context.

3.6 - Disambiguation: the mutual exclusivity of two hypothesis or perhaps two word meanings may also be modelled by an activation passing network. Using the bank example again it will mean either a 'commercial building' or the 'side of a river' not both. Mutually inhibitory links between sibling nodes ensure that only the node receiving most supporting evidence remains active.

3.7 - Feedback: sometimes the information necessary to disambiguate two hypothesis is only available at a higher level process, in order to make use of this information it is necessary to implement feedback links whereby a parent node can affect the activation of a child.

3.8 - Arbitrary Mapping: as has been demonstrated in numerous connectionist papers a three layer activation passing network is capable of embodying an arbitrary mapping between any two sets of patterns provided there are enough nodes. Further, the Universal Approximation Theorem demonstrates that a three layer network with one input, one output and sufficient nodes in the hidden layers can learn any function. This adds powerful computational power to our simple knowledge architecture.

Published in: APPLICATIONS AND INNOVATION IN EXPERT SYSTEMS II (1994)
Eds Milne, R. and Montgomery, A. p319-332

3.9 - Logical Composition: by virtue of the fact that all logical gates (OR, AND, NOT etc.) can be modelled by an interconnected network, and a standard vonNeuman architecture computer can be implemented entirely with such gates it should be possible to emulate any formal computation within the resource constraints.

The above points demonstrate that our chosen knowledge architecture has not only powerful representational flexibility but is capable of performing computation under both connectionist and symbolic processing paradigms.

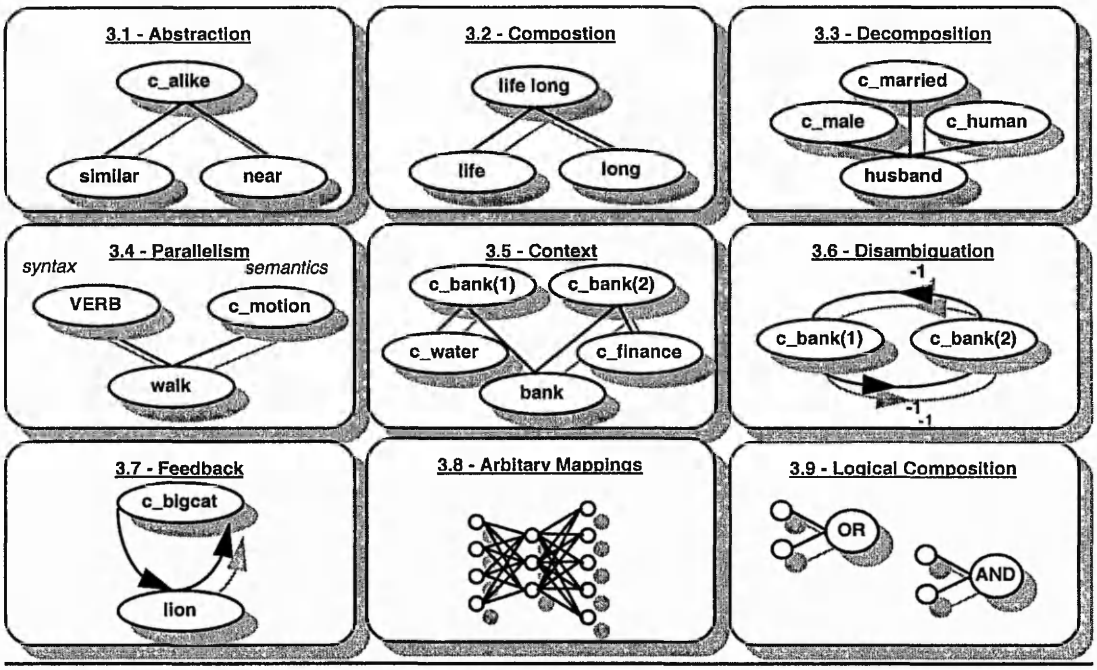


Figure 1: Network Qualities

4 - NODE CHARACTERISTICS

The prototype implementation of the knowledge architecture has most but not quite all of the above specified features. The file description of the individual nodes specify the minimal time-invariant properties.

- Name: this is a unique key used to identify a particular node.
- Threshold value: is the value a node's input must exceed before the node itself becomes active.
- List of parents: a list of nodes to which activation is passed when the node itself becomes active (referenced by name).
- List of evidence: a list of words which if presented to the KR module will lead to the activation of that node.

Upon loading a knowledge base into memory all references are resolved into memory pointers. Also two further node attributes are created.

- Activation: the nodes activation, at present calculated as the sum of all inputs.

Published in: APPLICATIONS AND INNOVATION IN EXPERT SYSTEMS II (1994)
Eds Milne, R. and Montgomery, A. p319-332

- List of children: essentially the mirror image of list of parents i.e. all nodes from which input is received. This redundancy was introduced at the data level for reasons of procedural efficiency.

4.1 - Node's Functional Capacities

In typical usage nodes tend to operate in one of three capacities (see figure2):

4.2 - Evidential: Evidential nodes are used to represent the low word level knowledge. Each has associated with it a list of words (evidence list) whose presence in the sentence will lead to the activation of the node. An evidential node is analogous to a lexeme.

4.3 - Abstract: An abstract node can be used to represent the abstract sense of several different nodes. It is then a disjunction of nodes and will become active if ANY of its children become active. In its simplest sense can be used to represent synonyms. (Logical OR).

4.4 - Compound: A compound node is used to represent the compound sense of several nodes. It is a conjunction of nodes and becomes active only if ALL of its children become active. A fact is represented with a compound type as a conjunction of abstract types. (Logical AND).

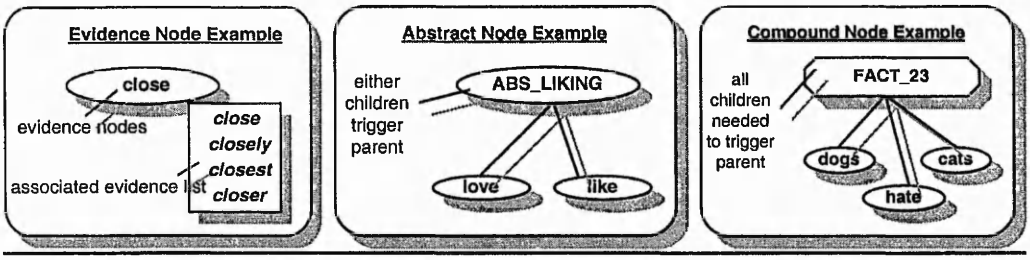


Figure 2: Node's Functional Capacities

5 - IMPLEMENTATION:

The following outlines the final implementation and how the modules which comprise the system interact with each other (see figure 3).

Published in: APPLICATIONS AND INNOVATION IN EXPERT SYSTEMS II (1994)
Eds Milne, R. and Montgomery, A. p319-332

APPENDIX F

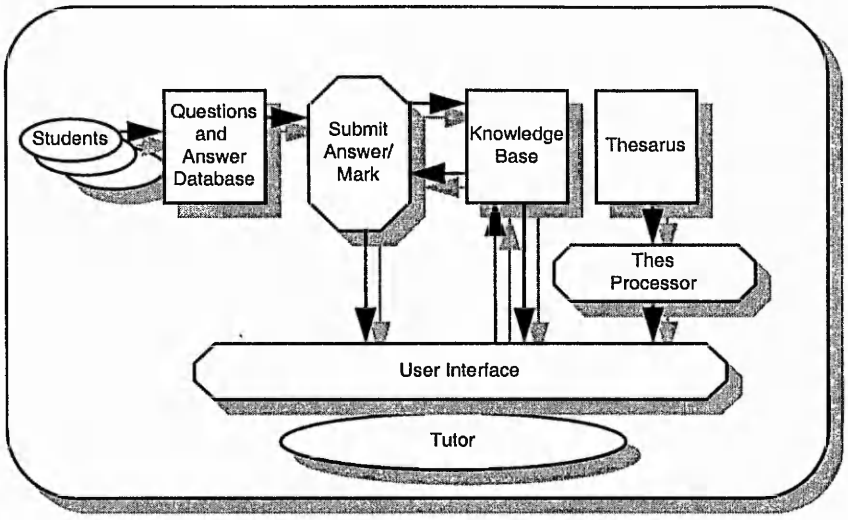


Figure 3: Implementation

The entire system was implemented on a 486 66Mhz DX2 PC. The system consists of four distinct modules, all of which are written in portable C++, except the user interface which requires version 3.1 of the Windows™ SDK:

5.1 - Core KR base: contains the functions needed to load, save and edit the knowledge base. Also functions that control the flow of activation between nodes. The module interfaces at three points: (a) where individual words are presented to the network to trigger activation from the evidence lists (b) where the network reports its internal state to the outside world (i.e. what facts/hypothesis have been triggered), (c) at the user interface which allows reconfiguration of the network.

5.2 - Submission module: essentially a text processing module which operates on the central database file presenting student replies to questions, word by word to the knowledge base.

5.3 - User Interface: is a graphical interface which allows viewing, editing and processing of student replies, thesaurus, and knowledge base.

5.4 - Thesaurus: provides processing functions to a text based version of the thesaurus. This provides access in one of two ways: (a) produces a list of synonyms for a words and list of links to other thesaurus entries (b) a more sophisticated interface looks up the entry number for every word for every student for a particular question. By printing the most frequent entry numbers we can isolate the word senses used most frequently by students for a particular question. This cuts down on the knowledge base production time considerably.

Published in: APPLICATIONS AND INNOVATION IN EXPERT SYSTEMS II (1994)
Eds Milne, R. and Montgomery, A. p319-332

5.5 - Sample K-Base

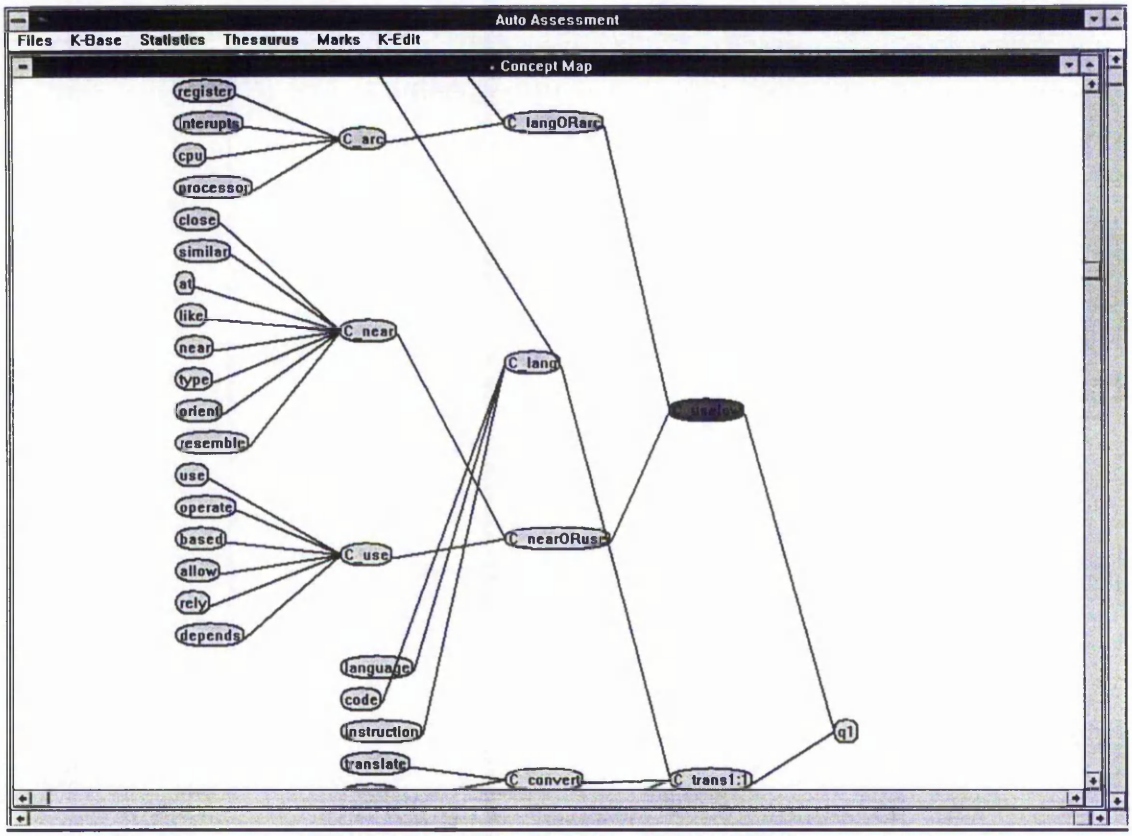


Figure 4: Sample Knowledge Base

6 - TESTING

Using the system as described two experiments were performed upon the system:

6.1 - Experiment 1 - Retrospective

1) The tutor set 20 questions on a general programming nature which in the tutor's opinion could be answered satisfactorily with a single sentence reply.
2) The questions were presented over the network to the students and the responses were concatenated into a central database file.
3) Replies were hand marked by the tutor and the marks recorded.
4) Using the tutors own intuition a knowledge base is constructed which attempts to capture the sense and the variety of language which can be used to express a valid reply. At this stage the interfaced Roget's thesaurus can be used to help anticipate the variety of language that could be used.
5) Each student reply is presented to the knowledge base one at a time and the automatic marker's decision is recorded.
6) For each student reply, the tutors mark is compared with the knowledge base decision. The two are correlated with each other to give a measure of similarity, and exceptions recorded.

Published in: APPLICATIONS AND INNOVATION IN EXPERT SYSTEMS II (1994)
Eds Milne, R. and Montgomery, A. p319-332

7) The knowledge base may then be edited by the tutor in order to better capture the exceptions and so increase the correlation rates. Return to step 5 until acceptable model of the human marker's performance achieved.

Using this technique a good model (average correlation 85%) could be produced for each question in less than 5 minutes. With slightly longer construction time correlation rates approaching 100% could be achieved in almost all cases. In order to approach these higher correlation rates, many of the *fringe* answers need to be incorporated into K-base. Interestingly many of the problems in incorporating the last few answers were attributable to spurious decisions on the part of the marker rather than genuine difficulties in extending the K-base.

The first experiment was essentially a test of the expressive sophistication of the knowledge architecture. That is, was it possible to construct a structure which could discriminate between right and wrong answers? The above results seem to confirm that this is possible. However, an obvious danger when constructing these knowledge bases is that we are arbitrarily discriminating between right and wrong answers, whereas we hope we are synthesising a K-base which truly embodies generalised elements of the problem domain. Further, this experiment does not prove the system is usable as in normal scenarios we would not have the answers pre-hand marked.

6.2 - Experiment 2 - Blind

There is no obvious method for determining the arbitrariness of a constructed knowledge base at face value. However to get some measure of the generality of the K-base we could apply constructed bases against new unseen data and compare correlation rates. This is what has been done here. Knowledge bases constructed in Experiment 1 were applied to new data (actually the responses of the following years students). This data set was then hand marked and coverage rates compared.

When the knowledge base created in the previous phase was applied to a blind set of data an average coverage rate of 65% was obtained. Considering the present limitations of the system this is considered very encouraging.

Published in: APPLICATIONS AND INNOVATION IN EXPERT SYSTEMS II (1994)
Eds Milne, R. and Montgomery, A. p319-332

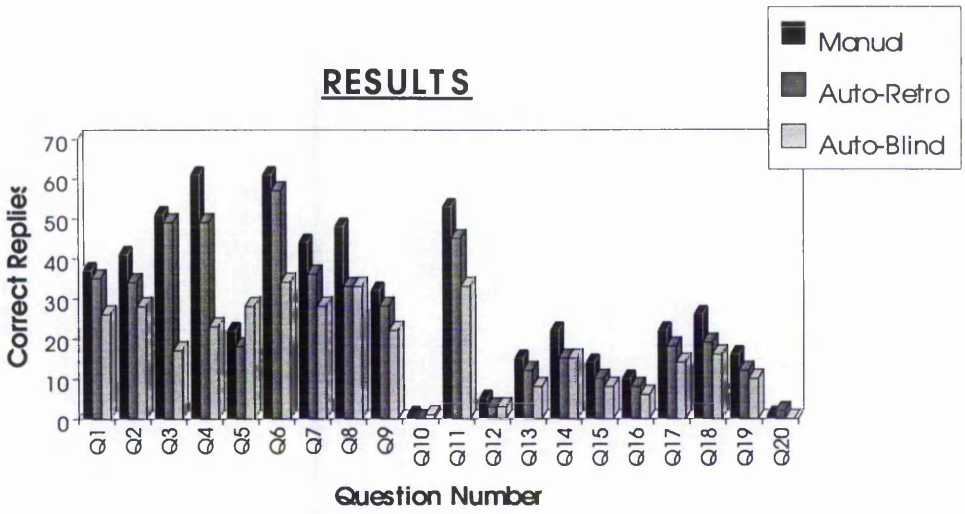


Figure 5: Graphed Results

6.3 - Comments

A knowledge construct which provides an approx. 85% coverage will typically be about 30 nodes in size. The time needed to mark is negligible: a single question answered by 64 students takes approx. 1 second to mark. At this stage the time required to both construct and mark a set of student sentences is slightly greater than that required by the tutor to do so by hand. However it anticipated that long term benefits (in terms of man hours) would accrue from the reuse of questions over different students. Probably the best mode of use would be the random selection of questions from a stable pre-configured bank of questions and knowledge bases.

The knowledge constructs generated necessarily embody information for both the KR and NLP processes in the one tree-like structure. A distinction, if one is to be made, is one of degree rather than absolute. Knowledge pertinent to language processing, to do with the status and category of a word, is held low down in the tree structure. Knowledge of a higher level which embodies facts etc. is held high up in the tree. As such it is expected that the low level nodes will be reusable across questions especially within the same domain. A level of cross domain reusability is provided through the thesaurus.

The automatic marker systematically marked fewer answers correct than the tutor. It is also noticeable that very rarely does the system mark an incorrect response correct. In a real application we can be reasonably certain (in our tests 98%) that a reply marked correct will indeed be correct. We can legitimately focus future efforts on capturing those unexpected correct replies.

