Large-Scale Connectionist Natural Language Parsing Using Lexical Semantic and Syntactic Knowledge

Dianabasi Edet Nkantah

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

Doctor of Philosophy

October 2007

ABSTRACT

Syntactic parsing plays a pivotal role in most automatic natural language processing systems.

The research project presented in this dissertation has focused on two main characteristics of connectionist models for natural language processing: their adaptability to different tagging conventions, and their ability to use multiple linguistic constraints in parallel during sentence processing. In focusing on these key characteristics, an existing hybrid connectionist, shift-reduce corpus-based parsing model has been modified.

This parser, which had earlier been trained to acquire linguistic knowledge from the Lancaster Parsed Corpus, has been adapted to learn linguistic knowledge from the Wall Street Journal Corpus. This adaptation is a novel demonstration that this connectionist parser, and by extension, other similar connectionist models, is able to adapt to more than one syntactic tagging convention; this implies their ability to adapt to the underlying linguistic theories used to annotate these corpora.

The parser has also been adapted to integrate shallow lexical semantic information with syntactic information for full syntactic parsing. This approach was used to investigate the effect of shallow lexical semantic information on full syntactic parsing.

In pursuing the aims of this project, a novel algorithm for semantic tagging of nouns in the Wall Street Journal Corpus has been developed. The lexical semantic information used in this semantic annotation algorithm was extracted from WordNet, an online lexical resource.

Using only syntactic information in making parsing decisions, this parsing model was tested on test sets of sentences that were not used during training. The parser generalised to parse these test sentences with an F-measure of 72.5% and 59.5% on sentences from the Lancaster Parsed Corpus and Wall Street Journal Corpus, respectively. On the integration of shallow lexical semantic information with syntactic information in its input representation, the parser generalised to parse test sentences from the Wall Street Journal Corpus with an F-measure of 56.75%. Although this integration did not seem to improve the parser's overall training/generalisation performance, given its present configuration, it did appear to improve the parser's decision making concerning preposition phrase attachment.

Acknowledgements

My deep appreciation goes to my supervisors, Jon Tepper, Heather Powell, and Nasser Sherkat for their immense contribution towards the success of my PhD programme. Your understanding, encouragement, guidance and patience have been invaluable to me. Special thanks also to Tony Allen for his kind words of advice and encouragement.

Many thanks to Doreen Corlett, Julie Bradshaw, Mel Borland and Collette Jackson for the splendid administrative support received from you during the course of my PhD programme.

I would like to thank my father, Obong Edet Nkantah, mother-in-law, Mrs Comfort Inyang and my entire family for words of encouragement, advice and prayers.

I would also like to thank my friends and colleagues at Nottingham Trent University, Life at the Centre and Oakdene, who showed concern and encouraged me in various ways. Special thanks go to Christine Roach for her support and encouragement.

I appreciate the great friendship, support and encouragement given to me and my family by Boniface, Tikha, Steve and Anapiri. Thanks also to Akan Obot for encouraging me to embark on this project.

I am very grateful to my wife, Idorenyin, and our sons, Akanimo and Enobong, for the constant support, encouragement, and love you have showered on me. You have always shared my concerns and anxiety. You have been my inspiration.

To God be the glory. His grace and favour have seen me through this journey.

TABLE OF CONTENTS

List of Figures

List of Tables

1. Introduction

1.1 Background

Automatic natural language understanding systems are built with the purpose of getting them to generate and/or interpret natural language. They have been used in language tasks such as machine translation, information retrieval, human-machine interfaces, text analysis, corpora analyses and knowledge acquisition. At the centre of most, if not all, of such systems is syntactic analysis [1, 2], or parsing [3, 4]. Syntactic parsing is the process of identifying and representing the structural relationships between words with respect to phrases, clauses and sentences. It is an integral part of accurately interpreting a sentence. Given the complex, ambiguous and potentially unbounded nature of natural language, the ability to parse realistic subsets of unconstrained natural language is a significant problem for practical natural language-based systems and progress has been limited. Other than in automatic natural language understanding systems, parsers have also been employed in simpler and more constrained problem domains such as compiler construction, database interfaces, document preparation and conversion, typesetting chemical formulae and chromosome recognition, to mention but a few.

To date, traditional statistical parsing models [5, 6, 7, 121] continue to represent the state-of-the-art for broad coverage natural language parsing, achieving, at best, an accuracy rate of 90% for sentences of 40 words or less. They implement the parsing process by estimating parse probabilities from pre-parsed corpora. The level and quality of improvements reported for such methods are decreasing every year [5, 6, 7, 121]. Besides, these models lack an in-built ability to combine multiple types of linguistic information (semantics, pragmatics, discourse, etc) in syntactic analysis, due to inherent coupling with symbolic representation of linguistic information. This reduces their usefulness for practical language

applications, such as information extraction and knowledge–base inferencing via query/answering interfaces, which require semantically-aided processing. In contrast to statistical models, the connectionist (also known as the artificial neural network-based, parallel distributed processing, or subsymbolic) approach [8, 9, 122] offers inherent robust representations that are able to naturally combine, inter alia, syntactic and semantic information.

It is widely accepted that semantic information is needed to resolve common syntactic ambiguities [120]. However, a lot of parsing models do not attempt to incorporate this information into initial syntactic parsing. They typically adopt the two-stage 'Fodorian' approach [10, 11, 12] whereby semantic information is considered along with other linguistic information during a second independent post-processing stage. The first stage of the process considers syntactic information alone.

A growing body of research, however, refutes the two-stage model and advocates a multiple constraint-based approach. These researchers argue that the human sentence processor (HSP) does not only use syntactic information during sentence processing, but is a multiple constraint-satisfaction process that allows syntactic, semantic, pragmatic and discourse information to simultaneously interact (to varying degrees) during on-line processing [13, 14, 15, 16, 22].

It is, therefore, necessary to determine the effect of integrating additional word-level (or lexical) semantic information with syntactic information on the performance of full syntactic parsers.