When constructing knowledge bases our major problem is anticipating the variety of language which can be used to reply to a question. We have partially overcome that here by insisting on two runs of system. The first on a set of prototype information to help generate these anticipation's and a second true run. The next logical step in reducing the time required to generate the knowledge base and to improve coverage rates is to introduce secondary knowledge sources.

Published in: APPLICATIONS AND INNOVATION IN EXPERT SYSTEMS II (1994)
Eds Milne, R. and Montgomery, A. p319-332

APPENDIX F

7 - FURTHER DEVELOPMENT

The following have been identified as possible areas for further development:

7.1 - Running Parallel NLP Tasks Under the Same Architecture

Within the same underlying architecture it may be possible to implement many of the other natural language processing tasks, allowing techniques to be run in parallel: sharing the same information, and providing a mechanism for combing results. These other tasks (syntactic analysis in particular) may well prove essential to boost the correlation rates to the near 100% that would be necessary for the system to be adapted by staff and student alike.

- Syntactical analysis: a bottom up parse would be ideal for such an architecture. Further, contextual information is easily added (as in semantic processing) to augment its power. Variable node activation and mutually inhibitory connections would provide ambiguity resolution.
- Spelling correction and word recognition: by producing a similar architecture for the sub word level (i.e. dissolving words into hierarchically structured letter clusters) an information rich, context sensitive mechanism for word recovery may be provided. An algorithm for producing such sub-word structures has already been developed [15].
- Semantic analysis: more theoretical semantic methods such as componential analysis may also be implemented within the same architecture.

7.2 - Introducing Secondary Information Sources

In order to reduce the considerable overhead involved in generating the knowledge base and to improve the performance by utilising information about language variability secondary information sources need be integrated into the system. There are two immediate sources that have been identified:

Most immediately thesauri: we have already demonstrated two modes of interaction with a thesaurus. The next stage will be to integrate the thesaurus seamlessly into knowledge representation module. WordNet has also been identified as a possible supplier of such information. It has a richer hierarchical description of word meanings than conventional thesauri

Associated work within the department has produced extensive word-lexeme maps. This is precisely what is needed to produce the evidence lists required by the evidence nodes. This will reduce the Knowledge Base construction phase by another order of magnitude.

7.3 - Enhancing the architecture

Finally, the present system is limited in that node activation and message passing is a Boolean all or nothing event. Fuzzy data would be better processed by introducing some variability to the activation and hence the decision making processes. Also message passing may be enhanced by allowing messages to be passed down and across the network. This would

Published in: APPLICATIONS AND INNOVATION IN EXPERT SYSTEMS II (1994)
Eds Milne, R. and Montgomery, A. p319-332

give stronger ambiguity resolution via a winner take all strategy implemented by mutually inhibitory connections between sibling nodes.

8 - CONCLUSIONS

Automatic marking is a problem that serves as an ideal testbed for integrated knowledge representation and natural language processing theories. This is because (a) the domain is highly constrained, and (b) we have a large sample of possible sentences which when processed may then be correlated against a human decision to give a good statistical measure of the systems reliability.

Our solution to the problem has required the development of combined NLP and KR architecture, which has borrowed heavily from existing techniques. The major advantage of our chosen architecture is that not only does it provide a powerful representational scheme, but offers computational power along symbolic and/or connectionist processing paradigms. Any success achieved is attributable largely to our refusal to consider the problems of grammar, justified we believe by the constrained nature of expected responses. Our results at this stage are very encouraging and seem to confirm many of the assumptions we have made in our approach to the problem.

We have identified areas for future work which can be summarised briefly as (a) improve the reasoning mechanism of the architecture and (b) reduce the time and effort required to produce the knowledge bases. Our meter of success (i.e. correlation rates) is in place.

REFERENCES

[1] Gadzar, G and Mellish, C., *Natural Language Processing in Lisp*, Addison Wessley, 1989.
[2] Winograd T., *Understanding Natural Languages*, New York Academic, 1972.
[3] Anderson J. R., *The Architecture of Cognition*, Cambridge: Harvard University Press, 1983.
[4] Collins A.M. and Loftus E.F., *A Spreading Activation Theory of Semantic Processing*, Psychological Review, 85, 407-28, 1975.
[5] Gronlund N. E., *Measurement and Evaluation an Teaching*, New York: Macmillan, 1981.
[6] Chase C. I., *Measurement for Educational Evaluation*, Addison Wesley, 1978.
[7] Nelson C.H., *Measurement and Evaluation in the Classroom*, New York: Macmillan, 1970.
[8] Quillian R., *Semantic Memory*, in Semantic Information Processing, editor Minsky M., Cambridge: MIT, 1968.
[9] Minsky M., *A Framework for Representing Knowledge* in The Psychology of Computer Vision, editor Winson P., NewYork: McGraw Hill, 1975.

Published in: APPLICATIONS AND INNOVATION IN EXPERT SYSTEMS II (1994)
Eds Milne, R. and Montgomery, A. p319-332

APPENDIX F

[10] Schank R.C. and Colby K.M., *Computer Models of Thought and Language*, San Francisco: Freeman, 1973.
[11] Rumelhart D.E. and McClelland J., *Parallel Distributed Processing*, Cambridge: MIT, 1968.
[12] Gelerenter, D, *Programming Linguistics*, Cambridge: MIT, 1990.
[13] Sells, P., *Foundational Issues in Natural Language Processing*, Cambridge: MIT, 1991.
[14] Kummel, P., *Formalisation of Natural Languages*, Springer Verlag, 1979.
[15] Allott N., *A Clustering Algorithm for Natural Language Processing*, Internal Report - The Nottingham Trent University, 1994.

Published in: APPLICATIONS AND INNOVATION IN EXPERT SYSTEMS II (1994)
Eds Milne, R. and Montgomery, A. p319-332

AppendixG.

A Clustering Algorithm to Produce Context Rich Networks

Nick Allott
Consensus Software Ltd
email: nma@uk.ac.ntu.doc

Pete Halstead, Pauline Fazackerley
The Nottingham Trent University
Computing Department
Burton St
Nottingham
NG1 4BU

Abstract

This paper describes a formal symbolic algorithm for producing context rich networks by processing corpora of linear strings of discrete units, identifying from the statistical co-occurance of elements' probable compound units and the transition probabilities between the same. The networks produced this way are essentially connectionist in nature, in that the network comprises multiple inter-linked nodes each of which has a distinct activation level. However the individual nodes are symbolic in that each node represents a unique discrete phenomena within the problem domain. The produced networks are deeply structured embodying useful contextual information from the problem domain which is ideal for applying to recognition type tasks.

The outlined process closely parallels other statistical techniques that infer transition probabilities from statistical co-occurrence such as Markov models or N-Gram analysis [1], [2] and [3]. With these techniques, however, when an attempt is made to analyze context over a wide field (ie. large N) the size of the corpus necessary to provide a good esimate of the transition probability rises exponentially [4]. This problem is attributable to the need to specify N globally. The tree producing algorithm outlined here, by the use of some modifiable heuristics, estimates a unique value of N dynamically and locally. That is, we estimate the transition probabilities to and from a node if and only if our heuristic identifies the node as a suitable candidate.

Algorithm

A standard statistical procedure attempts to identify a relationship between N variables. The algorithm outlined here does this with a set of primitive nodes, however when a strong relationship is found between two or more nodes they

Published in: NEURAL NETWORKS AND THEIR APPLICATIONS (1997)
Eds. Taylor, G. p265-269.

are concatenated into a new node and the procedure will then look for relationships with this node also. The number of variables is therefore constantly growing.

Relationships between nodes are identified using a context history which records the context for each node. Conversely a node can be defined as something for which a context history is recorded. Take the example where we are analyzing written language, we will start with a set of 26 primitive nodes for which we will record the context history. When, for example, the letter *'h'* has been found in the context of *'t'* many times *'th'* become a node in itself. After some time we may identify *'e'* as a candidate link for creating the *'the'* node. The nature and depth of the produced networks may be modified through the use of two re-definable heuristics: the *linkage heuristic* and the *clustering heuristic*.

In outline, a new unconnected primitive node is instantiated for each primitive unit encountered in a string. Whenever this node is subsequently encountered within a string this node is made salient, and remains salient for a specified lifespan. That is, instance information is recorded temporarily for it on a list. The salient list is monitored at all times, so should two nodes be discovered which satisfy the *linkage heuristic* the appropriate context history is immediately attached to the concerned nodes.

When the link/context history combining two (or more) nodes exceeds a value specified by the *clustering heuristic* a new compound unit is instantiated which describes the conjunction of the children nodes.

Certain metrics are attached to the various entities in the tree. It is with respect to these metrics that the *clustering* and *linkage heuristic* are defined. Of significance is the measure of frequency (attached to nodes, links and global measures). An absolute measure of frequency would be inadequate for inter-node comparisons as new nodes are being created all the time and the new nodes are unaware of how often they occurred before they were created. More suitable would be a measure of acquisition velocity, that is to take an estimate of the differential of frequency over time. It was found that the regular resetting of all frequency values served as a satisfactory approximation to this.

The algorithm as discussed may be defined in pseudo code as follows:

```
LOOP (for all strings)
     {
     LOOP (for all units)
          {
          IF (node not previously seen)
               {
               DO Instantiate_Node
               }
          ELSE (node already seen)
               {
               DO Activate_This_Node
               DO Pass_Activation_To_Parents
```

Published in: NEURAL NETWORKS AND THEIR APPLICATIONS (1997)
Eds. Taylor, G. p265-269.

APPENDIX G

```
                DO Make_Node_Salient
                LOOP (all salient nodes)
                     {
                     IF (Nodes match LINKAGE HEURISTIC)
                          {
                          DO Add_Link_To_History
                          IF (Link > THRESHOLD HEURISTIC)
                               {
                               DO Make_Nodes_Into_Compound
                               DO Link_Children_To_Parent
                               }
                          }
                     }
           DO Delete_Salient_Nodes
           }
}
```

The nature of the parent or compound nodes needs some clarification. Firstly, note that a parent node receives activation from all its child nodes. The parent itself though, will not become active until it receives activation from all child nodes such that the combination satisfies the linkage heuristic. The linkage heuristic is therfore defining the nature of the network i.e. the conditions that two nodes must satisfy for the two to be considered a compound node.

Secondly we need to make a distinction between the type and token instance of a compound node. This is because unlike primitive nodes a compound can exist in overlapping positions within a string. Take the example of '*banana*': the cluster '*ana*' exists twice in non-distinct positions.

Saliency

The distinction between the type and token instance of a node relates to the principle of saliency mentioned earlier, this probably needs a few words of explanation.

Within a standard neural net architecture it is difficult to pass on the postional information of a node explicitly within the activation it passes to its linked nodes This is because an individual node is not computationally powerful enough to discriminate its input and effect the functional changes that a new node position would imply. It would be easier to pass on the postional information implicitly by only letting nodes in position 1 pass on their activation to nodes in position 2 etc., however the number of nodes necessary to encode the input sting and model the relationship between clusters in their various positions would rise exponentially with the string length.

The network discussed in the previous section is in essence only recording the relationship between nodes of a particular type. However in order to both build and use the net we need to be able to identify instances of a node within an input sting and be able to propagate this instance information through the

Published in: NEURAL NETWORKS AND THEIR APPLICATIONS (1997)
Eds. Taylor, G. p265-269.

network. The principle of saliency is a mechanism by which this information may be stored and passed throughout the network. It is in many ways a type of working memory. The type of instance information that is stored with the node identifier on the salient list is dependent upon what is required by the linkage heuristic. In most simple cases this will simply be the position of the node within the string.

Modifying the Network

As mentioned the nature and quality of the network is modifiable through two heuristics, to illustrate this consider the following examples. Firstly the *linkage heuristic* which effects the quality of the network. In the simplest cases the metrics passed to salient nodes on instantiation will be node identifier and position found in the string. From this we could define linkage as simple left or right adjacency giving immediate left or right context. Alternatively we could extend this to give context within a pre-specified window i.e. found within 5 units.

Secondly the *clustering heuristic* which determines the depth. The simplest definition is a measure of the absolute frequency for a node divided by the total number of primitive units encountered. The next probable extension to this would be to modify this value with respect of the length of the compound, on the legitimate assumption that smaller compounds are likely to occur more frequently. This will stop trees becoming bottom heavy.

The second level of sophistication is to introduce a measure of a node's suitability for a task into the network modification procedure. When the information necessary to do this is available at node creation this measure may be readily incorporated into the *linkage heuristic*. However, this is not always practical. Take the recognition task: a good measure of a node's usefulness is whether the distribution of transition probabilities to new nodes is flat or not. A flat distribution can be used to no predictive effect. This information is available only after the node has existed for some time. In such cases we should consider tree generation as a two stage process. The first to generate a bushy, deep tree on pure statistical grounds, a second to prune the tree on functional grounds.

Formal Specification of Data Structure and Heuristics

Where N the nodelist is a set of s nodes.

$$N = \{n_1, n_2, n_3, \ldots\ldots, n_s\}$$

Here each component n is a composite of the with the following attributes:

$$n = \{WordID\ ,\ Frequency\ ,\ Children\ ,\ Parents\ ,\ Linkage\ \}$$

Published in: NEURAL NETWORKS AND THEIR APPLICATIONS (1997)
Eds. Taylor, G. p265-269.

APPENDIX G

WordID is a string to identify the node, *Frequency* is an integer in which we record the number of times the node has been found. *Children*, *Parents* and *Linkage* are all lists and defined as follows:

$$Children_x = \{c_{1x}, c_{2x}, c_{3x}, \ldots, c_{ix}\}$$

where each member is of type *c* comprising the single attribute:

$$c = \langle WordID \rangle$$

Similarly

$$Parents = \{p_{1x}, p_{2x}, p_{3x}, \ldots, p_{ix}\}$$

and:

$$p = \langle WordID \rangle$$

Finally

$$Linkage_x = \{l_{1x}, l_{2x}, l_{3x}, \ldots, l_{wx}\}$$

where *l* is a composite type with two attributes

$$l = \langle WordID\ ,\ Frequency \rangle$$

If we use the syntax where a function of the same name as an attribute performed on an appropriate composite extracts the value of that attribute. It follows that:

$$Freq(n_x) = \sum_1^s l_{ix}$$

In other words the frequency of a node is equal to the sum of the frequency of the distinct contexts it has been found in. A useful variable is the total number of primitive nodes encountered

$$Global = \sum_1^s freqprims(i)$$

where

$$freqprims(i) = \begin{cases} [illegible]eq(n_i), & |word(n_i)| = 1 \\ & |word(n_i)| \neq 1 \end{cases}$$

and

$$|word(n_i)| = \text{length of string}$$

The input string we may interpret mathematically as a set of ordered pairs.

Published in: NEURAL NETWORKS AND THEIR APPLICATIONS (1997)
Eds. Taylor, G. p265-269.

APPENDIX G

$$String = \{k_1, k_2, k_3, k_z\}$$

Each element is a composite with the first attribute the *letterID* the second is the *position* in the string.

Given this the linkage heuristic may be defined as a Boolean returning function. The simple right adjacency heuristic then becomes

$$Link(x,y) = \begin{cases} position(k_x) = position(k_y) + 1 \\ position(k_x) \neq position(k_y) + 1 \end{cases}$$

And left adjacency

$$Link(x,y) = \begin{cases} position(k_x) = position(k_y) - 1 \\ position(k_x) \neq position(k_y) - 1 \end{cases}$$

A windowed left right adjacency may be

$$Link(x,y) = \begin{cases} |position(k_x) - position(k_y)| < 3 \\ |position(k_x) - position(k_y)| \geq 3 \end{cases}$$

Clustering heuristics are also Boolean returning functions. Below we list three simple useful ones.

An heuristic relying on absolute frequency

$$Cluster(x,y) = \begin{cases} \dfrac{freq(l_{xy})}{Global} \geq 0.01 \\ \dfrac{freq(l_{xy})}{Global} < 0.01 \end{cases}$$

An heuristic which is dependent upon the frequency of the parent node:

$$Cluster(x,y) = \begin{cases} \dfrac{freq(l_{xy})}{freq(n_x)} \geq 0.01 \\ \dfrac{freq(l_{xy})}{freq(n_x)} < 0.01 \end{cases}$$

Or perhaps an heuristic which makes an adjustment for the proposed length of the compound node (on the assumption that longer clusters are less frequent):

$$Cluster(x,y) = \begin{cases} \dfrac{freq(l_{xy})}{freq(n_x)} \times (|word(n_x)| + |word(n_y)|)^p \geq 0.01 \\ \dfrac{freq(l_{xy})}{freq(n_x)} \times (|word(n_x)| + |word(n_y)|)^p < 0.01 \end{cases}$$

Published in: NEURAL NETWORKS AND THEIR APPLICATIONS (1997)
Eds. Taylor, G. p265-269.

where

p = constant to offset decreasing frequency

Application of Produced Trees

As discussed it was originally envisaged that these trees be applied to recognition type tasks. In their simplest they can be shown to produce identical information to that produced by N-Gram analysis or the transition matrix of a Markov model. However, we do not have to specify our grain of analysis (bigram - trigram etc.) before processing takes place nor must the grain be unique throughout the analysis. If the heuristics are set up correctly the algorithm dynamically and locally determines these for each node. We can therefore use the trees in the same way as we would with N-gram or Markov i.e. produce estimates of word probability from supplied evidence, as a multiple of transition probabilities between primitives. But using the context tree we may augment these estimates by describing our units in higher level terms and using transition probabilities between these to modify our original figure.

A second tangential application of the algorithm is identifying the ideal grain of analysis for a problem. For both formal algorithms which must specify processes between sets of primitive units and connectionist networks which must specify a set of input units, a process which can produce higher level more compact descriptions of the data could prove very useful. Further by defining the qualities we require of our units in the linkage heuristic we can ensure the nodes are appropriate to our needs. Along a similar vain, the networks can be used to identify repeating features in the problem domain which is useful for compression algorithms.

Most of the explanations so far have referenced the problem of text recognition and consequently refer to letters as primitive nodes. Note it is equally feasible to operate on phonemes, syntax or possibly on words or semantic category.

Further Work

The above describes a method by which a symbolic algorithm queries the essentially connectionist tree to produce probability estimates for a recognition task. What is seen as a far more attractive solution to this problem is to use the network in a more native connectionist manner, whereby on the presentation of activation to the primitive nodes, the tree, through interactive activation and competition, resolves the best fitting highest level nodes. This has the advantage of producing a far more sophisticated method of implementing feedback and disambiguation.

Published in: NEURAL NETWORKS AND THEIR APPLICATIONS (1997)
Eds. Taylor, G. p265-269.

In such an implementation the interconnected network becomes a dynamic system with N stable states where N is the number of words that can be recognized. The primitive node could be activated wholly or partially according to the certainty with which they exist in the input string. The internal dynamics of the system can be adjusted to inhibit multiple states becoming active. The constraints within the problem domain can be absolute or probabilistic. For example in written text there is an absolute constraint that 'u' must follow 'q', but a probabilistic constraint that 'h' frequently follows 't'. However constraints also exist between the higher level elements (e.g. 'e' frequently follows 'qu'). The network produced provides both the clusters and the contextual information from which we may derive likely internode weights. Specifically the type of weights for which estimates must be provided are: excitatory links between child and parent nodes, feedback links between parent and child and inhibitory links between competing sibling nodes.

The second extension is to enhance the algorithm to identify *abstracted* as well as *compound* nodes (e.g. cluster letters into vowels and consonants or by phonetic characteristics etc.). To be able to automatically cluster units from simple statistical distribution would add important flexibility to our higher level description, not to mention improve the estimates of transition probabilities.

References:

[1] Jelinek F, Mercer R and Bahl (1983), *'Continuous Speech Recognition: Statistical Methods'*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol PAM-5.

[2] Brown P, Lee H and Spohrer, *'Bayesian Adaptation in Speech Recognition'*, Proceeding of the ICCASP, Boston, 761-764.

[3] Riseman E and Hanson A (1974), *'A Contextual Postprocessing System for Error Correction Using Binary N-Grams'*, IEEE Transactions on Computers, Vol C-23, 490-493.

[4] Keenan F (1992), *'Large Vocabulary Syntactic Analysis for Text Recognition'*, PhD Thesis, The Nottingham Trent University.

Published in: NEURAL NETWORKS AND THEIR APPLICATIONS (1997)
Eds. Taylor, G. p265-269.

AppendixH.

APPENDIX H

Connectionism and Symbolism in Symbiosis

N. Allott, P. Fazackerley and P. Halstead

Computing Department,
The Nottingham Trent University,
Burton St,
Nottingham,
NG1 4BU.
England
email: nma@uk.ac.ntu.doc

Abstract

In this paper we examine a previously published algorithm which addresses the problem of network growth by implementing a clustering algorithm to operate on time dependant data. The computational constraints of the problem forced the development of an architecture, which in retrospect can be analysed in terms of a computational and symbolic module operating symbiotically. Here we attempt to identify the computational constraints that necessitate the use of this architecture, and any further merits it has. Further we analyse the nature of the interaction between the two modules and highlight the manner in which the behaviour the symbiotic modules correlates with what is known of human problem solving behaviour.

1. Introduction

This paper both develops and outlines the computational merits of a symbiotic symbolic and connectionist network used within previously published clustering algorithm [1]. Within this algorithm a connectionist network was used to embody the relationship between discrete observable elements, for example letters of the alphabet. In the simplest case the relationship modelled is the relative statistical distribution, or context, of the letters. As such the network can be shown to be very similar to N-gram analysis [4][5][6] or the transition network of a Markov model [3]. However it is superior in that the scope of analysis to be considered to does not have to be specified globally, but is dynamically and locally determined for each node.

An algorithm was required to produce these [1]connectionist *trees* from empirical data. It was the original intention that the algorithm be developed within the context of the connectionist paradigm. By this we mean capable of being implemented in a parallel manner such that, the functions used to compute the working parameters for each node have access only to those items to which the node is architecturally linked (such as the back propagation algorithm or simple Hebbian learning). However the fact that (a) the algorithm attempts to *grow* the network (b) time dependant data was being handled, made this design goal difficult to satisfy. In the next section we attempt to formalise the source of the difficulty.

2. Formal Definition of Problem

The network can be characterised as an n-tuple:
$\langle P, N, L, a, \Sigma \rangle$. Where
P is the set of primitive nodes,
N is the set of all nodes, initially N=P
L is the set of links between nodes, initially L=∅, and each element of L is a 3-tuple $\langle p, c, s \rangle$, parent, child, strength, where s=1.
a is the activation function, $a: N, t, \Sigma^* \rightarrow \{0,1\}$
Σ is the set of possible primitve evidence

Derived from this we have
Σ* the set of sentences possible from Σ.
P(N) the power set of all nodes
A the set of abstract nodes, defined A=N∩P'
c is the set of children of a node, $c: A \rightarrow P(N)$ and can be defined in terms of L and N.

To simplify the problem, in the initial case the strength of all links is assumed to be 1, and the activation function is Boolean returning {0,1}.

If S∈Σ* then S[1]∈ Σ and is the first element of S. It follows for simple sequence analysis (such as the text string discussed above) where there is a 1:1 mapping between P and Σ, the activation function for a node n, where n∈P

[1] We would like to acknowledge the original flash of insight from Nick Porter, which lead to the analysis presented in this paper.

Published in: ARTIFICIAL NEURAL NETS AND GENETIC ALGORITHMS (1997)
Eds. Smith, G. et al Springer, New York, p570-575

APPENDIX H

$$a(n,t,S) = \begin{cases} 1, & S[t] = n \\ 0, & S[t] \neq n \end{cases}$$

And for node n where n∈A, the activation is some function f() of the activation of the children of n, $a(n,t,S) = f(c(n),S,t)$.

To illustrate, take the simple problem of clustering with the node immediately adjacent on the right. If c(n)[x] is the x^{th} child of node n, the activation function for a parent with two children becomes: $a(n,t,S) = a(c(n)[1],t,S) \wedge a(c(n)[2],t+1,S)$

If it is our aim to grow the network, the process of growth is to identify a new node x such that N=N∪x and to add links such that $L = L \cup \bigcup_{i \in c(x)} \{\langle x,i,1 \rangle\}$. It is hoped that each new identified node x should capture some abstract feature of the input domain thus giving the network greater depth of perception.

When it comes to implementing the above procedure on a connectionist network there are essentially two problems to be solved:

2.1. Type token distinction

In a network where there is localist representation (one node represents one feature in the problem domain) it is conceivable that a single feature occurs twice in the input pattern. This feature could occur in distinct positions within the input sting, in overlapping positions, or the most difficult: in recursively embedded positions. (The string "*ana*" as it occurs in the input string "*banana*" is an example of an overlapped position) The problem is how does a single node within the network simultaneously take on two distinct activations to reflect the two instances of the feature within the input string?

2.2. New nodes, new links

Most mainstream connectionist learning algorithms have a predefined set of nodes and a predefined set of links (often all nodes fully interconnected). The learning algorithm then adjusts the strength of the existing links to reflect the relationship between the nodes. This algorithm addresses the problem of network growth, all nodes are therefore initially unconnected and there are no links. The aim is to identify new nodes and new links that represent relationships between the existing nodes. However if we are to implement the algorithm to identify these relationships in a truly parallel, connectionist manner we have a problem, each node has no direct architectural link to any other node from which parameters could be computed which could lead to the instantiatation of a new node.

2.3. Solution

To solve these both problems a symbolic processing module was added to the connectionist network. Within this symbolic layer nodes are instantiated when they become active within the network. It therefore identifies and records *salient* nodes. Each of these instances can be regarded as a token of the particular node type. Further, the symbolic layer provides a local area where the parameters of an instanced object may be compared against one another, in order to identify relationships.

3. Architecture

The architecture described in summary consists of a symbiotic connectionist and symbolic process. The connectionist process both provides a permanent store for the associations found between units and the perceptual framework for the overall process (i.e. identified units within data). The symbolic process provides a type of working memory for our network. Let us consider each of these processes in greater detail.

3.1. Symbolic Process

A symbolic process by definition operates on symbols. The question of what do these symbols represent is usually defined prior to the instigation of the process. However in the outlined architecture we circumvent the need to do this. We define only the lowest levels symbols - those at atomic level. The interaction between the connectionist and symbolic processes serves to identify new symbols that are hopefully more appropriate for the task in hand.

It is part of the function of the symbolic process to identify the relationship between active units. In the example outlined above this association is simple adjacency. There is no reason why this association may be considerably more complex than this.

3.2. Connectionist Process

The connectionist layer represents the relationship between identified clusters within the problem domain. This could well map out the hierarchical description of the problem domain, or by the use of excitatory and inhibitory links describe a causal link between nodes. However in the outlined design the exact nature of the relationship is embodied in the symbolic layer. By extracting this information from the network itself we allow for specialisation of the network.

4. Advantages

4.1. Type-Token Distinction

To maintain a type/token distinction between nodes within a connectionist architecture is a far from trivial task. Further we must ask ourselves the question when is it necessary to make the distinction. Using the letters example from above we need to distinguish between

Published in: ARTIFICIAL NEURAL NETS AND GENETIC ALGORITHMS (1997)
Eds. Smith, G. et al Springer, New York, p570-575

two instances of the node "ana" only within working memory i.e. within the symbolic phase. Should we later need to create a more permanent distinction of the individual tokens of the "ana" type we must ask ourselves what is going the be the discriminating aspect of the tokens. Necessarily this will have to be context, and a cluster contextually discriminated is a longer cluster, which would naturally be incorporated into the connectionist layer.

We have therefore made a distinction between node tokens which only need to be discriminated in the learning phase of the network and those which are to become part of the network structure itself. Symbolic memory, which is needed to make the token distinction, is therefore needed in the learning phase only.

4.2. Functional normalisation

One of the prime differences between a symbolic and a connectionist system is the clarity of the distinction between data and process. Within a symbolic process the distinction is clear cut whilst one of the key reasons for the flexibility of the connectionist model is that there may be no such distinction. Data may be represented locally (the activation of one node represents one atom of data) or data may represented in a distributed manner (there is no one to one relationship between the activation of any set of nodes and a piece of data). A process is modelled by the entire spread of activation through a set of nodes.

Take the situation where we wish to model complex functional relationship between many pieces of data (this is the function f() in the above formalisation). That functional relationship must be modelled itself by a set of nodes and its interconnecting links. If this is to hold between several pieces of data the nodes necessary to model the functional relationship must be repeated throughout the network many times, an obvious redundancy. In the design outlined above the functional relationship desired can be modelled outside of the connectionist network, eliminating this redundancy. This analogous to the process of data normalisation as used in databases, for this reason we call the process functional normalisation.

4.3. Specialisation of Networks

To follow on from this point, by encapsulating the relationship that holds between items in a central place (in this case the symbolic layer) we allow for greater specialisation of networks. Where the connectionist layer models the hierarchical structure of, or the causal relationships that are to hold between, many items, and the symbolic layer models the relationship itself, we can use essentially the same type of network to model all types of things. For example, referring back to the clustering example, it is possible to use essentially the same network to represent left adjacency and right adjacency of clusters by just changing the function f() the symbolic process.

4.4. New data

If we are producing a network which is to reflect the structure of items between which a particular relationship holds, by maintaining this relationship outside of the network we have the means by which to test new, unseen and hence unrecorded items against each other.

5. Cognitive Correlations

There are certain aspects of the outlined model which correlate well with the what has been observed of our own human problem solving behaviour. We comment on the similarities in idle speculation only and do not consider the similarities to constitute any form of proof of the validity of the proposed model.

5.1. Memory Types

Psychologists have for some time maintained a distinction between Short Term Memory (STM) and Long Term Memory (LTM) [7]. The three major distinctions being: duration, capacity and coding [8]. Within the two modules discussed here there are similar distinction to be made. Items instantiated within the symbolic layer have a short life span (the length of the current input pattern) whilst the connectionist links are permanent. The coding of the nodes within the connectionist layer is contextual; all that is known about that node is inherent within the links attached to it. The tokens used within the symbolic layer are arbitrary representations. However, the capacity of the symbolic layer does not seem to be limited in the same way that STM seems to be. This could be due to their differing implementations (see later.)

Note, also, in overall function the symbolic layer is similar to Klatzky's [9] description of STM as a "mental workbench".

5.2. Symbolic Whilst Learning

It seems a feature of learning that, when presented with learning a new task or skill, the processing tends to be symbolic and procedural in the initial phases, less so in the latter phases. For, example consider learning to type, to drive a car or learning to read. Again we see similarities with the symbiotic design. Consider the clustering problem discussed above. In the initial phases the identification of new features is performed entirely within the symbolic layer. Once identified the apparatus necessary to *perceive* the feature is incorporated within the nodes and links of the connectionist layer, so this is where the processing now takes place.

Published in: ARTIFICIAL NEURAL NETS AND GENETIC ALGORITHMS (1997)
Eds. Smith, G. et al Springer, New York, p570-575

5.3. Speed and Implementation

The architectural implementation of the modern computer is in most cases a serial, symbolic, Von Neumann process. Whereas there is no doubt that the brain is made up from nerves which can operate in parallel.

People seem to have two modes of operation, to quote Norman [10] "one rapid, efficient, subconscious, the other slow, serial and conscious". The proposed architecture also seem to perform in two modes: a fast serial symbolic process, a slow connectionist process. A distinction between processes is preserved, although the performance ratios disagree with one another. It seems possible, however, that the juxtaposition of performance ratios is attributable to the differing implementations. Certainly the connectionist network discussed above is in reality a serial emulation of a connectionist process and so it would be reasonable to expect a performance drop.

The issue of emulating a serial process within a connectionist architecture is more complex. Clark [2] has speculated for some time that "the human mind might effectively simulate a serial, symbol processing Von Neumann architecture." And it is interesting to note that in his book [11] he notes two shortfalls of connectionist networks in explaining human capacity:

1) to be able to perform serial reasoning in which the ordering of operations is vital.
2) to be able to utilise a control structure in order specify salient micro features for inductive generalisation.

As this is precisely the type of functionality being satisfied here by the symbolic layer.

Rumelhart and Smolensky [12] have also pondered on the human capacity to engage in conscious, symbolic reasoning, and Touretsky [13] has contributed to the debate with his proof that neural nets can be used as Turing Machines. All we can do here is to leave the open ended question: would a neural implementation of a symbolic process have a limited capacity and be relatively slow to process? If so, this would fall in line with the arguments presented above.

6. CONCLUSIONS

In the above we outline a symbiotic architecture encapsulating both a symbolic and connectionist process. The architecture was conceived initially as the solution to a clustering problem which could not easily be solved using solely connectionist techniques. However the combined model has several interesting architectural properties some of which seem to reflect observations of the brain's own problem solving behaviour.

Within the model the symbolic layer provides a form of working memory for the network as it learns from new data. The connectionist layer in return provides the perceptual framework for the symbolic layer: supplying the symbols upon which the symbolic process is to operate. The two are symbiotic in that they modify each other's data. The model as a whole is providing a learning schema where data is initially analysed symbolically, and the trace of this data later imprints itself onto the connectionist network.

7. REFERENCES

[1] Allott,N, Fazackerley, P, Halstead P. 'A Clustering Algorithm for Producing Context Rich Networks.' In J. Taylor ed., Neural Networks and their Applications, pp220-29, Chichester: Wiley, 1995.

[2] Clark,A. 'Connectionism and Cognitive Science.' In J. Hallam and C. Mellish eds., Advances in Artificial Intelligence, pp3-15, Chichester: Wiley, 1987

[3] Jelinek F, Mercer R and Bahl 'Continuous Speech Recognition: Statistical Methods', IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol PAM-5, (1983).

[4] Brown P, Lee H and Spohrer, 'Bayesian Adaptation in Speech Recognition', Proceeding of the ICCASP, Boston, 761-764.

[5] Riseman E and Hanson A. 'A Contextual Postprocessing System for Error Correction Using Binary N-Grams', IEEE Transactions on Computers, Vol C-23, 490-493, 1983.

[6] Keenan F. 'Large Vocabulary Syntactic Analysis for Text Recognition', PhD Thesis, The Nottingham Trent University, 1992.

[7] Atkinson, R.C., and Shiffrin, R.M. 'Human Memory: A Proposed System and its Control Processes' In Human Memory: Basic Processes, Bower, G.H.(ed) New York : Acedemic Press, 1977.

[8] Wicklgren, W.A. 'The Long and Short of Memory'. Psychological Bulletin, 80, 425-438, 1991.

[9] Klatzky, R.L. 'Human Memory: Structure and Processes. Freeman, San Francisco, 1980.

[10]Norman, D. 'Reflections on Cognition and Parallel Distributed Processing' In McClelland and Rumelhart (ed) Parallel Distributed Processing, vol2, p110-146. MIT Press, Cambridge, 1986.

[11]Clark, A. 'Microcognition'. MIT Press, Cambridge, 1990.

[12]Rumelhart, D., Smolensky, P. McClelland, J & Hinton, G. ' Schemata and Squential Though tProcesses in PDP Models' Parallel Distributed Processing, vol2, p110-146. MIT Press, Cambridge, 1986.

[13]Tourtsky, D.S. 'BoltzCONS: Dynamic symbol Structures in a Connectionist Network' Artificial Intelligence vol46, p5-46, 1990.

Published in: ARTIFICIAL NEURAL NETS AND GENETIC ALGORITHMS (1997)
Eds. Smith, G. et al Springer, New York, p570-575

AppendixI.

APPENDIX I

Connectionist Pattern Matching for Information Extraction

N. Allott, M. Allott[†], P. Fazackerley and P. Halstead

Computing Department,
The Nottingham Trent University,
Burton St,
Nottingham,
NG1 4BU.
England
email: nma@uk.ac.ntu.doc
[†]Consensus Software Ltd.

Abstract

A connectionist schema is presented for performing robust matches on natural language which can be applied to information extraction tasks. Out of primitive connectionist components (nodes and links) macro nodes are created which have representational and computational properties which closely reflect the specific requirements of natural language. In particular the problem of bootstrapped contextual disambiguation is raised and a network structure, built from macro nodes is presented which addresses this problem. Such networks represent a theoretical advance upon the conventional Boolean search strings typically used for text searching.