1.2 Aims and Objectives of Research

This research is based on an existing modular/hybrid, shift-reduce, connectionist parser [18, 19, 111], which integrates three connectionist modules (two temporal sequence processing modules and a phrase structure recognition module) with three symbolic modules to automatically learn syntactic structure from a subset of the Lancaster Parsed Corpus [20]. The parser has shown good learning ability by achieving an average parsing accuracy of 73% on a test set of sentences that were not used during its training. Considering its connectionist nature, the method is adaptable to other corpora and for other syntactic and semantic annotations without its architecture being changed. In order to improve the natural language acquisition capability of this parser, and other connectionist parsers, a number of research questions must be addressed:

a) Can this parser's (and by extension, other similar connectionist models') ability to acquire linguistic knowledge from more than one corpus (with different tagging conventions) be demonstrated?
b) What level of lexical semantic information will provide improvement to full syntactic parsing?
c) What lexical resources are available for the extraction of lexical semantic information?
d) How will the lexical semantic information be extracted and how will this information be integrated with syntactic information in the process of full syntactic parsing?
e) What is the effect of the integration of shallow lexical semantic information with syntactic information on the performance and behaviour of the parser?

Including lexical semantic information during syntactic parsing could help to resolve common syntactic ambiguities and preposition phrase attachment cases in sentences such as:

i. *The boy ate the pasta with the sauce.*

ii. *The boy ate the pasta with the fork.*

The aim of this dissertation is to investigate whether additional word-level semantic information can improve decisions in full syntactic parsing (involving syntactic ambiguity resolution). The investigation will determine the level of semantic abstraction necessary for improvement to occur. The sensitivity of the system to the type of input representation used would also be determined. This project also aims to investigate the re-usability and adaptability of this parser to other corpora.

In pursuing the aims of this research, investigations designed to specifically improve the performance of the temporal sequence processing modules of the existing parser were carried out. These modules form the bedrock of the parser as they tackle the most challenging aspects of natural language processing: the sequential nature of language and the existence of dependencies (sometimes, long-distance) between words in sentences. The re-usability and adaptability of this connectionist parser was investigated by adapting it for the internationally accepted benchmark corpus, The Wall Street Journal Corpus [101].

In further pursuing the aims of this research, there was the need to develop an algorithm for abstract word sense tagging of nouns. The implementation of this algorithm resulted in the extraction of lexical semantic information for nouns/pronouns from the online lexical resource, WordNet [17]. While maintaining the neural network and corpus-based nature of the parsing model, the lexical semantic representation realised from the algorithm was integrated into the existing syntactic representation already developed for sentences in the Wall Street Journal Corpus. That is, rather than expecting the parser to learn syntactic structure from sequences of part-of-speech (POS) word tags (e.g. noun, verb), the

parser will be expected to deduce syntactic structure from sequences of linguistically richer word tags containing syntactic and semantic information. This demonstrates the inherent ability of the parser to combine multiple types of linguistic information. It also helps determine to what extent non-syntactic information plays a role during the syntactic parsing process.

1.3 Structure of the Thesis

Chapter 2 of this dissertation presents a literature survey of the field of syntactic parsing. Different types of connectionist parsing models are reviewed in this chapter. Chapter 3 reports the aims, process and outcome of investigations carried out to optimise the temporal sequence processing modules of the original parser. In chapter 4, the processes and results of adapting the original parser to the Wall Street Journal Corpus are presented. Chapter 5 looks at the processes and results involved in combining lexical semantic representation with the syntactic representation of the parsing model. The sentence level performance of the parser on the Wall Street Journal Corpus is looked at in detail in chapter 6. This chapter also presents the effects of integrating lexical semantic representation into the syntactic representation of the parsing model. In chapter 7, the dissertation is concluded with a presentation of the key contributions of this research and further work that have arisen from it.

2. Literature Survey

2.1 Introduction

Natural language, such as English, French, Russian and Japanese, is fundamental to human cognition and culture. It serves as the main medium by which humans communicate and record information. Used as text or speech, it wields enormous power and influence in our lives. Getting computers to automatically process and understand natural language enables them to capture, to some extent, this power and influence.

Natural language understanding systems are built with the aim of making them generate and/or interpret natural language. Such systems have been employed in tasks such as machine translation, information retrieval, human-machine interfaces, text analysis, and knowledge acquisition. Language analysis is an important aspect of these systems. This could take the form of sentence analysis, which involves the processing of individual sentences, or discourse and dialog structure analysis, which involves the processing of a group of sentences. Analysing discourse structure would still require sentence analysis. A crucial component of sentence analysis is syntactic parsing.

This chapter presents syntactic parsing and its role in natural language understanding in section 2.2. Different parsing methods and the different approaches by which these methods are implemented are also treated in this section. Localist, distributed, and hybrid and modular parsing models are reviewed in sections 2.3, 2.4, and 2.5 respectively. In reviewing these models, a critical look is taken at the complexity of language structure they can handle, their ability to automatically learn, and their scalability to realistic language subsets. Attention is also paid to their representational capacity and their ability to combine other types

of linguistic information (like semantic constraints) into parsing. Various semantic annotation schemes are reviewed in section 2.6.

2.2 Syntactic Parsing

2.2.1 Introduction

Syntactic analysis or parsing [3] is the process of producing the structural description of a sentence. This is done with a view to recognising the structural relationships between words with respect to phrases, clauses and sentences. It plays a major role in accurately interpreting a sentence and is at the centre of most automatic language processing systems [1].

Automatic natural language processing systems need to represent language and often look to hypothesised representations of the brain. The mental representation of language and how it translates to text and speech therefore becomes an issue. The question of how much prior knowledge should be built into parsing (and other NLP) systems comes to the fore. In this respect, two main approaches have dominated in recent years. These are the *rationalist* and *empiricist* approaches [21].

The rationalist approach to language processing has dominated the field largely due to the work of Noam Chomsky [23, 24, 25, 26, 27]. It places the focus of analysis of natural language on the intuition of the native speaker. It is of the view that a major part of the knowledge in the human mind is fixed early in life, possibly at birth, and not derived by the sense organs. This view of the knowledge in the human mind extends to natural language and suggests that significant parts of language are fixed in the brain at birth. This suggestion by Chomsky stems from the difficulty he finds in envisioning how children can learn natural language,

considering its complexity, from the limited input that their senses pick up during their early years; the problem of the "poverty of the stimulus" [27].