1. INTRODUCTION

One of the advantages of storing text on computer is that documents may be searched extremely quickly. The typical search interface, for example those used on an application's help and web search engines, comprises a search string, which is usually a list of relevant words. Optionally these words may be combined with the Boolean connectives AND and OR in order to further refine the search criteria. Although largely successful this approach is limited in at least two senses:

- Natural language is incredibly diverse; the same thing may be said many different ways.
- The searches performed are exact string matches which take no account of polysemy or homonymy [1] (in other words the opposite of above: one thing may mean many different things).

We present in this paper a connectionist formalism which specifically models those aspects of the sense of natural language which can prove difficult to capture within conventional logical expressions. The formalism was originally developed to tackle the problem of automated assessment [2][3][4][5] which attempts to discern from a database of single sentence student responses the sentences that most closely match the tutor defined criteria. However the core problems being tackled: (a) the robust definition of semantic criteria and (b) a process of matching natural language sentences against such criteria, are highly relevant to the more general problem of text searching and information extraction.

2. PROBLEMS OF NATURAL LANGAUGE MATCHING

Natural language is an evolved form of communication which has a complex composition that strongly resists reductive analysis. There are a number of features of natural language that can be identified which pose particular problems for the information extractor.

Synonymity: words are equivalent in meaning and therefore interchangeable in usage. Note few words are entirely interchangeable with another and depending upon context synonymity is largely a matter of degree.
Hyponymy: one word represents a subset of another, for example car is a hyponym of vehicle. Words are therefore interchangeable in one direction only.
Idioms: the meaning of two or more words combined is different to that implied by its constituent parts. This is a perfect example of non-reductivity.
Polysemy/Homonymy: where a single word has more than one possible meaning, thus giving rise to ambiguity.
Anaphora: the use of pronouns etc. which can be used as shorthand for previously encountered concepts within the text.
Metaphor: a flexible use of language whereby words or groups of words can be used outside of their normal context, and take on a meaning which implies some but not all of its defining characteristics.
Inference: the meaning of a phrase is not only the literal interpretation of the word definitions but all consequences that naturally follow from it. For example, *water is heavier than air* implies *air is lighter than water.*

Published in: NATURAL LANGUAGE PROCESSING FOR BUSINESS USE (1997)
Eds. Wilks, Y. et al Unicom, p256-262.

APPENDIX I

These are all well documented features of natural language and their relationship to the task of natural language processing is discussed in depth in [6], amongst other sources.

The problem of ambiguity (resolving the meaning of an individual word with two or more distinct meanings) however poses a problem of exceptional complexity and we shall briefly outline the extent of the problem below. Ambiguity is not an isolated problem that exists for a few words in every sentence. If we are to take a dictionary as the authority on the number of words senses, it is a fact that most words have more than one meaning. (Note there is considerable variation between dictionaries.)

To demonstrate the enormity of the problem look at the prototypically simple sentence.

The cat sat on the mat.

Using the relatively small Concise Oxford Dictionary as a the source for identifying distinct senses of a word, the following table is produced:

Word	THE	CAT	SAT	ON	THE	MAT
Part of Speech	*adjective adverb*	*noun*	*verb*	*preposition adverb*	*adjective adverb*	*noun verb*
Meanings	definite article	small furry quadruped	supported by buttocks	supported by or covering	definite article	course fabric or floor covering
		spiteful or malicious women	rested with hind legs bent	close to		small rug
		person(Jazz fan)	pose for portrait	concerning		piece of material laid on table
		whip (cat of nine tails)	to be an MP for a constituency	added to		to bring into a thickly tangled state
		any wild feline animal	remain on nest to hatch eggs	forward		
		abv for caterpillar vehicle	be member of committee	movement of operation being shown or performed		
		abv for catalytic converter	to be in session			
			to cause to be seated			
			remain in the same position			
Number of Meanings	**1**	**7**	**9**	**6**	**1**	**4**

Figure 2-1 Ambiguity in "the cat sat on the mat"

Assuming meaning independence of individual word this produces a possible (1x7x9x6x1x4=) 1512 different possible interpretations of this sentence and this was using a very small primitive dictionary. If a larger dictionary is used and even greater number of permutations are generated.

If we have no *a priori* reason for preferring one interpretation over another.

> *"The spiteful lady posed for a portrait close to the table covering."*

Is as valid an interpretation as.

> *"The small furry quadruped rested with its hind legs bent supported by the small rug"*

This presents a problem: if context is them means by which ambiguity is to be resolved (as is commonly thought), what do we do when this context is itself ambiguous?

A solution to this problem is particularly relevant to text searching. As any regular user of text searchers will testify, most search *mismatches* are attributable to incorrect senses in the target documents incorrectly matching against the search criteria.

Within this paper we shall present a connectionist representation which closely parallels Rumelhart and

Published in: NATURAL LANGUAGE PROCESSING FOR BUSINESS USE (1997)
Eds. Wilks, Y. et al Unicom, p256-262.

APPENDIX I

McClelland's [7] multiple constraint satisfaction models. The schema, then, by virtue of its structure allows for a bootstrapped mechanism of sense disambiguation which address the above problem.

3. REPRESENTATION SCHEMA

3.1.1. Node

The architecture proposed consists of a set of nodes. Each of these nodes has a unique identifier. Nodes can be further subdivided into nodes that receive input from the external world (evidence nodes), those that receive input only from each other (hidden nodes) and output nodes, thus following the conventional neural network paradigm.

3.1.2. Link

Any pair of nodes may be bound together with a link, which reflects the strength of relationship between two nodes. This link is the means by which nodes pass activation between one another and is unidirectional in nature. Therefore with respect to a link there is a child node (the node from which activation is passed) and a parent node (the node that receives activation.) A particular node can therefore be both a child and a parent node, but with respect to two distinct links.

3.1.3. Activation

Every node has an activation value, this represents the extent to which this feature exists within the input pattern, a value between 0 and 1. Evidence nodes assume an activation of 1 if their feature is identified within the input pattern. If a node does becomes active, activation is propagated through its output links.

3.1.4. Threshold

Each node also posses a threshold value. This is the value that the sum of the activation, which is attributable to the input links, must exceed if a node is to be activated.

3.1.5. Evidence Nodes

As mentioned above nodes are primarily divided into evidence nodes and hidden nodes. An evidence node receives input from somewhere other than the network itself. The mechanism by which a particular evidence node is activated is largely unimportant. It could be an external filtering mechanism which activates the nodes manually or the perceptual intelligence may be programmed into the node itself

Within the system to be used here, evidential nodes are used to represent the low word level knowledge. Each has associated with it a list of words (evidence list) whose presence in the sentence will lead to the activation of the node. This evidence list may contain all the morphological variations of a word, common spelling variations on a word or even abbreviations. In this case the evidential node is analogous to a lexeme.

For example

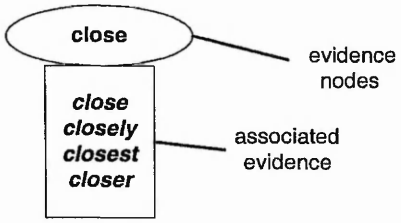


Figure 3-1 Evidence Node Example

3.2. MacroNodes

Depending upon the configuration of the network, in other words the topology of the links and nodes, and the threshold value of a particular node, groups of nodes can have distinct but useful functional and representational properties. A set of these has been devised so as to reflect the representational and computational requirements of natural language.

3.2.1. Abstraction:

Many words in certain contexts have similar meanings (i.e. synonyms). This abstract similarity may be adequately modelled by an interconnected network in which any ONE of a number of child nodes (synonymous words) may trigger the activation of a parent node (synonymous group/ abstract type). The same mechanism staggered onto two hierarchical levels can model hyponymy. Logically an abstracted node is equivalent to an OR gate.

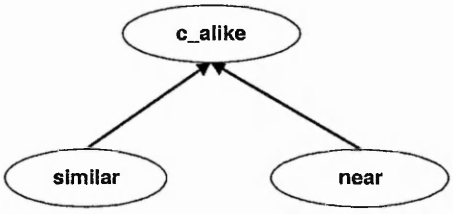


Figure 3-2 Abstraction

3.2.2. Composition:

Certain words have a composite sense which is distinct from the conjugate sense of its parts, for example idioms. (Not as much the whole is greater that the sum of its parts but the whole is different.) This phenomena may be modelled by a network where a single parent node requires ALL of its children nodes to become active before it itself activates. Similarly simple facts may be modelled this way as a composite of parts. Logically an abstracted node is equivalent to an AND gate.

Published in: NATURAL LANGUAGE PROCESSING FOR BUSINESS USE (1997)
Eds. Wilks, Y. et al Unicom, p256-262.

APPENDIX I

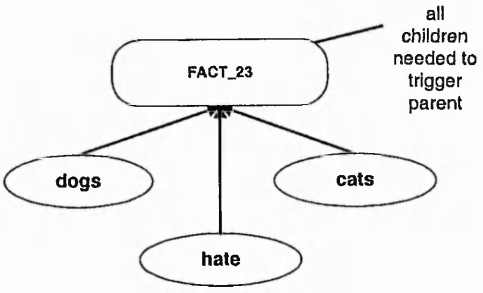


Figure 3-3 Composition fact

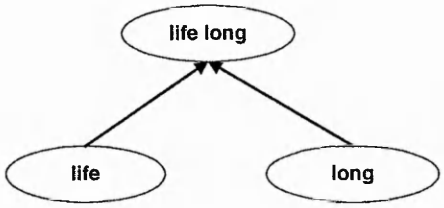


Figure 3-4 Composition idiom

3.2.3. Decomposition

Some words have a composite structure where its parts contribute different senses to the meaning of the word as a whole. For example, at the morphological level consider the word walk-ed: 'walk' informs us of the action, 'ed' informs us that it happened in the past. Or at the semantic level, the word husband, where we may infer that the entity referenced is (male), (human) and (married). Both these cases could be represented by a child node which passes its activation to several parent nodes.

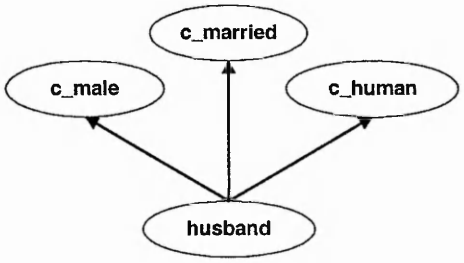


Figure 3-5 Decomposition

3.2.4. Parallelism

Closely associated with decomposition is the concept of parallelism whereby the separate implications (modelled by spreading activation) of the separate items may be computed concurrently.

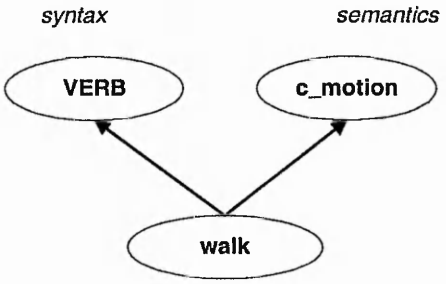


Figure 3-6 Parallelism

3.2.5. Context

Polysemy or homonymy may be modelled in a network by a single child node linking to two parent nodes in parallel. Both of these parents will require input from a node representing the correct context also before it, itself, becomes active. Take the classic example of 'bank', in a financial context it has a completely different meaning to a water/countryside context.

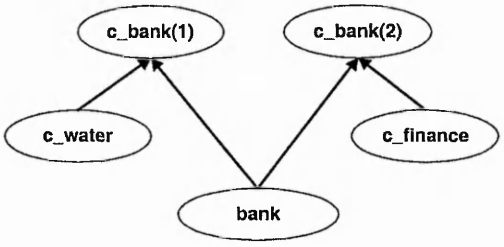


Figure 3-7 Context

3.2.6. Disambiguation

The mutual exclusivity of two hypothesis or perhaps two word meanings may also be modelled by an activation passing network. Using the bank example again it will mean either a 'commercial building' or the 'side of a river' not both. Mutually inhibitory links between sibling nodes ensure that only the node receiving most supporting evidence remains active.

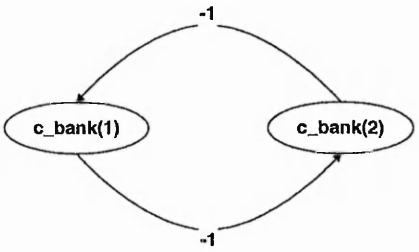


Figure 3-8 Disambiguation

Published in: NATURAL LANGUAGE PROCESSING FOR BUSINESS USE (1997)
Eds. Wilks, Y. et al Unicom, p256-262.

APPENDIX I

3.2.7. Feedback

Sometimes the information necessary to disambiguate two hypothesis is only available from a higher level process, in order to make use of this information it is necessary to implement feedback links whereby a parent node can affect the activation of a child.

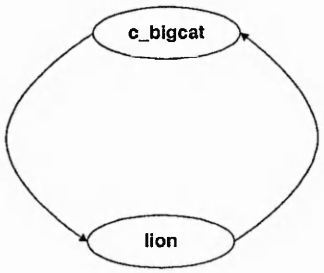


Figure 3-9 Feedback

3.2.8. Connectionist process

As has been demonstrated in numerous connectionist papers a three layer activation passing network is capable of embodying an arbitrary mapping between any two sets of patterns provided there are enough hidden nodes. Further, the Universal Approximation Theorem demonstrates that a three layer network with one input, one output and sufficient nodes in the hidden layers can learn any function. This adds powerful computational power to our simple knowledge architecture.

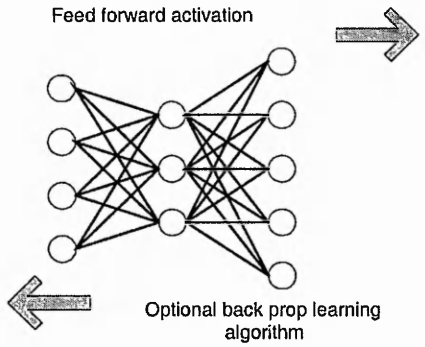


Figure 3-10 Connectionist Process

3.2.9. Logical Process

By virtue of the fact that all logical gates (OR, AND, NOT etc.) can be modelled by an interconnected network, and a standard vonNeuman architecture computer can be implemented entirely with such gates it should be possible to emulate any formal computation within the resource constraints.

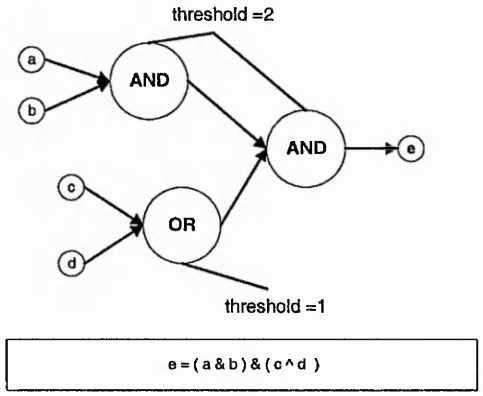


Figure 3-11 Logical Composition

The above points demonstrate that our chosen knowledge architecture has not only powerful representational flexibility but is capable of performing computation under both connectionist and symbolic processing paradigms.

4. FORMALISATION

4.1. Mathematical Formalisation

A particular knowledge base can be characterised by a n-tuple K.

$$K =< C, P(C), p, t, \Sigma, P(\Sigma), e >$$

where briefly

C	set of concepts
P(C)	power set of concepts
p	function that returns parents of a concept
t	function that returns the threshold of a concept
Σ	the set of potential evidence strings
P(Σ)	the power set of strings
e	function that returns the evidence strings

and more fully: C is a set of concepts and n is the number of concepts in C, $C = \{c_1, c_2, c_3, \ldots, c_n\}$. P(C) is the power set (the set of all sets of) C, i.e. $C = \{\{\}\{c_1\}, \{c_1, c_2\}, \{c_2\}, \ldots\}$.

p is function that returns parents of a concept the range of which defined on the power set of C and the domain of which is defined on C, $p: c \in C \rightarrow q \in P(C)$.

t is function that returns threshold of a concept; the range of which defined on the set of real numbers and the domain of which is defined on C, $t: c \in C \rightarrow \Re$

Σ is a finite set of string (say taken from a dictionary) where m is the number of strings, $\Sigma = \{\sigma_1, \sigma_2, \sigma_3, \ldots, \sigma_m\}$ and P(Σ) is the power set (the set of all sets of) Σ, $\Sigma = \{\{\}\{\sigma_1\}, \{\sigma_1, \sigma_2\}, \{\sigma_2\}, \ldots\}$

Published in: NATURAL LANGUAGE PROCESSING FOR BUSINESS USE (1997)
Eds. Wilks, Y. et al Unicom, p256-262.

APPENDIX I

e is a function that returns the evidence list for a concept; the range of which is defined on the power set of Σ and the domain of which is defined on C, $e{:}c \in C \rightarrow q \in P(\Sigma)$

Of use also is the function h which returns the children of a particular node and like the parent function the range of which is defined on the power set of C and the domain of which is defined on C. This need not be represented explicitly as it is redundant information. (However it is to be found in the implementation as redundant data for performance reasons. So h is $h{:}c \in C \rightarrow q \in P(C)$

and defined in terms of p as:

$$h(x) = \bigcup_{i=0}^{i=n} \left\{ \begin{array}{l} c_i \text{ iff } p(c_i) = x \\ \varnothing \text{ iff } p(c_i) \neq x \end{array} \right\}$$

where obviously x∈C.

Further an activation function a is defined for each node c. Again the domain of this function is defined on the set of real numbers. Thus this activation function can be defined in abstract as: $a{:}c \in C \rightarrow \Re$

But specifically in the simple model discussed above where activation is Boolean and all weights are unitary, activation can be defined recursively as.

$$a(c) = \left\{ \begin{array}{l} 1, \sum_{x \in h(c)} a(x) \geq t \\ 0, \sum_{x \in h(c)} a(x) < t \end{array} \right\}$$

Using this formalism to firm up the definition of some of the node types discussed above.