The rationalist approach, also known as generative (or Chomskyan) linguistics [2, 21] therefore seeks to study the abstract mental structures that form a basis for linguistic ability (referred to by Chomsky [27] as I-language – "internal" language) while not considering actual mental processes (referred to by Chomsky [27] as E-language – "external" language) such as text or recording of utterances. In championing this approach, Chomsky [26] distinguishes between linguistic competence and linguistic performance. Linguistic competence is an abstract characterisation of the knowledge of language structure that is assumed to be in the native speaker's mind while linguistic performance refers to the processes that actually determine what a language user will say (or write) or how he will understand an utterance (or text) given a particular context.

The empiricist approach to language processing (also known as structural linguistics [2]) bases linguistic analysis on the observation of language behaviour. While agreeing with the rationalist approach on the presence of some initial knowledge structure in the human brain at birth, this approach, however, disagrees with the level of knowledge present. It assumes that the structure of knowledge available in the human mind at birth is of a general form, catering for activities such as pattern recognition, generalisation and association, rather than being of a detailed form, as espoused by proponents of the rationalist approach. This approach further suggests that the detailed structure of natural language is learnt by children when they combine the general structure of knowledge with the sensory input they are exposed to. The empiricist approach therefore subscribes to the description of the actual use of language (E-language) in linguistic studies. This is done with the use of a collected corpus of naturally occurring text (or utterance). These corpora are assumed to be representative of language in a real world context.

In order to understand the problem of syntactic parsing, a brief review of the basic concepts of formal language theory and phrase-structure grammar [24] is necessary.

A language is a set of sentences. This set could be finite or infinite depending on the language. Natural languages have infinite sets. A sentence is a string of one or more symbols (words) from the vocabulary of the language. A grammar is a finite and formal specification of a language. A widely adopted method to specify formal and natural languages is the use of the phrase-structure grammar (also known as production grammar [28]).

A phrase-structure grammar is described by four parts: a set of non-terminal symbols (the non-terminal vocabulary consisting of syntactic category labels and used in specifying the grammar); a set of terminal symbols (the terminal vocabulary of the language being defined); a special member of the set of non-terminal symbols designated as the start symbol of the grammar; and the production set of the grammar (set of re-write rules). The re-write rules are used for the basic operation of a phrase-structure grammar which involves rewriting a string of symbols as another.

Mathematically, a phrase-structure grammar, G, is an ordered quadruple of the form:

$$\mathbf{G = (V_N, V_T, S, P)}$$

Where

$\mathbf{V_N}$ = non-terminal vocabulary of G

$\mathbf{V_T}$ = terminal vocabulary of G

S = Starting symbol of G

P = Production set of G

Phrase-structure grammars may be classified according to their descriptive power. This considers the variety of languages a grammar can be used to define. More powerful grammars can be used to define and describe a wider variety of languages than weaker ones. This descriptive power (level of language that can be described by a particular grammar type) corresponds to the type of automata that can process it. Automata are abstract mathematical models of machines that perform computations on an input by moving through a series of states or configurations. For a parser to process a particular type of language, it must therefore simulate or adopt the computational properties of the appropriate type of automata.

The conventional classification scheme for phrase-structure grammars is the *Chomsky hierarchy* [23, 25]. This scheme identifies four types of phrase-structure grammar in order of their descriptive power. They are: unrestricted (type 0), context-sensitive (type 1), context-free (type 2) and finite-state or regular grammars (type 3). Higher numbered types are less powerful (more constrained) than lower numbered types; type 0 is the most powerful type, and type 3 is the weakest.

Type 0 grammars describe languages that are recursively enumerable. This is a type of language for which a program could be written to list out the sentences of the language one after the other. They have productions of the form:

$$\alpha \longrightarrow \beta$$

where α and β denote strings of terminals and non-terminals

There are no restrictions on their productions.

Type 0 grammars have equivalent computational power to Turing machines (TM) [29]. A Turing machine is an abstract machine (or, computer) introduced by Alan Turing to give a mathematically precise definition of algorithm or mechanical

procedure. It is the most general automaton that assumes an infinite memory capacity.

Type 1 grammars describe context-sensitive languages and have their production sets constrained such that the right hand side (RHS) of each re-write rule has, at least, the same number of symbols as its left hand side (LHS). Each rule specifies the replacement of only one non-terminal in its LHS. Re-writing symbols depend on context as different rules may re-write a particular non-terminal symbol to different values depending on its surrounding symbols. Their productions are of the form:

$$\alpha_1 A \alpha_2 \longrightarrow \alpha_1 \beta \alpha_2$$

where **A** is a non-terminal

and α_1, α_2, and β are strings of terminals and non-terminals

This type of grammars has equivalent computational power to linear bounded automata (LBA). An LBA is a restricted form (in terms of tape length or *memory*) of a Turing machine; it consists of a tape with cells that can contain symbols from a finite alphabet, a head that can read from or write to one cell on the tape at a time and can be moved, and a finite number of states.

Type 2 grammars describe context-free languages and have their production sets restricted such that the LHS of each re-write rule is a single non-terminal symbol while its RHS is a string of one or more terminals and non-terminals. Their production rules are of the form:

$$A \longrightarrow \beta$$

Where **A** is a non-terminal

and β is a string of terminals and non-terminals

This type of grammars has equivalent computational power to *push-down automata (PDA)*. A push-down automaton is equivalent to a finite state automaton (FSA) with

a stack-like memory. A finite state automaton is a finite collection of states and transitions, with certain states designated as start and end states. The addition of a stack-like memory enables states to be stored whilst intermediate states are being processed. While context-free grammars alone are inadequate in describing natural languages, they can be extended with complex linguistic categories to do so. However, the number of categories must not be restricted [30]. The resulting grammar from this type of extension is referred to as an index grammar. Index grammars have equivalent computational power to nested stack automata [31]; their descriptive power is greater than that of context-free grammars but less than that of context-sensitive grammars.

Type 3 grammars describe regular languages. They have their production sets constrained such that the LHS of each re-write rule is a single non-terminal symbol while its RHS is either a single terminal symbol or a terminal symbol and a non-terminal symbol. Their productions are of the form:

$\mathbf{A} \longrightarrow \mathbf{a}$, or, $\mathbf{A} \longrightarrow \mathbf{aB}$

Where **A** and **B** are non-terminals

and **a** is a terminal

This type of grammars has equivalent computational power to finite state automata (FSA).

Several grammatical frameworks have been employed in the syntactic analysis and generation of natural languages. Among these are generative grammars which are the most traditionally used grammatical framework in natural language processing systems. This group of linguistic formalisms include Transformational Grammar (TG) [26], Government Binding Theory (GB) [136], and Generalised Phrase Structure Grammar (GPSG) [137, 139]. Both TG and GB theory consist of deep and surface syntactic structures. GPSG, on the other hand, consists of only surface

structure. GPSG augments phrase structure grammar such that linguistic constructions beyond the reach of phrase structure grammar can be handled.

Also within the generative grammar ambit are lexicalised frameworks which encode syntactic and semantic information in the lexicon. This group of frameworks also produce parse trees that comprise lexical items and direct relationships between them. They include Lexical Functional Grammar (LFG) [138], Head-Driven Phrase Structure Grammar (HPSG) [140], Dependency Grammar [141], Categorial Grammar [142], Lexicalised Tree-adjoining Grammar (LTAG) [143].

Besides generative grammar, there are semantic-based grammars which seek to create semantic representation of sentences. This group of grammatical framework include Semantic Grammar [146] and Case Grammars [144, 145]. Aside from generative and semantic-based grammars, there are stochastic grammar such as Probabilistic Phrase Structure Grammar, and functional grammar such as Role and Reference Grammar [148].

A key problem that designers of syntactic parsing systems for natural language have to contend with is ambiguity. Ambiguity increases the range of possible parse trees for a given sentence. There are various types of ambiguity such as lexical ambiguity, structural ambiguity, and referential ambiguity. Lexical (or categorical) ambiguity arises when a word in a sentence can be assigned to more than one syntactic category depending on its linguistic context. It is usually resolved at the tagging phase, where input to a parser consists of syntactic tags.

Structural ambiguity could be local or global. It is local ambiguity when part (and not the whole) of a sentence such as a phrase can be assigned to various structures and meanings if taken out of context. An example is: *The man who accompanied the lady paid the bill*. Here, the phrase, *the lady paid the bill* has a meaning that is

different from that of the whole sentence. Global ambiguity arises when a whole sentence has more than one possible interpretation. An example is: *Visiting lecturers can be expensive*.

Referential ambiguity arises when more than one object is being referred to by a noun phrase. An example is: *When they had finished writing their test, the students and lecturers left*. Here, "*they*" could refer to only the students, only the lecturers or both groups.

2.2.2 Role of parsing in Natural Language Understanding

Syntactic parsing is an essential part of the process of understanding natural language texts. In assigning tree structures to sentences in a text, parsing identifies the roles of words in a sentence. A parse of the following sentences, *James bit Jane* and *Jane bit James*, would identify the noun phrase/subject and enable the understanding of who did what to whom in each case. Besides, parsing highlights the structural relationships between words and phrases in a sentence. A parse of the following sentences, *Visiting lecturers ARE exciting* and *Visiting lecturers IS exciting*, would identify the subject – verb agreement and enable an understanding of the different meanings conveyed by each of these two sentences. Parsing can also identify relationship between words in neighbouring sentences.

However, most successful natural language analyses do not consider syntax alone. Other aspects of analysis considered include semantics, pragmatics and discourse integration. In adopting these components for the realisation of successful syntactic parsing systems, psycholinguistic researchers have used two different approaches: treating parsing as a two-stage model [10, 11, 12] and as a multiple constraint-based model [13, 14, 15, 16, 22].

The two-stage parsing model (also known as the Garden-Path or "Fodorian" model) considers parsing to be a two-stage process with syntactic information playing a crucial role in the first stage. The second stage, which acts as a post-processing stage independent of the first, uses other linguistic information (semantics, pragmatics, discourse, etc) to evaluate and possibly revise the analysis done at the first stage. Any structural ambiguity that arose at the first stage is likely to be resolved at the second stage.

Multiple constraint-based models implement parsing by allowing various components of linguistic information (syntax, semantics, pragmatics, discourse, etc) to interact simultaneously during online processing. They restrict the use of particular components of linguistic information in sentence processing and allow for parallel evaluation of alternative syntactic analyses.

2.2.3 Parsing Methods

The two main methods of parsing are top-down and bottom-up parsing. Top-down parsers construct parse trees by working from the start symbol (the root of the parse tree) to the terminals (the leaves of the parse tree) that make up the input sentence. Bottom-up parsers construct parse trees, beginning from the terminals that make up the input sentence and work up to the start symbol. Apart from this main classification, parsing techniques can also be grouped based on directionality [32].

Focusing on the directionality classification, parsing techniques can be directional or non-directional. Non-directional parsers construct parse trees by processing the terminals in the input string in an arbitrary order. This method needs the entire input to be in working memory before parsing can commence. An example of a

non-directional parser is the Unger parser [33]. The Unger parser can also be classified as a top-down parser. Another non-directional parser, which is also a bottom-up parser, is the CYK parser [34, 35, 36, 37].

Directional parsers construct parse trees by processing the terminals in the input string from left to right (or, from right to left). With this method, and as with the human sentence processor (HSP), the entire input does not need to be in working memory as parsing can commence, and progress, before the last symbol (or the first symbol if parsing from right to left) in the input string is seen. This group of parsers could be further grouped into non-deterministic and deterministic parsers. Non-deterministic directional parsers often have several moves to choose from, in their bid to solve parsing problems, with the particular choice not being predetermined. Search for the solution could either be depth-first or breadth-first. Recursive descent parsers, which are top-down, depth-first parsers, are examples of non-deterministic directional parsers. Other examples are Earley parsers [38] and Tomita parsers [39, 40] which are both bottom-up parsers that employ the breadth-first search technique.