An *evidence* node has a non empty evidence list $|e(c)| \neq \varnothing$. A *compound* node has a threshold value equal to the sum of its children $|h(c)| = t(c)$. An abstract node has a non zero threshold value less than the sum of its children $0 < |h(c)| < t(c)$.

5. EXAMPLES

5.1. Examples

To demonstrate how the above components can be bolted together into more complex functional units with interesting and useful computational and representational properties, two example models are presented.

5.1.1. Disambiguation Model

This example demonstrates how a search may implemented which attempts to match word senses (as opposed to exact strings) using context to identify sense. Further, the sense disambiguation process is bootstrapped in order to deal with the common real world case, outlined above, where the context itself can be ambiguous.

the problem of bootstrapped contextual disambiguation may be solved with an appropriately configured network.

Take the phrase:

"get the train off the tracks"

Let us concentrate on the two words: train and tracks. Further let us take just two senses of each of these words. The source used is Collins Concise Dictionary (1989)[1].

a) Train (n): a line of coaches or wagons coupled together and drawn by a railway locomotive.
b) Train (n): something drawn along, such as the long back section of a dress that trails across the floor.

a) Track (n): a rail or a pair of parallel rails on which a vehicle such as a locomotive runs.
b) Track (n): a course for running of racing.

If train and track are free to take on each of these senses there are four distinct possible interpretations of the above sentence.

The diagram below demonstrates a network with appropriate excitatory and inhibitory links that can bootstrap this disambiguation with not further information.

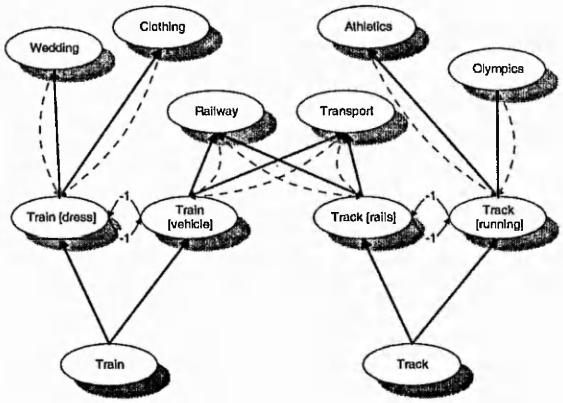


Figure 5-12 Disambiguation model

The network is arranged hierarchically. Those items at the lowest level correspond to the lexical items 'train' and 'track' respectively. Both these items have excitatory links to both their respective senses. Each pair of senses are linked with mutually inhibitory weights to model that fact that a single lexical item can not simultaneously take on two senses. Each sense has excitatory links to its *decomposed* sense primitives. Each sense primitive has a mild excitatory feedback link to its child sense.

[1] Note in actual fact the above dictionary gives 20 distinct definitions for track and 12 definitions for train. This gives 240 possible interpretations of the sentence if only train and track are ambiguous. To complicate matters further the item "tracks" has four senses in its own right which are distinct from the pluralisation of the noun track.

Published in: NATURAL LANGUAGE PROCESSING FOR BUSINESS USE (1997)
Eds. Wilks, Y. et al Unicom, p256-262.

APPENDIX I

If train and track are simultaneously activated all fours sense will receive equal activation. Because of the mutually inhibitory links if either of a pair was slightly more active than the other, this differential would be exacerbated and that node would emerge as a definite winner. However at this stage they are perfectly balanced. This activation will spread to the sense primitives and these primitives will feedback a proportion of this activation to their children nodes. Note that due to the semantic overlap the *railway* and *transport* primitives receive a greater input activation than the others. In turn, through the feedback links, the *Track[railway]* and *Train[vehicle]* sense node receive a disproportionately higher activation. Thus two clear winners will emerge.

5.1.2. Addition Model

To demonstrate the possible computational properties of the network the following model shows how a composite macro-node architecture can be used to search for correct statements of arithmetic. In this case matches can be made against procedurally defined states or truths.

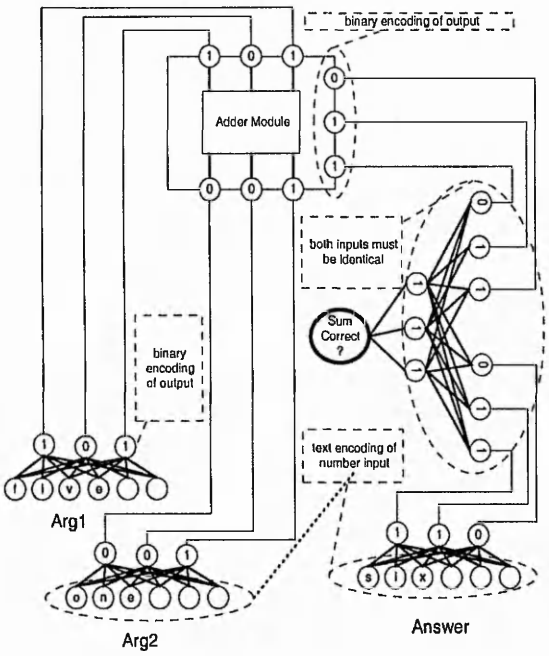


Figure 5-13 Arithmetic

The above network is intended to identify only correct statements of simple addition, for example:

one plus one equals two	✔
three plus three equals seven	✘
4 plus 2 equals siz	✔
2 plus 5 equals six	✘

For reasons of diagrammatic simplicity the macro node units required to validate the operator and equivalence sign (i.e. plus and equal) have been omitted, however it should be easy to see how these would be integrated into the system.

There are three inputs to the system (Arg1,Arg2,Answer) and a single output which validates "sum correct". The three inputs are encoded within in the same form on a fixed string length encoding. If a variable activation node model is used which has *biased* inputs the characters may be encoded as partial activations on the node: (1/26 = 'a' 2/26 = b for example). This is the model demonstrated here, however the same is possible on a Boolean activation model, however more nodes have to used for each character position and each character would have to be either *distributed* or *bucket* encoded.

Each input maps to a series of three nodes upon which the number is binary encoded (allowing the numbers 0-7 to be represented). Two fully interconnected layers are used to perform this mapping, for it is considered a fairly simple mapping. If the mapping were to be more complex, for example if not only the text string "one" were to map to 001 but the text string "1" then hidden layers could be used. (This would be necessary when the features in the input domain were not linearly separable Rumelhart McClelland 1996)

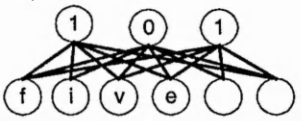


Figure 5-14 Text-binary mapping

The weightings of the networks necessary to perform this mapping could be learned by any of a number of learning algorithms (back propagation for example).

The binary encoding resulting from the two arg values flow into the outputs of the *adder module* whilst the binary encoding of the result is indirectly validated against the output of the adder module.

The adder module is itself a network, with six input nodes and three output nodes. The internal are not represented in the above diagram since by treating it as a modular component it is possible to see two distinct implementations of the function.

Firstly it may be implemented as an adder circuit using nodes to replicate the functionality of AND and OR gates. Alternatively it may be implemented as a feed forward neural network where addition is solved by a simple pattern match process. Either way the end result is the same.

Using another network the output of the adder module can be compared against the binary encoded translation of the result string.

The end result is a network implemented purely from nodes and weights that is capable identifying procedurally defined statements (in this case correct statements of arithmetic) from a data set. Further a high degree of robustness is inherent within the system due to the network mapping between text sting and binary encoding.

This is a network capable of representation and transforms between representations. Representations are chosen in accord with the functionality required and how it effects the desired process. It is capable of and uses both connectionist and logical processes.

Published in: NATURAL LANGUAGE PROCESSING FOR BUSINESS USE (1997)
Eds. Wilks, Y. et al Unicom, p256-262.

APPENDIX I

6. CONCLUSIONS

Above we present a representational schema which is based upon the primitive components of the connectionist model. Upon this it is possible to build macro-nodes which specifically reflect the peculiar representational and computational requirements of natural language. Two examples are presented: the first tackling the problem of bootstrapped sense disambiguation; the second demonstrating the how the computational properties of the network may be used to match truths that are defined procedurally as opposed to declaratively.

Within such a network it is possible to specify search criteria in a far more sophisticated manner than Boolean search strings. It is possible to specify searches which are not solely dependant upon exact string matches but can approximate to the intended sense of that string, by the use of contextual disambiguation. To speculate, a mode of operation is possible whereby domain specific networks could be shared between users, which would map out the meanings of commonly searched for elements. These would form the foundation upon which specific searches would be built.

7. REFERENCES

[1] Crystal, D. (1987) The Cambridge Encyclopaedia of Language. Cambridge University Press: Cambridge.

[2] Lou B, Foxley F (1994a), 'A Simple Text Automatic Marking System'. The AISB Conference of Computational Linguistics for Speech and Handwriting Recognition. Leeds.

[3] Lou, B, Foxley, E (1995) Semantic Understanding for an Automatic Marking System Using Fuzzy Techniques. The 5th Scandinavian Conference on AI (SCAI95) p436-440. IOS Press.

[4] Allott N, Halstead P, Fazackely P (1994), 'A Knowledge Driven Aid to the Automated Assessment of Free Text', AISBQ Vol88.

[5] Allott N, Halstead P, Fazackely P (1994), 'Automated Assessment: Evaluating a Knowledge Architecture for Natural Language Processing', Applications and Innovations in Expert Systems II.

[6] Allott, N. A Novel Knowledge Architecture for Natural Language Processing, PhD Transfer Report, The Nottingham Trent University.

[7]Rumelhart D.E. and McClelland J., Parallel Distributed Processing, Cambridge: MIT, 1968.

Published in: NATURAL LANGUAGE PROCESSING FOR BUSINESS USE (1997)
Eds. Wilks, Y. et al Unicom, p256-262.

AppendixJ.

APPENDIX J

Sequence Clustering Using Time Delay Networks

Nick Allott, Pete Halstead, Pauline Fazackerley
Computing Department,
The Nottingham Trent University,
Burton St,
Nottingham,
NG1 4BU.
email: nma@uk.ac.ntu.doc

Abstract

This paper outlines the form and structure of an activation passing context rich network ideally suited to tasks such as; recovering from noisy or damaged data, recognition or spelling correction. Such a network has been shown to be in many ways similar to N-Gram analysis or the transition matrix of Markov Models. However it is superior in that the scope of context (N) to be considered is dynamically and locally identified for each node. A pair of algorithms are presented which can be used to produce such networks. The first uses global working memory to deal with the time dependant nature of the input data, the second uses time delay nodes within the network. The latter, however, is considered superior as it reduces algorithmic complexity and increases computational efficiency.

1. Introduction

One of the keys to successfully recovering from noisy, damaged or incomplete data within the fields of spelling correction or recognition is to maintain and apply an accurate model of the orthographic regularity of sub-word components and using this to modify the probability estimates from a user error module. N-gram analysis and Markov Models have both been used successfully to perform this task [1][2][3], statistically deriving their information from a corpus. However both are limited in that the depth of context (the value of N) must be specified before the corpus analysis commences: low values of N mean corpora need only be small however transition estimates are of limited value due to the narrow context; high values require ridiculously large corpora for reliable probability estimates [4]. In this paper we look at several algorithms that produce context rich networks from corpus analysis. The network production processes differ from above in that the value for N is locally and dynamically determined for each node.

For convenience, hereon, we use the term CDC (Composition-DeComposition) network to identify a particular type of connectionist network originally developed in [5]. This network was developed initially in order to represent high level semantic information for application to an automated assessment task. However, it was the intention of the authors as stated in [5] that the same representational and problem solving scheme could be applied to more primitive natural language processing tasks. Such tasks as spelling correction or error detection, where a sophisticated models of orthographic or phonetic regularity could improve performance.

We shall in this paper consider firstly the format and specification of the proposed network. Secondly, we shall consider two alternative algorithms for the production of these networks. Finally shall consider a fuzzy element retrieval from the network using the second of these algorithms.

2. Formal Specification of Data Structure and Heuristics

All the algorithms considered below generate a network of the same general form and structure [9] and are modifiable by two configurable heuristics. Each network initially consists of a set of non-connected primitive nodes. In the case of spelling correction these primitive nodes are the 26 letters of the alphabet. When a string is presented to the network the appropriate primitive nodes are activated. A linkage heuristic must be defined which specifies a relationship that must hold between two activated nodes. If the linkage heuristic holds for any two nodes, a context history is recorded for those nodes. These context histories are constantly updated as new strings are processed. When the context history exceeds a threshold defined by the clustering heuristic the two nodes found within each other's context are made into a new parent node and child links are created and maintained. If in the future, the two child nodes become active in the correct order, the parent node becomes active also.

Published in: ARTIFICIAL NEURAL NETS AND GENETIC ALGORITHMS (1997)
Eds. Smith, G. et al Springer, New York, p454-459.

APPENDIX J

Formerly this is specified as: N the nodelist is a set of s nodes, $N = \{n_1, n_2, n_3, \ldots, n_s\}$ .Here each component n is a composite of the with the following attributes: $n = (WordID, Fequncy, Children, Linkage)$. WordID is a string to identify the node, Frequency is an integer in which we record the number of times the node has been found. Children, Parents and Linkage are all lists and defined as follows: $Children_x = \{c_{1x}, c_{2x}, c_{3x}, \ldots c_{qx}\}$, where each member is of type c comprising the single attribute: $c = (WordID)$.

Similarly $Parents_x = \{p_{1x}, p_{2x,} p_{3x}, \ldots p_{rx}\}$ and: $p = (WordID)$.

Finally $Linkage_x = \{l_{1x}, l_{2x}, l_{3x}, \ldots l_{tx}\}$ where 1 is a composite type with two attributes $l = (WordID, Frequency)$

If we use the syntax where a function of the same name as an attribute performed on an appropriate composite extracts the value of that attribute, it follows that: $Freq(n_x) = \sum_1^s frequecny(l_{ix})$. In other words the frequency of a node is equal to the sum of the frequency of the distinct contexts it has been found in. A useful variable is the total number of primitive nodes encountered $Global = \sum_1^s freqprims(i)$ where:

$$freqprims(i) = \begin{Bmatrix} freq(n_i), & |word(n_i)| = 1 \\ 0, & |word(n_i)| \neq 1 \end{Bmatrix}$$

and $Word(n_i) =$ length of string

The input string we may interpret mathematically as a set of ordered pairs. $String = \{k_1, k_2, k_3, \ldots, k_z\}$. Each element is a composite with the first attribute the letterID the second is the position in the string. i.e. $k = (letterID, position)$.

Given this the linkage heuristic may be defined as a Boolean returning function. Simple right adjacency heuristic could be defined as:

$$Link(x, y) = \begin{Bmatrix} 1, & position(k_x) = position(k_y) + 1 \\ 0, & position(k_x) \neq position(k_y) + 1 \end{Bmatrix}$$

Clustering heuristics are also Boolean returning functions. A simple heuristic relying on absolute frequency could be defined as

$$Cluster(x, y) = \begin{Bmatrix} 1, & \frac{freq(l_{xy})}{Global} \geq 0.01 \\ 0, & \frac{freq(l_{xy})}{Global} < 0.01 \end{Bmatrix}$$

As a general note the linkage heuristic defines the form or nature of the network, i.e. what the network actually represents. In the simplest case this is right adjacency. The clustering heuristic determines the breath and depth of the network, and is usually a statistical threshold. A low threshold will produce a deep tree that will have nodes that correspond to full lexical items.

3. Time Series Through Global Working Memory

A standard statistical procedure attempts to identify a relationship between N variables. The algorithm outlined does this with a set of primitive nodes, however when a strong relationship is found between two or more nodes they are concatenated into a new node and the procedure will then look for relationships with this node also. The number of variables is therefore constantly growing as the data is recursively applied to itself.

If we are to implement this as an adaptive network there are two problems that have to be overcome:

1) With strings we are dealing with time dependent data. Within a standard connectionist network it is difficult to come up within an encoding where a nodes position within the string (the time dependant data) is preserved.
2) Within the learning phase we have the problem of making the distinction between the type and token of a node. This is best exemplified in the word banana. Here there are two instances of the cluster 'ana' that occupy non unique positions within the string. If there is a single node to represent the type 'ana' we must at least temporarily be able to discriminate two distinct instances of the token 'ana' at different but overlapping positions.

One solution to this problem is to couple a working memory onto the connectionist network. Working memory becomes the blackboard upon which all instances of nodes are recorded as they are activated. A node instance is created within working memory as soon as it receives input from its first child node and the time at which it was first activated is bound to this instance. When the node receives input from its final child node the node itself is activated and this time is also recorded.

In outline, a new unconnected primitive node is instantiated for each primitive unit encountered in a string. Whenever this node is subsequently encountered

Published in: ARTIFICIAL NEURAL NETS AND GENETIC ALGORITHMS (1997)
Eds. Smith, G. et al Springer, New York, p454-459.

within a string this node is instanced in working memory, and remains there for a specified lifespan. The linkage heuristic is constantly applied to working memory and contextual information is recorded for any node instances satisfying the criteria.

When the context history combining two (or more) nodes exceeds a value specified by the clustering heuristic a new compound unit is instantiated which describes the conjunction of the children nodes.

Certain metrics are attached to the various entities in the tree. It is with respect to these metrics that the clustering and linkage heuristic are defined. Of significance is the measure of frequency (attached to nodes, links and global measures). An absolute measure of frequency would be inadequate for inter-node comparisons as new nodes are being created all the time and the new nodes are unaware of how often they occurred before they were created. More suitable would be a measure of acquisition velocity, that is to take an estimate of the differential of frequency over time. It was found that the regular resetting of all frequency values served as a satisfactory approximation to this.

The algorithm as discussed may be defined in pseudo code as follows:

```
LOOP (for all strings)
{
   LOOP (for all units)
   {
      IF (node not previously seen)
         DO Instantiate_Node
      ELSE (node already seen)
      {
         DO Activate_This_Node
         DO Pass_Activation_Parents
         DO Instance_Node_In_Memory
         LOOP (all instanced nodes)
         {
            IF (LINKAGE HEURISTIC)
            {
               DO Add_Link_To_History
               IF (THRESHOLD HEURISTIC)
               {
                  DO Make_Nodes_Compounds
                  DO Link_Children+Parent
               }
            }
         }
      }
      DO Clear_Working_Memory
   }
}
```

Although there are several inefficiencies in this procedure it does work and rapidly produces networks of considerable complexity embodying deep contextual information from the problem domain. Training time is roughly proportional to the square of the current number of identified nodes and therefore assuming a constant rate of node acquisition increases exponentially over time. The rate of node acquisition is however completely determined by the clustering heuristic and therefore further generalisation is difficult.