Deterministic directional parsers are restricted to one possible move in each decision case, while solving parsing problems. The moves to be made are determined by the input string. An example of this group of parsers is the LL(k) (**L**eft-to-right, "identifying the **L**eft-most production"; k is the number of look-ahead symbols) parser [41], which is also a top-down parser. Examples of bottom-up parsers which belong to this group are LR(k) [42], LALR [43, 44], and SLR (Simple LR) parsers.

All the parsing methods treated above could be classified as top-down or bottom-up. However, a method of parsing that can not be classified into one of these two main groups, because it is a hybrid between them, is left-corner parsing [45]. Left-

corner parsers have the right-hand-side of each production rule split into two parts, the left part (called the left corner) and the right part. The left corner is identified with a bottom-up method. When the left-corner has been identified, the right part is then parsed with a top-down method.

In implementing these methods of parsing, symbolic, statistical and connectionist approaches have variously been employed. Hybrids of these approaches have also been used.

2.2.4 Connectionist Parsing

Connectionist (or, artificial neural network-based) parsing systems make use of parallel distributed processors which consist of simple processing units that interact to acquire and store linguistic knowledge and make this knowledge available for solving parsing problems. These networks are presented with representations of sentence examples, from which linguistic knowledge is acquired through a learning process. The knowledge acquired is stored in the networks' synaptic weights, which link the simple processing units (or, nodes) together.

Knowledge representation in connectionist networks could either be localist or distributed. Localist networks are designed in such a way that individual units denote particular concepts or features. These individual units are clearly labelled making their roles in such networks obvious. However, information is not shared among the different components of the network, creating inefficiency in terms of connections and nodes. Distributed networks are designed in such a way that concepts or features are denoted as patterns of activation distributed across several units in the network. They exhibit a high fault-tolerance as the loss of one or more units may not necessarily lead to the network losing all of its representation of a particular concept.

2.3 Localist Parsing Models

Localist models were the earliest connectionist attempts at parsing. Parsing models developed by Small [46], Cottrell [48] (an extension of word-sense disambiguation work done by Small, Cottrell, and Shastri [47] and implemented by Cottrell [49]), Howells [50], and Waltz and Pollack [51] are multiple constraint based localist models that allow syntactic and lexical-semantic constraints to interact in attempts to solve parsing problems. These models, apart from [50] and [51], accommodate only fixed-length sentences and, therefore find it difficult to deal with recursive, context-free structure which is likely to lead to long sentences.

Fanty [52] and Rager's [53] models are localist models that implement the CYK parser [34, 35, 36, 37], while Selman and Hirst's [54, 55] model implements a variation of the Boltzman machine [56]. These models can only deal with fixed-length sentences; they therefore rely on redundant structure. This reduces the complexity of language structure they can handle. They, also, do not incorporate other types of linguistic information into parsing.

Charniak and Santos' parsing model [57] uses a sliding input window on a localist network. This makes it able to process sentences of unbounded length. However, it is unable to process long-distance dependencies. Generally, localist networks manifest difficulty in functioning as language processors because of the manner of their input representation [9, 109, 110].

2.4 Distributed Parsing Models

Distributed parsing models exhibit the fault-tolerance associated with neural networks that use distributed representations. Early distributed models employed in language processing [58, 59, 60] used feed-forward multi-layer perceptrons (FF-

MLP) architectures for their networks. These networks were mostly trained with Back-propagation algorithm. These early models were able to learn and, given their distributed representation, had the potential to combine several types of linguistic information into their input vectors. However, these models were limited by fixed length restrictions on their input vectors. In designing these models, maximum sentence lengths had to be pre-determined so that the number of input units could be fixed. This meant that apart from the redundancy that would result in processing sentences of lengths lower than the maximum, sentences with lengths above the maximum could not be processed. These models were, therefore, not well equipped for linguistic inputs which require sequential processing.

In a bid to erase the limitation from the fixed inputs, several distributed parsing models [61, 62, 63] adopted sliding input windows. With these windows, a fixed number of sentence tokens were presented to the networks per time step, instead of presenting whole sentences at once. This ensured that sentence lengths were not restricted. However, input window sizes limited temporal context and, therefore, restrained the disambiguation capability of the models.

In view of the temporal sequence processing needs of language, language processing models [18, 19, 64, 65] have increasingly turned to recurrent neural networks [66, 67, 68, 69]. A modified version of backpropagation, referred to as Backpropagation Through Time (BPTT) [104], is commonly used to train recurrent networks to better learn temporal dependencies using gradient-based information. Although marginal improvements have been reported, these are very limited due to gradient information about previous input items diminishing rapidly as sequence length increases [105]. It has also been recently reported that although BPTT may not be suitable for learning complex temporal problems, recurrent neural network architectures are themselves capable of representing the solution [106] – a more effective learning algorithm is required to determine the optimum weight values.

Subsequently, there is much research on either improving gradient descent learning [107, 108] or searching for alternative learning algorithms.

2.5 Hybrid and Modular Parsing Models

To contend with the complexity of processing natural language, connectionist parsers have had modularity and hybridity introduced into them. The modularity feature in connectionist parsers involves breaking the parsing problem into simpler tasks and employing specialist modules (some of which may be non-connectionist) to solve these tasks. In doing this, the learning task is simplified and different connectionist network architectures (e.g. Feed forward Multilayer Perceptron (FF-MLP), Simple Recurrent Network (SRN) [66] and Recursive Auto-associative Memory (RAAM) [70]) are used to their strengths. Modular connectionist parsing models could be pure (with all the modules being connectionist) or hybrid connectionist parsers.

Hybrid connectionist parsers involve the combination of connectionist and non-connectionist (like symbolic, statistical) modules in a parser. Symbolic modules have generally been employed in such parsers to provide storage, symbol manipulation and control [8]. The storage provided by the symbolic modules could be made to temporarily hold input states, intermediate parse states and full sentential parses. They could also be made to permanently hold structured knowledge about the language being processed.

Modular/hybrid connectionist networks [18, 63, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89] have exhibited a better ability to learn than single network parsing models. They have also exhibited a greater potential to handle more complex language structure and continue to make progress [18, 72, 81].