In summary global working global working memory is being used to perform two distinct activities.

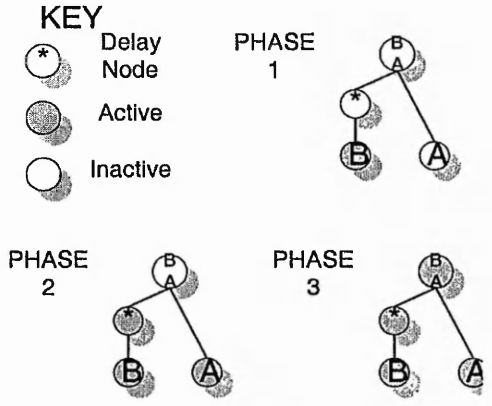
1) Sequence Processing: to allow time dependant information to propagate up the network so that the 'th' node only becomes active when the 't' node and the 'h' node are activated immediately after one another.
2) Learning: to provide a blackboard on which activated nodes can be recorded and upon which the linkage heuristic can be applied.

4. Time Series Through The Synchronous Activation Update Of Gateway Nodes

With a more sophisticated activation model it is possible to remove the need for working memory to model the propagation of time dependant data. A stricter activation model must be applied where each single unit evidence node is activated at distinct phases. Consider the example of 'banana': 'b' must be activated on phase 1, 'a' activated on phase 2, 'n' activated on phase 3 etc.

A composite when identified must be gatewayed with activation delay nodes to offset the activation latency of primary and secondary child nodes. For example if 'ba' was identified as a composite node 'b' and 'a' would both be recorded as child nodes, but a delay node would be inserted between 'b' and its parent 'ba'. This would give an activation schedule as follows:

Phase 1: 'b' becomes active.
Phase 2: 'b' passes activation to delay node, 'a' becomes active.
Phase 3: delay node passes activation to 'ba', a passes activation to 'ba'



This way 'ba' receives its activation from both its child nodes at the same time even though the respective child nodes themselves were active at different times.

Published in: ARTIFICIAL NEURAL NETS AND GENETIC ALGORITHMS (1997)
Eds. Smith, G. et al Springer, New York, p454-459.

Not only is this computationally more efficient but its is architecturally more consistent with the connectionist model. Working memory is now only necessary for the application of the learning phase , i.e. the application of the linkage heuristic.

Further, as we shall see below, such a model makes it easier to introduce a notion of variable activation which will be necessary for the recovery of damaged data.

5. Application of Network

As discussed such context rich trees are ideally applicable to recognition type tasks. In their simplest they can be shown to produce identical information to that produced by N-Gram analysis or the transition matrix of a Markov model. However, we do not have to specify our grain of analysis (bigram - trigram etc.) before processing takes place nor must the grain be unique throughout the analysis. If the heuristics are set up correctly the algorithm dynamically and locally determines these for each node. We can therefore use the trees in the same way as we would with N-gram or Markov Models i.e. produce estimates of word probability from supplied evidence, as a multiple of transition probabilities between primitives. But using the context tree we may augment these estimates by describing our units in higher level terms and using transition probabilities between these to modify our original figure. Similarly we could use the identified tokens as the principal resource for Ngram [8][10] indexing as used for spelling correction, with parallel benefits.

But, by extending the synchronous activation model discussed in the final algorithm we could introduce the notion of partial activation into the network. By applying a time decay function to each node we can see how two perfectly timed child nodes will lead to full activation of a parent node, whereas if the child nodes are slightly misplaced the parent node will only be partially activated. If the clustering heuristic was adjusted such that the tree was built to the deepest level ie full lexical items, the lexical items that most closely matched the supplied evidence would become most active. Further by implementing mutually inhibitory nodes between all full lexical items through interactive activation and competition [7] the network itself could resolve the best fitting match.

6. Further Work

The learning algorithms presented above consider only the composition element of CDC networks. No algorithm has yet been developed which attempts to automatically identify the decomposition element and hence the various subtypes of nodes. This is an area within the problem domain ripe for exploitation. For example constants and vowels are the two crudest subtypes within the domain each of which has a completely distinct contextual distribution which could be used to great effect in modifying probability estimates for data recovery. This is seen as the next phase for development.

7. References

[1] Jelinek F, Mercer R and Bahl. 'Continuous Speech Recognition: Statistical Methods', IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol PAM-5, 1983.

[2] Brown P, Lee H and Spohrer, 'Bayesian Adaptation in Speech Recognition', Proceeding of the ICCASP, Boston, 761-764.

[3] Riseman E and Hanson A. 'A Contextual Postprocessing System for Error Correction Using Binary N-Grams', IEEE Transactions on Computers, Vol C-23, 490-493, 1974.

[4] Keenan F. 'Large Vocabulary Syntactic Analysis for Text Recognition', PhD Thesis, The Nottingham Trent University, 1992.

[5] Allott N, Halstead P, Fazackely P. 'A Knowledge Driven Aid to the Automated Assessment of Free Text', AISBQ Vol88, 1994.

[6] Anick P, Berger, 'Lexical Structures for Linguistic Inference', Lecture Notes in Artificial Intelligence, Vol 627, 121-135, 1992.

[7] Rumelhart D E, McClelland J, 'Parallel Distributed Processing' Cambridge, MIT, 1968.

[8] Du M W, Chang S C.'An Approach to Designing Very Fast Approximate String Matching Algorithms' IEEE Transactions on Knowledge and Data Engineering, Vol 6, No 4, 620-633, 1994.

[9] Allott N, Halstead P, Fazackely P. 'Clustering Algorithm to Produce Context Rich Networks', Proceedings of Applied Decision Technologies, p265-269, 1995.

[10] Kukich K. 'Techniques for Automatically Correcting Words in Text', ACM Computing Surveys, Vol 24, No 4, 377-439, 1992.

Published in: ARTIFICIAL NEURAL NETS AND GENETIC ALGORITHMS (1997)
Eds. Smith, G. et al Springer, New York, p454-459.

APPENDIX K

Composition, Clustering and Predictor Pruning in Hierarchical Networks

N. Allott, M. Allott, P. Fazackerley and P. Halstead

Computing Department,
The Nottingham Trent University,
Burton St,
Nottingham,
NG1 4BU.
England
email: nma@uk.ac.ntu.doc
Consensus Software Ltd.

Abstract

This paper develops some previously published ideas on clustering within connectionist networks. Networks are grown from individual unattached nodes which represent the attributes within the un-mined data base. Three distinct algorithms are iteratively applied to unattached nodes. The first recursively identifies composite nodes which model conjunctions of attributes; the second identifies clustered nodes which model disjunctions of attributes and the third prunes the network, stripping out identified nodes which have no predictive value. A developed network can serves a perceptual framework which simplifies a complex database, and through the sensible choice of pruning criteria can be tuned for application to specified predictive tasks.

1. Introduction

We address in this paper the issue of growth in connectionist networks [1] [2]. A process is described which consists of three distinct processing components. Each of these components is a applied iteratively to a primitive node set, in order to identify new nodes and the appropriate linkages between them. Each of these new nodes is derived statistically from the input domain and therefore represents the regularities implicit within the data. The network is also subjected to a pruning process so that identified nodes also serve some functional value. The produced network is therefore a sophisticated reflection of the distribution of the units to be found within the input domain.

Let us examine the process of data mining in abstract. Typically we have at our disposal a data set consisting of a series of entities. Each of theses entities is characterised by a set of attributes. This set of attributes we shall call the atomic description, for it is the most detailed description of the data for which there is empirical support. A typical data mining application will want to do one of several thing with this data:

1) predict the behaviour of unseen entities on the basis the data available in the set.
2) segment the data set (and new entities) into one of several predetermined categories that have known behaviour.
3) identify any regularities within the data set that could be used to competitive advantage.

Obviously the computational processes to perform these prediction or segmentation tasks must be a function of the atomic description. However there is a problem, learning algorithms such as back propagation do not perform well with large data sets. A large atomic description can lead to a combinatorial explosion of linkages between nodes and consequently exponentially increasing processing times.

The algorithms we present here are an attempt to produce a representation of the atomic description which is both efficient, economical and tuned towards the desired application. However the processes used to identify such descriptions have some interesting side effects. We find that in order to produce optimal and tuned descriptions it is necessary to record the contextual history of each node. An evolved network therefore contains rich information concerning the distribution of the respective attributes, information that can be utilised for predictive purposes.

Typically what do we want from the data?

1) pertinent statistical relationships between data elements that can be monopolised on.

Published in: NEURAL NETWORKS AND DATA MINING (1997)
Eds. Gregory, P. et al Unicom, p181-209.

APPENDIX K

2) high order elements that mat be identified within the data set which expose the internal regularity of the data.
3) most importantly: a simplification of the data set (only possible after (2)) which can be used for descriptive, computational and modelling applications.

What precisely is meant by higher order elements? Imagine each entity within the data base is described by just two variables, height and weight for example. Further imagine each of these variables is segmented into seven categories: ([very low][low][below average][average][above average] [high][very high]). If this is to be the atomic description what high order characteristics can be derived from this which simplify the description for use in processing tasks?

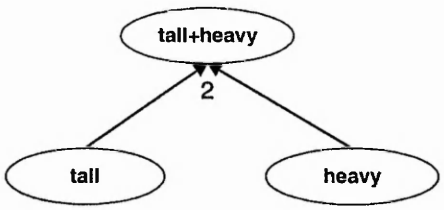
We identify two such high order elements: composite elements and clustered elements.

A composite element requires two or more atomic attributes to be active within the data set. Within the above data set this could be [very tall] and [very heavy] simultaneously, for example. In many ways this is similar to a logical AND node.

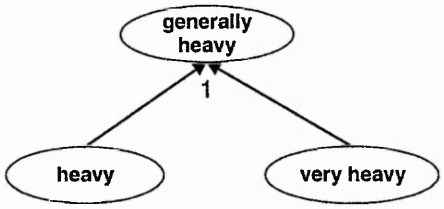
A compound node requires any one of a group of atomic attributes to be active within the data set. Within the above data set this could be any weight above average [above average] [heavy][very heavy], for example. In many ways this is similar to a logical OR node.

Interestingly both these high order types can be easily modelled with primitive connectionist units.

A composite node can model a composite attribute by having a single parent node that has a threshold value equal to the number of children it has. Both children will therefore have to be active before the parent node itself becomes active.



A clustered node can model a clustered attribute by having a single parent node that has a threshold value equal one. Any of the children will therefore lead to the activation of the parent node.



By applying algorithms to identify composite and clustered nodes iteratively interesting recursive relationships build up between the nodes themselves.

For composite nodes we can see how that once identified the [[very tall] [very heavy]] composite could be further bound to a further variable [very young] for example to identify an even more tightly defined subset of the data. Similarly clustered nodes may be clustered: the [generally heavy] cluster could be further clustered with [very small] to define the set of people that were above average weight or small.

But also we can have composites of clusters and clusters of composites. [generally heavy] and [generally tall] is a *potentially useful* composite of clusters. [[very tall][average weight]] or [[tall][below average weight]] identifies a cluster of composites which describes tall people that are an equivalent amount underweight.

The use of the phrase *potentially useful* brings us to another issue. The nodes that are identified by the above algorithms are identified on a purely statistical basis. How are we to ensure that these nodes are actually useful for a specific purpose? This is where the pruning phase comes in. Within the pruning phase we objectify the purpose of our network and use this as the functional criteria by which by prune (or don't prune) a particular node.

Now to discuss each of these processes in more detail.

2. Composition

The aim is to identify nodes that represent composite attributes within the data set. If a composite attribute is to be defined in statistical terms it is those attributes that are found to frequently co-occur. This is done as follows.

An unattached node is instantiated for each atomic attribute within the data set. The attributes of a particular entity within the data set are presented to the network and the appropriate nodes are activated. Each node is *aware* of the others activation and their mutual co-occurrence is recorded in their respective history links. A new entity is presented and the process is repeated. The history links between certain nodes will slowly strengthen, reflecting the respective distributions of the nodes. Once a history link surpasses a predetermined critical strength a new node is instantiated which is to represent the composition of the two relevant *children* nodes and two activation links are created which will pass activation from children to parents.

Published in: NEURAL NETWORKS AND DATA MINING (1997)
Eds. Gregory, P. et al Unicom, p181-209.

APPENDIX K

If in the future both children nodes are to become active the parent node will be activated also.

In this manner a network will slowly grow from the primitive seeds. Also the entire statistical process is recursive in that new identified nodes are also recorded in the history lists and so can be used in the formation of yet further , more complex nodes.

2.1. Formalisation

A particular network can be characterised by a 5-tuple $\langle P,N,t,l,h\rangle$. Where P is the initial set of primitive nodes and N is the entire list of nodes, initially N=P. t is a function that represents the activation threshold of each node. l is a function which models the linkages that hold between nodes and is thus defined: $l: N,N \rightarrow \{0,1\}$ if linkages are to be Boolean in nature, and $l: N,N \rightarrow [0,1]$ if the interval is to be continuous. h is a function that models the contextual history of a nodes and so is defined $h: N,N \rightarrow W$, where W is the set of whole numbers. The frequency with which a particular node has been observed f(N) can therefore be calculated from its history: $f(N) = \sum_{i \in P} h(N,i)$. The global frequency g(), which is the total number of nodes processed can then be defined: $g() = \sum_{i \in P} f(i)$.

The process of clustering is to identify new node x such that N=N∪x, and x is to represent the cluster of two nodes n_1 and n_2. The threshold between of this new node is two, t(x)=2. And two linkages must hold between the children and the parent composite: $l(n_1,x)=1$ and $l(n_2,x)=1$.

The criteria that must be satisfied if a new x is to be identified, as mentioned above, is entirely adaptable, but is most simply be defined on a statistical basis. For example the criteria

$$\frac{h(n_1,n_2)}{f(n_1)} \geq \kappa \text{ AND } f(n_1) \geq \lambda$$

Where κ is a statistical threshold and λ is a constant to ensure statistical significance.

3. Clustering

The aim is to identify nodes that represent clustered attributes within the data set. If this cluster is to be identified from the data available, that is the other attributes, the only criteria by which this cluster may be made is commonality of contextual frequency. Fortunately the history links required by the former algorithm is the ideal resource with which to evaluate contextual history.

In order to identify the best two nodes to be clustered a process of what we call *bounce back activation* is used. This is done as follows. Select a node at random and activate this node fully. Pass the activation out through the "*history links*" to the others and let these other nodes take on an activation proportional to the strength of the history link. Once received bounce back this activation, again down the history nodes but this time let the strength of this activation be proportional to the difference between the strength of the activation of the node and the strength of the history link. The emitting node, should of course be fully activated. But the most similar nodes will also take on a high activation.

3.1. Formalisation

Take the same network formalisation given preciously. Further assume the existence of a distance function δ which returns a measure of the similarity between two nodes,. $\delta: N,N \rightarrow [0,1]$ If N_1 and N_2 are to be identified as a cluster then N_1 should be closest to N_2:

$$\forall x \in N \setminus \{N_2\} \ \delta(N_1,x) > \delta(N_1,N_2)$$

The distance function δ itself is best defined in terms of the intermediate bounce back function b(). The bounce back function is determined for each node in respect of the specific node c which is being clustered, and is simply a function of the strength of the link between the two thus $b(c,n) = h(c,n)$. The distance function is then:

$$\delta(c,n) = \frac{\sum_{i \in Qn} sim(b(c,i),h(n,i))}{|Qn|}$$

Where sim() is a measure of the similarity between activation and link strength. And Qn is the set of nodes which are attached by history links to n.

4. Predictor Pruning

The above two algorithms identify high order nodes from the primitive data set on purely statistical grounds. If we wish to tune the network towards a particular application, the statistical grounds on their own a insufficient to justify the nodes existence within the network. For this reason the network is regularly pruned. That is nodes that do not satisfy some predefined functional criteria are taken out of the network. The pruning function itself should be the formalisation of the purpose of the network.

There is an important issue here to do with the regularity of the pruning. It does not pay to prune too often. Much as in a game of chess where the fruits of a particular strategy are not apparent until several moves into the game, it frequently pays to let several layers of non-performing nodes build up before the entire branch is stripped.

4.1. Formalisation

The process of the removal of node x is simply: $N = N \setminus \{x\}$.

All connected activation links must be nulled.

$$\forall_{n \in N} l(x,n) = 0 \wedge l(n,x) = 0$$

And similarly history links:

Published in: NEURAL NETWORKS AND DATA MINING (1997)
Eds. Gregory, P. et al Unicom, p181-209.

$$\forall_{n \in N} h(x,n) = 0 \wedge h(n,x) = 0$$

The criteria under which this node removal takes place is varied and functionally dependant. If the network is being used for variable prediction, such a criteria may be the Bayesian probability of a variable v given a node x being less than some predetermined threshold θ. For example $p(v|x) < \theta$.

5. Application - How could the data be used?

We have described how the above procedures can be used to generate networks of considerable complexity that model any deep statistical regularity within the data., and simplify the data through the identification of higher order elements. But how can these networks be used?

5.1.1. Net input

Using a back propagation type algorithm for training neural networks to segment or predict the behaviour of a population is an obvious application of neural network technology. However the size of the databases frequently prohibits this due to combinatorial explosion of linkages and consequent exponentially rising processing times. The above process can be used to alleviate this problem. If successful a network will be produced consisting of high order nodes from the data set. A particular member of the population can be characterised with these higher order nodes more economically and efficiently. Further by choosing an appropriate pruning function we can tune the data to the task in hand essentially performing useful pre-processing on the data.

5.1.2. Predictor

The generated node itself can be used as a source from which to predict the value of an independent variable, particularly if this variable is used within the pruning heuristic. We can of course analyse the Bayesian probability of this independent variable given the presence of the various atomic attributes. With a fully evolved network this initial estimate can be greatly enhanced.

5.1.3. Query for logical composition

The network discussed is localist, consequently the nodes may be interpreted symbolically. Also due to the similarity between composite and clustered nodes, and the logical notions of AND and OR respectively, the grown network may be interpreted in terms of logical rules. This means if a node has been identified which is particularly useful, as a criteria for classification for example, it is very easy to translate this network into a set of production rules.

5.1.4. Query for statistical distribution

Similarly, the symbolic nature of the network renders it ripe for statistical exploitation. The history linkages of the network contains detailed information concerning the contextual distribution of the individual nodes, in much the same way as Markov model [3] or N-gram analysis [4][5] would. However unlike these and similar statistical techniques were the scope of analysis is predetermined before processing takes place, the scope on analysis is dynamically and locally determined at each node.

A grown network serves as a rich source of statistical information. As a general rule if the existence of a particular node/attribute is an independent random event then the contextual history of that node will have a random distribution. Where the contextual history is not random there is useful information ready to exploit.

6. Conclusions

In general the three algorithms discussed above provide an evolutionary procedure for synthesising the optimum description of a data set, for use in a specified application. The two relationships presented, composition and clustering, although somewhat arbitrary are easy to model in a connectionist network and have a solid logical foundation. The iterative application of the network therefore leads to sophisticated models of considerable complexity. The third algorithm, the pruning process, provides the evolutionary forcing factor to ensure that the network has relevance to the application in hand.

A produced network and the activation links that exist between nodes therefore provides a perceptual framework for the original data set, both simplifying and tuning the data.

Specifically, however, the implementation of the above algorithms necessitates the use to the history linkages, which prove to be useful items in their own right. They provide a deep contextual history of each node, which can be used to divine the relationships that exist, not only between the original atomic attributes, but any clustered and composite attributes subsequently identified.

7. References

[1] Allott,N, Fazackerley, P, Halstead P (1995) .A Clustering Algorithm for Producing Context Rich Networks. In J. Taylor ed., Neural Networks and their Applications, pp220-29, Chichester: Wiley.

[2] Allott,N, Fazackerley, P, Halstead P (1997) Sequence Clustering Using Time Delay Networks, submitted ICANNGA conference.

[3] Jelinek F, Mercer R and Bahl (1983), 'Continuous Speech Recognition: Statistical Methods', IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol PAM-5.

[4] Brown P, Lee H and Spohrer, 'Bayesian Adaptation in Speech Recognition', Proceeding of the ICCASP, Boston, 761-764.

[5] Riseman E and Hanson A (1974), 'A Contextual Postprocessing System for Error Correction Using Binary N-Grams', IEEE Transactions on Computers, Vol C-23, 490-493.

[7]Rumelhart D.E. and McClelland J., Parallel Distributed Processing, Cambridge: MIT, 1968.

Published in: NEURAL NETWORKS AND DATA MINING (1997) Eds. Gregory, P. et al Unicom, p181-209.

AppendixK.

APPENDIX K

Composition, Clustering and Predictor Pruning in Hierarchical Networks

N. Allott, M. Allott, P. Fazackerley and P. Halstead

Computing Department,
The Nottingham Trent University,
Burton St,
Nottingham,
NG1 4BU.
England
email: nma@uk.ac.ntu.doc
Consensus Software Ltd.

Abstract

This paper develops some previously published ideas on clustering within connectionist networks. Networks are grown from individual unattached nodes which represent the attributes within the un-mined data base. Three distinct algorithms are iteratively applied to unattached nodes. The first recursively identifies composite nodes which model conjunctions of attributes; the second identifies clustered nodes which model disjunctions of attributes and the third prunes the network, stripping out identified nodes which have no predictive value. A developed network can serves a perceptual framework which simplifies a complex database, and through the sensible choice of pruning criteria can be tuned for application to specified predictive tasks.

1. Introduction

We address in this paper the issue of growth in connectionist networks [1] [2]. A process is described which consists of three distinct processing components. Each of these components is a applied iteratively to a primitive node set, in order to identify new nodes and the appropriate linkages between them. Each of these new nodes is derived statistically from the input domain and therefore represents the regularities implicit within the data. The network is also subjected to a pruning process so that identified nodes also serve some functional value. The produced network is therefore a sophisticated reflection of the distribution of the units to be found within the input domain.

Let us examine the process of data mining in abstract. Typically we have at our disposal a data set consisting of a series of entities. Each of theses entities is characterised by a set of attributes. This set of attributes we shall call the atomic description, for it is the most detailed description of the data for which there is empirical support. A typical data mining application will want to do one of several thing with this data:

1) predict the behaviour of unseen entities on the basis the data available in the set.
2) segment the data set (and new entities) into one of several predetermined categories that have known behaviour.
3) identify any regularities within the data set that could be used to competitive advantage.

Obviously the computational processes to perform these prediction or segmentation tasks must be a function of the atomic description. However there is a problem, learning algorithms such as back propagation do not perform well with large data sets. A large atomic description can lead to a combinatorial explosion of linkages between nodes and consequently exponentially increasing processing times.

The algorithms we present here are an attempt to produce a representation of the atomic description which is both efficient, economical and tuned towards the desired application. However the processes used to identify such descriptions have some interesting side effects. We find that in order to produce optimal and tuned descriptions it is necessary to record the contextual history of each node. An evolved network therefore contains rich information concerning the distribution of the respective attributes, information that can be utilised for predictive purposes.

Typically what do we want from the data?

1) pertinent statistical relationships between data elements that can be monopolised on.

Published in: NEURAL NETWORKS AND DATA MINING (1997)
Eds. Gregory, P. et al Unicom, p181-209.

APPENDIX K

2) high order elements that mat be identified within the data set which expose the internal regularity of the data.
3) most importantly: a simplification of the data set (only possible after (2)) which can be used for descriptive, computational and modelling applications.

What precisely is meant by higher order elements? Imagine each entity within the data base is described by just two variables, height and weight for example. Further imagine each of these variables is segmented into seven categories: ([very low][low][below average][average][above average] [high][very high]). If this is to be the atomic description what high order characteristics can be derived from this which simplify the description for use in processing tasks?

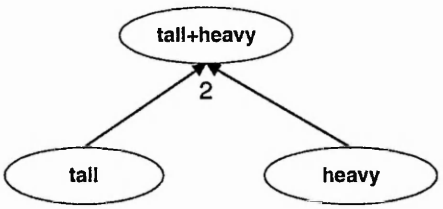
We identify two such high order elements: composite elements and clustered elements.

A composite element requires two or more atomic attributes to be active within the data set. Within the above data set this could be [very tall] and [very heavy] simultaneously, for example. In many ways this is similar to a logical AND node.

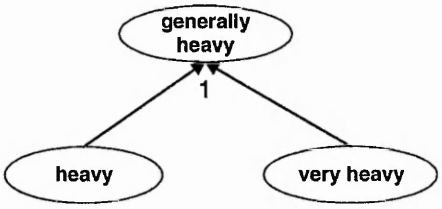
A compound node requires any one of a group of atomic attributes to be active within the data set. Within the above data set this could be any weight above average [above average] [heavy][very heavy], for example. In many ways this is similar to a logical OR node.

Interestingly both these high order types can be easily modelled with primitive connectionist units.

A composite node can model a composite attribute by having a single parent node that has a threshold value equal to the number of children it has. Both children will therefore have to be active before the parent node itself becomes active.



A clustered node can model a clustered attribute by having a single parent node that has a threshold value equal one. Any of the children will therefore lead to the activation of the parent node.



By applying algorithms to identify composite and clustered nodes iteratively interesting recursive relationships build up between the nodes themselves.

For composite nodes we can see how that once identified the [[very tall] [very heavy]] composite could be further bound to a further variable [very young] for example to identify an even more tightly defined subset of the data. Similarly clustered nodes may be clustered: the [generally heavy] cluster could be further clustered with [very small] to define the set of people that were above average weight or small.

But also we can have composites of clusters and clusters of composites. [generally heavy] and [generally tall] is a *potentially useful* composite of clusters. [[very tall][average weight]] or [[tall][below average weight]] identifies a cluster of composites which describes tall people that are an equivalent amount underweight.

The use of the phrase *potentially useful* brings us to another issue. The nodes that are identified by the above algorithms are identified on a purely statistical basis. How are we to ensure that these nodes are actually useful for a specific purpose? This is where the pruning phase comes in. Within the pruning phase we objectify the purpose of our network and use this as the functional criteria by which by prune (or don't prune) a particular node.

Now to discuss each of these processes in more detail.

2. Composition

The aim is to identify nodes that represent composite attributes within the data set. If a composite attribute is to be defined in statistical terms it is those attributes that are found to frequently co-occur. This is done as follows.

An unattached node is instantiated for each atomic attribute within the data set. The attributes of a particular entity within the data set are presented to the network and the appropriate nodes are activated. Each node is *aware* of the others activation and their mutual co-occurrence is recorded in their respective history links. A new entity is presented and the process is repeated. The history links between certain nodes will slowly strengthen, reflecting the respective distributions of the nodes. Once a history link surpasses a predetermined critical strength a new node is instantiated which is to represent the composition of the two relevant *children* nodes and two activation links are created which will pass activation from children to parents.

Published in: NEURAL NETWORKS AND DATA MINING (1997)
Eds. Gregory, P. et al Unicom, p181-209.

APPENDIX K

If in the future both children nodes are to become active the parent node will be activated also.

In this manner a network will slowly grow from the primitive seeds. Also the entire statistical process is recursive in that new identified nodes are also recorded in the history lists and so can be used in the formation of yet further , more complex nodes.

2.1. Formalisation

A particular network can be characterised by a 5-tuple $\langle P,N,t,l,h \rangle$. Where P is the initial set of primitive nodes and N is the entire list of nodes, initially N=P. t is a function that represents the activation threshold of each node. l is a function which models the linkages that hold between nodes and is thus defined: $l: N,N \rightarrow \{0,1\}$ if linkages are to be Boolean in nature, and $l: N,N \rightarrow [0,1]$ if the interval is to be continuous. h is a function that models the contextual history of a nodes and so is defined $h: N,N \rightarrow W$, where W is the set of whole numbers. The frequency with which a particular node has been observed f(N) can therefore be calculated from its history: $f(N) = \sum_{i \in P} h(N,i)$. The global frequency g(), which is the total number of nodes processed can then be defined: $g() = \sum_{i \in P} f(i)$.

The process of clustering is to identify new node x such that N=N∪x, and x is to represent the cluster of two nodes n_1 and n_2. The threshold between of this new node is two, t(x)=2. And two linkages must hold between the children and the parent composite: $l(n_1,x)=1$ and $l(n_2,x)=1$.

The criteria that must be satisfied if a new x is to be identified, as mentioned above, is entirely adaptable, but is most simply be defined on a statistical basis. For example the criteria

$$\frac{h(n_1,n_2)}{f(n_1)} \geq \kappa \text{ AND } f(n_1) \geq \lambda$$

Where κ is a statistical threshold and λ is a constant to ensure statistical significance.

3. Clustering

The aim is to identify nodes that represent clustered attributes within the data set. If this cluster is to be identified from the data available, that is the other attributes, the only criteria by which this cluster may be made is commonality of contextual frequency. Fortunately the history links required by the former algorithm is the ideal resource with which to evaluate contextual history.

In order to identify the best two nodes to be clustered a process of what we call *bounce back activation* is used. This is done as follows. Select a node at random and activate this node fully. Pass the activation out through the "*history links*" to the others and let these other nodes take on an activation proportional to the strength of the history link. Once received bounce back this activation, again down the history nodes but this time let the strength of this activation be proportional to the difference between the strength of the activation of the node and the strength of the history link. The emitting node, should of course be fully activated. But the most similar nodes will also take on a high activation.

3.1. Formalisation

Take the same network formalisation given preciously. Further assume the existence of a distance function δ which returns a measure of the similarity between two nodes,. $\delta: N,N \rightarrow [0,1]$ If N_1 and N_2 are to be identified as a cluster then N_1 should be closest to N_2:

$$\forall x \in N \setminus \{N_2\} \; \delta(N_1,x) > \delta(N_1,N_2)$$

The distance function δ itself is best defined in terms of the intermediate bounce back function b(). The bounce back function is determined for each node in respect of the specific node c which is being clustered, and is simply a function of the strength of the link between the two thus $b(c,n) = h(c,n)$. The distance function is then:

$$\delta(c,n) = \frac{\sum_{i \in Qn} sim(b(c,i),h(n,i))}{|Qn|}$$

Where sim() is a measure of the similarity between activation and link strength. And Qn is the set of nodes which are attached by history links to n.

4. Predictor Pruning

The above two algorithms identify high order nodes from the primitive data set on purely statistical grounds. If we wish to tune the network towards a particular application, the statistical grounds on their own a insufficient to justify the nodes existence within the network. For this reason the network is regularly pruned. That is nodes that do not satisfy some predefined functional criteria are taken out of the network. The pruning function itself should be the formalisation of the purpose of the network.

There is an important issue here to do with the regularity of the pruning. It does not pay to prune too often. Much as in a game of chess where the fruits of a particular strategy are not apparent until several moves into the game, it frequently pays to let several layers of non-performing nodes build up before the entire branch is stripped.

4.1. Formalisation

The process of the removal of node x is simply: $N = N \setminus \{x\}$.

All connected activation links must be nulled.

$$\forall_{n \in N} l(x,n) = 0 \wedge l(n,x) = 0$$

And similarly history links:

Published in: NEURAL NETWORKS AND DATA MINING (1997) Eds. Gregory, P. et al Unicom, p181-209.

$$\forall_{n \in N} h(x,n) = 0 \wedge h(n,x) = 0$$

The criteria under which this node removal takes place is varied and functionally dependant. If the network is being used for variable prediction, such a criteria may be the Bayesian probability of a variable v given a node x being less than some predetermined threshold θ. For example $p(v|x) < \theta$.

5. Application - How could the data be used?

We have described how the above procedures can be used to generate networks of considerable complexity that model any deep statistical regularity within the data., and simplify the data through the identification of higher order elements. But how can these networks be used?

5.1.1. Net input

Using a back propagation type algorithm for training neural networks to segment or predict the behaviour of a population is an obvious application of neural network technology. However the size of the databases frequently prohibits this due to combinatorial explosion of linkages and consequent exponentially rising processing times. The above process can be used to alleviate this problem. If successful a network will be produced consisting of high order nodes from the data set. A particular member of the population can be characterised with these higher order nodes more economically and efficiently. Further by choosing an appropriate pruning function we can tune the data to the task in hand essentially performing useful pre-processing on the data.

5.1.2. Predictor

The generated node itself can be used as a source from which to predict the value of an independent variable, particularly if this variable is used within the pruning heuristic. We can of course analyse the Bayesian probability of this independent variable given the presence of the various atomic attributes. With a fully evolved network this initial estimate can be greatly enhanced.

5.1.3. Query for logical composition

The network discussed is localist, consequently the nodes may be interpreted symbolically. Also due to the similarity between composite and clustered nodes, and the logical notions of AND and OR respectively, the grown network may be interpreted in terms of logical rules. This means if a node has been identified which is particularly useful, as a criteria for classification for example, it is very easy to translate this network into a set of production rules.

5.1.4. Query for statistical distribution

Similarly, the symbolic nature of the network renders it ripe for statistical exploitation. The history linkages of the network contains detailed information concerning the contextual distribution of the individual nodes, in much the same way as Markov model [3] or N-gram analysis [4][5] would. However unlike these and similar statistical techniques were the scope of analysis is predetermined before processing takes place, the scope on analysis is dynamically and locally determined at each node.

A grown network serves as a rich source of statistical information. As a general rule if the existence of a particular node/attribute is an independent random event then the contextual history of that node will have a random distribution. Where the contextual history is not random there is useful information ready to exploit.

6. Conclusions

In general the three algorithms discussed above provide an evolutionary procedure for synthesising the optimum description of a data set, for use in a specified application. The two relationships presented, composition and clustering, although somewhat arbitrary are easy to model in a connectionist network and have a solid logical foundation. The iterative application of the network therefore leads to sophisticated models of considerable complexity. The third algorithm, the pruning process, provides the evolutionary forcing factor to ensure that the network has relevance to the application in hand.

A produced network and the activation links that exist between nodes therefore provides a perceptual framework for the original data set, both simplifying and tuning the data.

Specifically, however, the implementation of the above algorithms necessitates the use to the history linkages, which prove to be useful items in their own right. They provide a deep contextual history of each node, which can be used to divine the relationships that exist, not only between the original atomic attributes, but any clustered and composite attributes subsequently identified.

7. References

[1] Allott,N, Fazackerley, P, Halstead P (1995) .A Clustering Algorithm for Producing Context Rich Networks. In J. Taylor ed., Neural Networks and their Applications, pp220-29, Chichester: Wiley.

[2] Allott,N, Fazackerley, P, Halstead P (1997) Sequence Clustering Using Time Delay Networks, submitted ICANNGA conference.

[3] Jelinek F, Mercer R and Bahl (1983), 'Continuous Speech Recognition: Statistical Methods', IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol PAM-5.

[4] Brown P, Lee H and Spohrer, 'Bayesian Adaptation in Speech Recognition', Proceeding of the ICCASP, Boston, 761-764.

[5] Riseman E and Hanson A (1974), 'A Contextual Postprocessing System for Error Correction Using Binary N-Grams', IEEE Transactions on Computers, Vol C-23, 490-493.

[7]Rumelhart D.E. and McClelland J., Parallel Distributed Processing, Cambridge: MIT, 1968.

Published in: NEURAL NETWORKS AND DATA MINING (1997) Eds. Gregory, P. et al Unicom, p181-209.

AppendixL.

APPENDIX L

Published in: PROCEEDINGS OF AISB 94 (1994)

Published in: PROCEEDINGS OF AISB 94 (1994)

DESIGN ISSUES

Design Issues

The data flow strategy outlined on the diagrams on the previous page implies that the computer embody the language processing and knowledge processing tasks previously performed by the marker. Before marking can take place the marker must then impart the knowledge/rules etc. necessary to mark reasonably. In order to produce a system which both encourges this data exchange and can reasonably model the marking process certain design issues need to be resolved before implementation can proceeded.