The parsing model used in this work [18, 19] is a modular/hybrid, shift-reduce parser that integrates three connectionist modules with three symbolic modules to automatically learn syntactic structure from a subset of the Lancaster Parsed Corpus [20]. The process of parsing employed by this model involves two stages: delimitation of phrases and recognition of phrase structure. The phrase delimitation process is further broken down into two sub-processes: right-to-left delimiter (RLD) process and left-to-right delimiter (LRD) process. Each of these two delimiter sub-processes is implemented with the Temporal Auto-associative Simple Recurrent Network (TASRN) [68]. The phrase structure recognition (PSR) process is implemented with a feed-forward Multilayer Perceptron (FF-MLP). The three symbolic modules in this parser are used to store tag and parse state information, the resulting parse tree and the current input state. The parser has shown good learning ability by achieving an average labelled precision/recall of 72.5% on a test set of sentences from the Lancaster Parsed Corpus that were not used during training. It is also adaptable to other corpora and for other syntactic and semantic annotations without biasing its architecture; in this work, it has been adapted to the BLLIP 1987-89 WSJ Corpus [90] and its input representation has also been further adapted to include semantic information.

2.6 Semantic Annotation Schemes

Reported work that made use of WordNet have been reviewed with the aim of noting how the WordNet taxonomy is applied. Paul Buitelaar and his colleagues [124] describe an unsupervised semantic tagger, applied to German, but which could be used with any language for which a corresponding "XNet" (WordNet, Germanet, etc), POS tagger and morphological analyzer are available. Their system treats all synsets and their hypernyms as semantic classes to which a word may belong. The evaluation corpus used was manually annotated by two annotators with

differences in annotation solved by arbitration. In cases where annotators were unable to distinguish between senses, they had the option of choosing more than one sense. This follows from Buitelaar's earlier work [117]. In reporting this earlier work, Buitelaar suggests that semantic tagging should be viewed as more than disambiguation between senses. He suggests further that some senses may be systematically related (systematic polysemy) and should therefore not be disambiguated; rather, they should be left underspecified. To support this, a "new" type of lexicon, "CoreLex" was created through a design based on systematic polysemous classes. "Corelex", which uses a set of 442 polysemous classes, was also used in annotation work reported by Pustejovsky and colleagues (including Buitelaar) [125].

Fellbaum [126], Palmer [127], Kingsbury[128] and Miller [129] report on sense tagging tasks that involved the annotation of content words with WordNet synsets. Miller's [129] work is the WordNet group's annotation of a subset of the Brown Corpus; it is the basis for the determination of frequencies for senses in WordNet. Semantic annotation work done by Fellbaum [126], Palmer [127], and Kingsbury [128] are on the Wall Street Journal Corpus. However, they are geared towards the representation of predicate-argument structure, rather than classical surface grammatical analysis. Resnik [120] reports on a method for automatic sense disambiguation of nouns, using WordNet senses. His method also permits the assignment of higher-level WordNet categories rather than sense labels.

The semantic annotation work reported above made use of sense distinctions that are too fine-grained for practical use, considering there are approximately 48,800 noun synsets (word meanings) in WordNet. Chang [130] present work that assigns domain tags to WordNet entries, using a domain taxonomy which they established (from a combination of WordNet and The Far East Dictionary). Their reported work was still at a preliminary stage.

Reported work not based on WordNet include that by Ceusters [123] which presents the "Cassandra II" syntactic-semantic tagging system, a bracketing technique combining phrase structure tagging with semantic tagging. It is used to annotate parallel corpora of medical texts in different languages for marking similarities independent of a specific grammar formalism. Its technique is akin to thematic case role assignment. Dill and his IBM colleagues [131] report on "SemTag", an application written on the "Seeker" platform (a platform for large-scale text analytics) to perform automated semantic tagging of large corpora. Berg [77] presents "XERIC" networks which parse and represent sentence structure while also performing number-person and lexical disambiguation. Mayberry, III and Miikkulainen [122] present "INSOMNET", a connectionist model trained on semantic representations from LINGO Redwoods HPSG Treebank of annotated sentences. Zelle and Mooney [132] present a system that employs inductive logic programming to learn a shift-reduce parser that integrates syntactic and semantic constraints to produce case-role representations.

Lowe and colleagues [133] present a frame-semantic approach to semantic annotation. They argue that the number and arrangement of semantic tags must be constrained, lest the size and complexity of the tag sets used for semantic annotation become unwieldy both for humans and computers.

2.7 Summary

Syntactic parsing is fundamental to automatic natural language processing systems. The two main methods of parsing are top-down and bottom-up. Parsing techniques can also be classified in terms of directionality – directional and non-directional parsers. All parsing techniques can be grouped as either top-down or bottom-up, apart from, left-corner parsing which is a hybrid of both classes.

Symbolic, statistical and connectionist approaches have been used to implement the various parsing techniques. Hybrids of these approaches have also been used. Connectionist (or, artificial neural network-based) parsing systems could be localist, distributed or modular/hybrid. Modular/hybrid connectionist parsing models have exhibited superiority over the other types of connectionist parsers because of their better ability to learn and their competence in handling complex language structure. They have also been used to combine several types of linguistic information and continue to make significant progress.

3. PRELIMINARY ANALYSIS OF THE EXISTING CONNECTIONIST PARSING MODEL

3.1 Introduction

The existing parsing model [18, 19] used in this work employs a parsing process that involves two stages: delimitation of phrases and recognition of phrase structure. The phrase delimitation process is further broken down into two sub-processes: a right-to-left delimiter (RLD) process to discover the beginning of a phrase and a left-to-right delimiter (LRD) process to discover the corresponding end of the phrase. Each of these two delimiter sub-processes is implemented with the Temporal Auto-associative Simple Recurrent Network (TASRN) [68]; training of the networks is done with the standard back-propagation algorithm. This network, given its sequential input and feedback to context nodes, is architecturally better equipped than feed forward multilayer perceptrons (which are well-equipped for general-purpose pattern recognition and function approximation) in dealing with the temporal sequential nature of language and the dependencies that exist between words/phrases in sentences [66, 111].

As part of the preliminary stage of this project, experiments were set up to improve the performance of back-propagation learning by the delimiters, considering their vital temporal sequence processing role in the parser. The aims of these experiments were to reduce the tendency for the hidden neurons to be driven into saturation and to obtain optimal learning rates, momentum constants and network sizes for the two modified delimiter networks. With the same motive of improving learning by the delimiter networks, cross-validation was introduced to determine the stopping criterion during training. The generalisation performance of the refined delimiter modules were compared with that of the existing parser's [18, 19] delimiter modules. Their average labelled precision/recall measure (defined in

section 3.7) was also compared. To enable this comparison, all experiments in this chapter were run with training/test data sets drawn from the Lancaster Parsed Corpus. These experiments are covered in the following sections of this chapter.

3.2 The Existing Parsing Model

The existing parsing model [18, 19, 111] used for the investigations in this work is a hybrid shift-reduce, syntactic parser that integrates modular connectionist architectures with symbolic structures to automatically learn syntactic structure from annotated sentence examples. These sentence examples were extracted from the Lancaster Parsed Corpus (LPC) [20]. The LPC is therefore used as the fundamental basis of linguistic knowledge for this parsing model. Using this corpus, instead of strict grammar rules, enables the connectionist networks employed to learn less constraining grammars implicitly.

The process of parsing employed by this parsing model involves two stages: delimitation of phrases and recognition of phrase structure. The phrase delimitation process is further broken down into two sub-processes: right-to-left delimiter (RLD) process and left-to-right delimiter (LRD) process. Each of these two delimiter processes is implemented with the Temporal Auto-associative Simple Recurrent Network (TASRN) [68]. The phrase structure recognition process is implemented with a feed-forward Multilayer Perceptron (MLP).

The model also uses three core symbolic structures to store input symbols (word and constituent tags), properties of these input symbols and the phrase structure tree. These are: a linked list used to store tag information, a stack to store parse state and another stack to store the current input state.

This model achieved an average labelled precision/recall of 73% on a test set of sentences that were not used during training.

3.2.1 The Lancaster Parsed Corpus (LPC)

The Lancaster Parsed Corpus (LPC), a sub-set of the Lancaster-Oslo/Bergen (LOB) Corpus [112], is a corpus of British English sentences selected from printed publications of the year 1961. Each word in the LPC is tagged with its syntactic category using the CLAWS [113] word tagger. Each sentence in the corpus has been syntactically analysed in the form of labelled bracketing. This syntactic analysis has been done by computer, using a HMM-based probabilistic parser [114]. The syntactic analysis is completed using manual correction by several researchers.

The LPC contains 134,740 words, distributed in 11,827 sentences (13.29% of the LOB corpus); an average of 11.39 words per sentence. Most sentences over 20-25 words in length found in the LOB corpus were omitted from the LPC; in setting up the LPC, the prototype probabilistic parser developed to automatically parse the whole of the LOB corpus was unable to achieve a parse of most sentences over 20-25 words in length. There are samples, in the LPC, from each of the 15 genre categories in the LOB corpus. These categories are Press (Reportage), Press (Editorial), Press (Reviews), Reviews, and Skills, Trades and Hobbies. Other categories are Popular Lore, Belles Lettres, Biography and Essays, Miscellaneous (government documents, etc), General Fiction and Learned and Scientific Writing. The remaining categories are Humour, Science Fiction, Mystery and Detective Fiction, Adventure and Western Fiction, and Romance and Love Story.

3.2.2 Tag Representation

In order to adapt the parsing model to the pre-tagged LPC, an input representation was designed. This input representation was designed in such a way as to enhance the training process by reflecting similarities between symbols into their coding. To attain this, the input space is separated into regions; each region represents a group of symbols of the same type. 12 different symbol groups (5 terminal symbol groups and 7 non-terminal symbol groups) exist. The 5 terminal symbol groups are: punctuation, conjunctions, nouns, verbs, and prepositions. The 7 non-terminal groups are sentences, finite clauses, non-finite clauses, major phrase types, minor phrase types, slash tag phrases and coordinated phrases. These groups are represented using separate fields of the input vector.

Linear binary coding is used to represent the symbols within their respective group fields. An additional bit is used in the field to denote a symbol of that particular group type. This implies that the number of bits in each field is the minimum number required to represent all the symbols in the particular group plus 1.This representation scheme, shown in figure 3.1, ensures that patterns for symbols in different groups are always orthogonal to one another; patterns for symbols within a group are not orthogonal to one another.

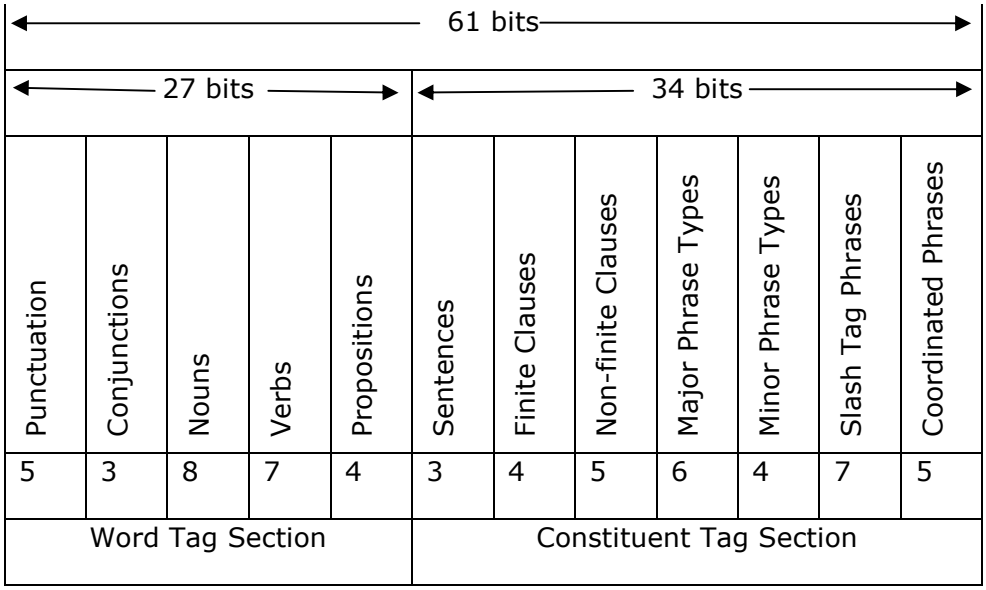

Figure 3.1: Word tag and constituent representation

A total of 61 bits are used, in this tag representation scheme, to encode all possible input symbols.