- Scope: single sentence
- Segmentation of Knowledge Representation and Natural Language Processing functions
- Understanding vs evidential reasoning
- Contribution of Syntax

Scope

For reasons of simplicity we rescrict ourselves initially to looking at single sentence replies. The hope is that each reply will embody one fact only which will be easier to test against our knowledge base.

Segmentation

Knowledge Reprentation and Natural Language Processing are traditionally thought of as two distinct but closely associated disciplines. In this application they have proved difficult to seperate in that:

- The natural script is being used to build and test the knowledge structures.
- The lanuage processing is a knowledge intensive process.

Here the two have been embodied in the same simple but powerful underlying tree-like architecture. The distinction, if one is to be made, is one of degree rather than absolute. Knowledge pertinent to language processing, to do with the status and category of a word, is held low down in the tree structure. Knowledge of a higher level which embodies facts etc. is held high up in the tree.

Contribution of Syntax

It is widely assumed that syntactic parse is a necessary precursor to any semantic analysis that is to take place. It is not deemed necessary in this case for the following reasons.

- We are dealing with sentences in a highly constrained context in that they all answer the same question. Any meaning to be usefully extracted from the grammatical status of a word may be safely inferred from context.
- Grammatical analysis is an extremely difficult problem in its own right.
- Real world, untidy data is being handled; replies are ungrammatical and badly spelt - a reliable parse would be impossible.

Understanding vs Evidential Reasoning

It is our aim simply to discrimate between correct and incorrect responses to a question. To achieve this true *understanding* is not, we believe, necessary. In this NLP problem where the context of the reponses is highly constrained it is possible to generate a strong expectation of what will be submitted. The underlying strategy then is to expicitly set down these expectations within the representational scheme offered by the chosen knowledge base. The submitted student sentence is then proffered as a set of evidence that either confirms or refutes this expectation.

The strategy then serves to focus information on the pertinent information only. It carries with it, however, the disadvantage of being unable to perceive what it can not expect.

KNOWLEDGE BASE

Choice of Knowledge Architecture

The knowledge architecture opted for is similar in many ways to a semantic net, in that the raw constituents are nodes and links. One reason for this was that such structures can be clearly visualised graphically, and will make user interaction easier. The fundamental distinction between semantic nets and the chosen architecture is that links are not labelled. A relationship between two objects is instead represented by the two objects along with a third object, representing the relationship, all linking to a higher abstract node. This higher node serves to group the constituents together and provide an abstract representation of (or a shorthand for) the fact itself.

Node Characteristics

- Name (identifier)
- Activation value
- Threshold value
- List of parents (to which activation passed)
- List of children(from whom actication sent)
- List of evidnce(evidence nodes)

Node's Functional Capacities

Each unit is encorporated into a hierachical tree type structrure. Within nodes tend to operate in one of three capacities:

- Evidential
- Abstract
- Compound

Overall Structure

The knowledge is embodied in a nodal based, hierachically structured message passing structure. The decision making becomes implicit in the message passing aspect of the structure. So to judge a sentence each component word is presented to each evidence node. If the resulting flow of activation ends with the top level compound node becoming active the answer is deemed correct

Hierachical tree

Evidential

Evidential nodes are used to represent the low word level knowledge. Each has associated with it a list of words (evidence list) whose presence in the sentence will lead to the activation of the node. An evidential node is analogous to a lexeme.

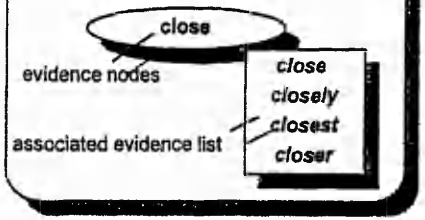
Abstract

An abstract node can be used to represent the abstract sense of several different words. It is then a disjunction of nodes and will become active if ANY of its children become active. In its simplest sense can be used to represent synonyms. (Logical OR).

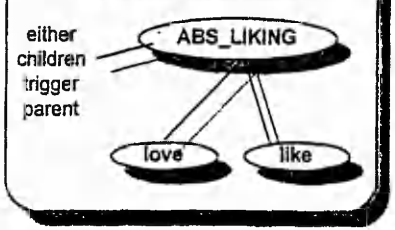
Compound

A compound node is used to represent the compound sense of several nodes. It is a conjuction of nodes and becomes active only if ALL of its children become active. A fact is represented with a compound type as a conjuction of abstract types. (Logical AND).

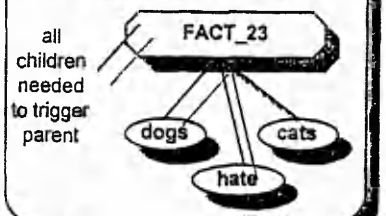
Evidence Node Example



Abstract Node Example



Compound Node Example



Published in: PROCEEDINGS OF AISB 94 (1994)

Published in: PROCEEDINGS OF AISB 94 (1994)

KNOWLEDGE FOR LANGUAGE

Requirements of Natural Language Processing

To feasibly process natural language our knowledge architecture must model or take into account certain properties of language:

- Synonymy: many words one meaning
- PolysmeyHomonymy: same words many meaning - needs context to disambigaute
- Idioms: meaning of two words not same as meaning of two component words
- Deep representation of meaning must be grammar independant
- Distinction between relationship and object

Synonymy

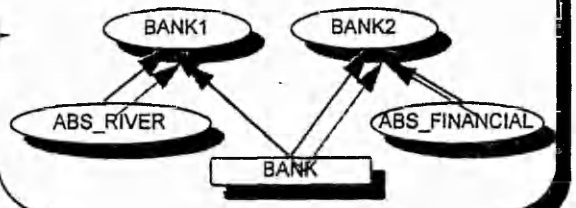
A synonym is represented simply with an abstract node. The abstract node then becomes active is any of its synonyms are found

Polysemy - Idioms

With polysemy(and/or homonymy) we need to disambiguate between many meanings. With an idiom we wish to overide the meaning generated by two nodes with the meaning implied by its compound. Both may be resolved by use of context which is handled in our knowledge architecture by the use of compund nodes.

Context Example

This demonstrates the use of context to disambiguate between two senses of the word bank. A sense only becomes active if the instance of the word occurs in the appropraite context ie. they are compound nodes requiring both children active.



Relation - Object Distinction

In the proposed K-Architecture no implicit distinction is made between an object and a relation between those objects. This has the following advantages:

- It is possible to express knowlegde about relationships in the same way knowedge about objects are expressed.
- We may recursively express knowledge about facts. Where a fact is a conjuction of nodes which map to a single compound node, this compound node may be put in a *relationship* with other nodes.
- We may explicitly represent distinciton between the type and token of a relationship as we may with objects.

Grammar Independance

We metioned earlier the inappropriateness of syntactical information for our present purposes. As a further example of this consider the following:

Romeo loved Julliette
Romeo felt great love for Julliette
Romeo felt lovingly towards Julliette
Romeo had a loving feeling for Julliette

The above all sentences have similar meanings however the key concpet *love* occurs as noun, verb, adverb and adjective. When we are attempting to crudely match the meaning of one sentence against another where the context of the sentence is heavily constrained, syntax and the issues it raises are best ignored, with little effect on the eventual performance of the system.

Published in: PROCEEDINGS OF AISB 94 (1994)

RESULTS

Experiments Run

Two tests were performed on the system

- The first set of data was initially hand marked. A knowledge base was then constructed which embodied all the qualities that the correct answers had in common. The answers were then applied to the network. Since we had the hand marked results also, we could correlate the automatic judgements against the manual and identify exceptions. With a series of stepwise refinements the knowledge base could be modified until a good model of the human marker's perfomance was reached.
- Secondly the resulting knowledge base was applied to a blind set of data (actually the reponses of the following years students). The set was then hand marked and coverage rates compared.

First Test

The first experiment was essentially a test of the expressive sophistication of the knowledge architecture. That is, was it possible to construct a structure which could discriminate between right and wrong answers? This run does not prove the system is usable as in normal scenarios we would not have the answers pre-hand marked.

However, it proved arbitarily easy to construct these stuctures with a coverage rates of 85%+ in very little time.

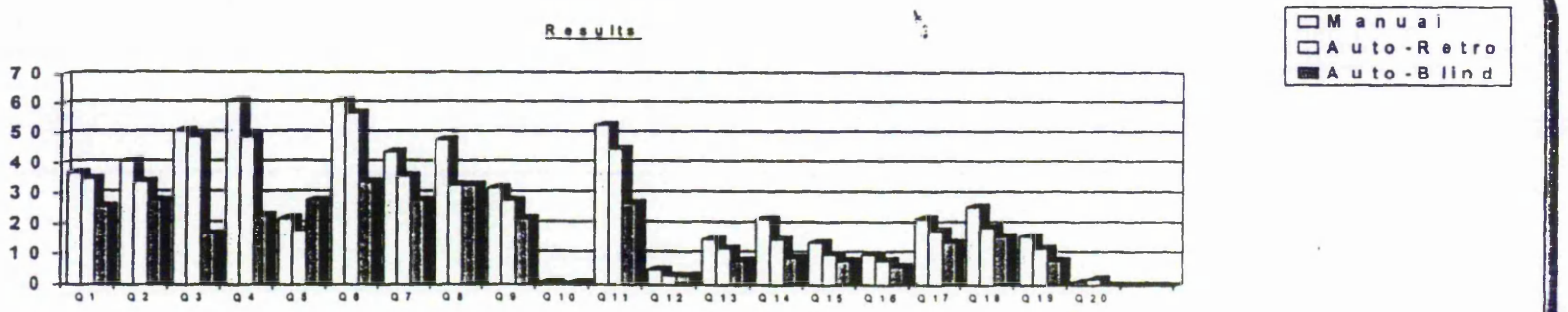
Second Test

When the knowledge base created in the previous phase was applied to a blind set of data an average coverage rate of around 60% was obtained. Considering the present limitations of the system this seems very encouraging.

General Comments

The automatic marker systematically marked fewer answers correct than the manual mark. It is also noticeable that very rarely does the sytem mark an incorrect response correct. We have in effect partially satisfactied of one of our initial criteria, that is the system has a measure of its own certainty (ie. high) when judging correct.

The major problem when trying to mark automatically is anticipating the full range of responses possible. We have partially overcome that here by insisting on two runs of system. One on a set of prototype information to help generate these anticipations and a second true run. The next logical step in reducing the time required to generate the knowledge base is to intergrate secondary knowledge sources (see next page).

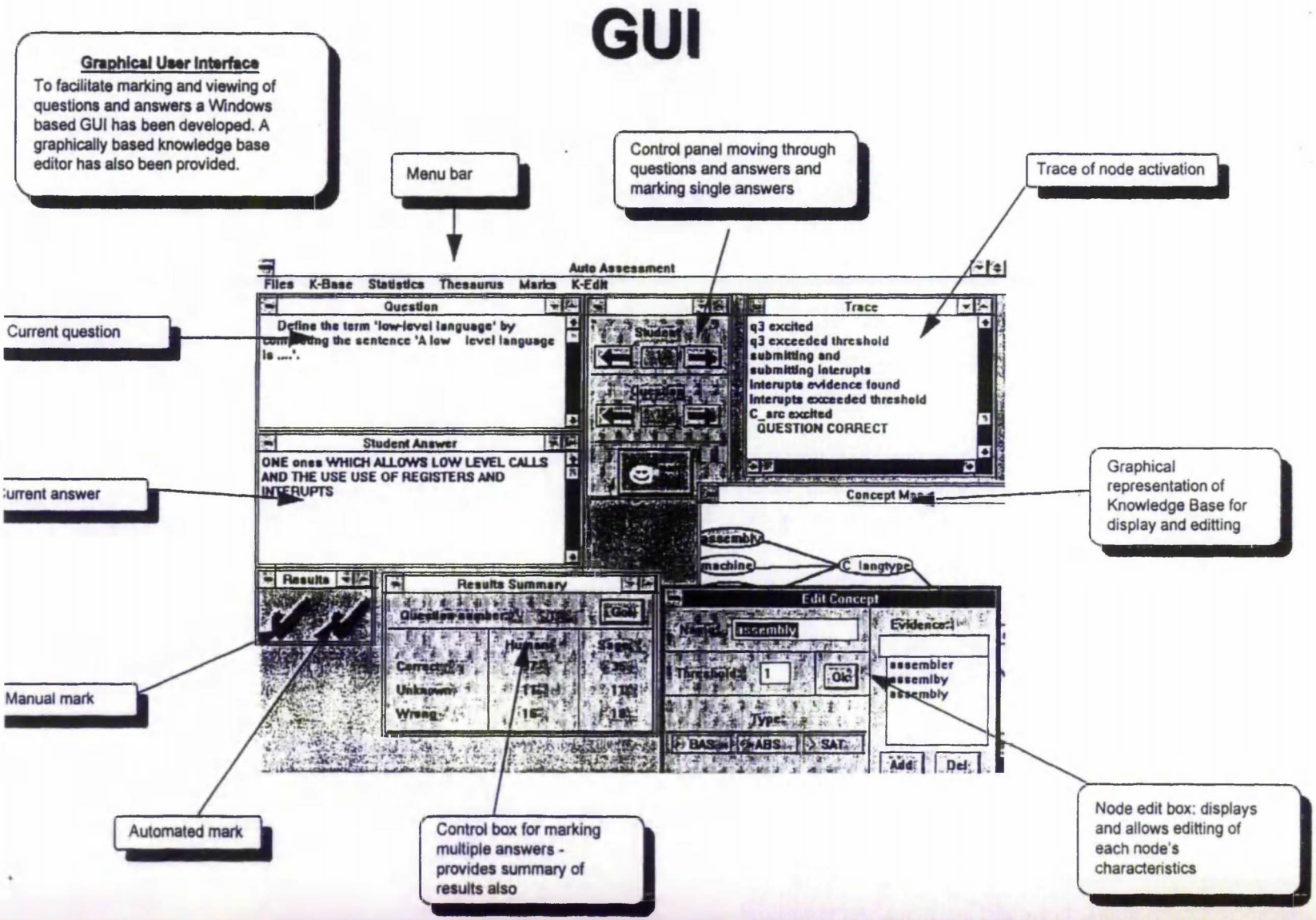


Published in: PROCEEDINGS OF AISB 94 (1994)

GUI

Graphical User Interface

To facilitate marking and viewing of questions and answers a Windows based GUI has been developed. A graphically based knowledge base editor has also been provided.



Published in: PROCEEDINGS OF AISB 94 (1994)

FURTHER WORK

Further Development

The follwing have been identified as possible areas for further development:

- Secondary information sources for Knowledge Base
- Enhancing the Knowledge Architecture
- Intergrating parallel processes under the same architecture

Running Parallel NLP Tasks Under the Same Architecture

Within the same underlying architecture it may be possible to implement many of the other natural language processing tasks, allowing techniques to be run in parallel, sharing the same information, and providing a mechanism for combing results.

- Syntactical analysis: a bottom up parse would be ideal for such an architecture. Further, contextual information is easily added (as in semantic) to augment its power. Variable node activation and mutually inhibitory connections would provide ambiguity resolution.
- Spelling correction and word recognition: by producing a similar architure for the sub word level (ie. disolving words into hierachically structured letter clusters) an information rich, contextually sensitive mechanism for word recovery may be provided. An algorithm for producing such sub-word strucures has already been developed.
- Semantic analysis: more theoretical semantic methods such as componential analysis may also be implemented within the same architecture.

Enhancing the architecture

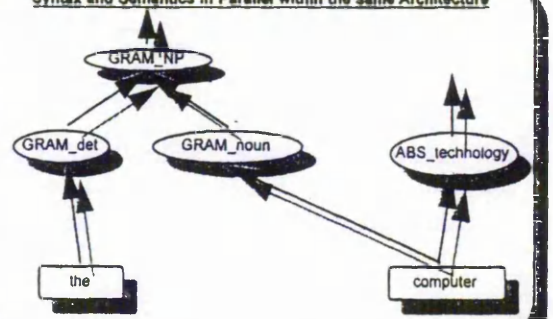
The present system is limited in that node activation and mesage passing is a boolean all or nothing event. Fuzzy data would be better processed by introducing some variability to the activation and hence the decision making processes. Also message passing may be enhanced by allowing messages to be passed down as well as across the network. This would give stronger ambiguity resolution via a winner take all strategy implemented by mutually inhibitory connections between sibling nodes.

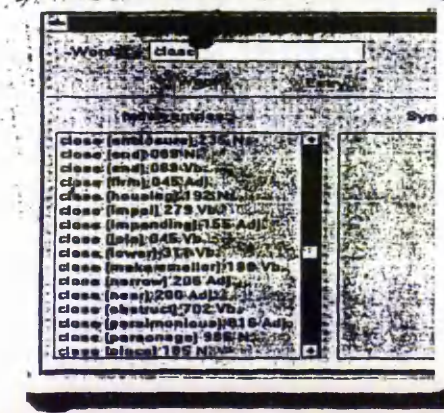
Introducing Secondary Information Sources

In order to reduce the considerable overhead involved in generating the knowledge base and to improve the performance by utilising infomation about language variability secondary information sources need be intergrated into the system. There are two immediate sources that have been identified:

- Thesuari
- Lexeme -Word maps

Syntax and Semantics in Parallel within the same Architecture





Thesuari

Thesauri essentially give us synonyms ie. words that mean the same thing. This is what abstract nodes within the knowledge base most frequenty embody. A version of Rogets thesaurus has already been implemented within the GUI as an aid to the marker. It is a relatively simple step to embed the information produced this way automatically into the network.

WordNet has also been identified as a possible supplier of such information. It has a richer hierachical description of word meanings than conventional thesuari

Word-Lexeme Maps

Associated work within the department has produced extensive word-lexeme maps. This is precisely what is needed to produce the evidence lists required by the evidence nodes. This will reduce the Knowledge Base construction phase by another order of magnitude.