3.2.3 Parsing Architecture and Algorithm

3.2.3.1 The Architecture

This parsing model employs a modular, hybrid parsing architecture comprising three connectionist and three symbolic modules. The connectionist modules are used for the two fundamental processes involved in syntactic parsing: phrase boundary identification and phrase structure recognition. The symbolic modules are used for storage and to enhance the flow of information between different connectionist modules.

3.2.3.2 Phrase Delimitation

The phrase boundary identification, or phrase delimitation process is further broken down into two sub-processes; phrase delimitation requires the identification of both the beginning and end of a phrase or clause. These two sub-processes (implemented by two of the three available connectionist networks) are the right-to-left delimiter (RLD) process to identify the beginning of a phrase, and the left-to-right delimiter (LRD) process to identify the corresponding end of the phrase.

Since the number of input symbols processed by the each of the delimiter networks before the beginning (or end) of a phrase is encountered is variable and not known a priori, a recurrent neural network that is able to sequentially process linguistic input is assigned to each of the delimiter processes. The recurrent network assigned to these tasks is the Temporal Auto-associative Simple Recurrent Network (TASRN) [68]. This network is shown in figure 3.2.

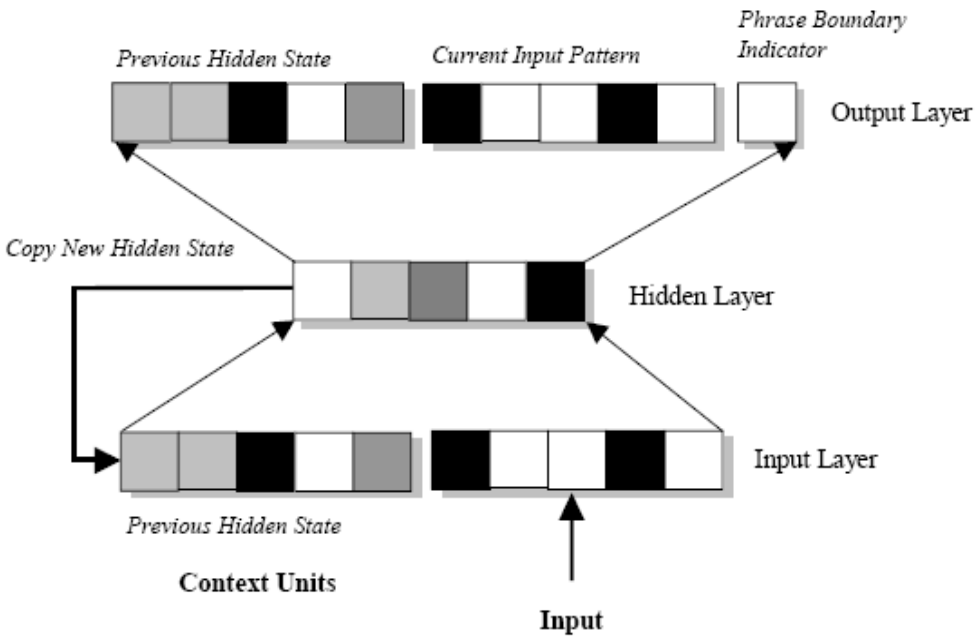

Figure 3.2: The TASRN architecture used for the LRD and RLD modules

The TASRN architecture has recurrent connections feeding back from the hidden units (these represent the internal reduced description of the input representation) back to the input units. The network also has, as elements of its output vector, the phrase boundary indicator (a bit in the output vector) and all the elements of the previous hidden and current input vectors. This architecture is set up to achieve auto-associative learning of the current input and hidden state during each stage of processing. It provides for temporal processing of linguistic input. Its memory limit is also enhanced by the availability of targets at every processing stage (input symbol and previous hidden state are produced as part of the output vector for every processing stage).

The phrase boundary indicator bit in the output vector of the delimiter networks uses a value between 0 and 1 to indicate when a phrase boundary has been encountered by the delimiters. A ramp followed by a step function is used to train this output unit; this indicates the phrase boundary when it is encountered while also indicating the proximity to the phrase boundary at each stage of processing a sequence. The output for the first symbol in a sequence is a 'don't care'. The output for the next symbol is 0. This ramps up to 0.4 for the penultimate symbol, and finally outputs 1 for the last symbol (beginning or end of phrase).

3.2.3.3 Phrase Structure Recognition

The Phrase Structure Recognition (PSR) module is used in the parsing model to classify collections of word and constituent tags as syntactic phrases, denoting each of these collections with a single syntactic tag. The collections of word and constituent symbols fed into the PSR module as input are derived from processing carried out by the two delimiter modules on the input sequences to the parser.

The PSR module is implemented using a feed-forward, Multilayer Perceptron (MLP) architecture, with a single hidden layer.

3.2.3.4 Symbolic Parse Organisation

While the connectionist modules cater for the linguistic constraints and actions to be carried out by the parser, the symbolic modules of this parser enable simple communication between the connectionist networks. These symbolic structures also allow the interpretation of the parser's actions as a whole. Three core symbolic structures, a linked list and two stacks, are used for storage in this parser. A linked list is used to hold tag information for the words and constituents, as provided in the used corpus. The Parse-stack is used to store parse state information and the resulting parse tree for each sentence. The Input-stack holds the current input state.

Apart from the core symbolic structures, temporary stacks are used to hold data passing between the RLD and LRD modules, and between the LRD and PSR modules.

3.2.3.5 The Algorithm

A supervisory code, the Scheduler, controls the interaction and flow of information between the connectionist and symbolic modules of this parsing model. The scheduler implements a deterministic shift-reduce parsing strategy which parses from right to left. The shift-reduce algorithm implemented by this parser is similar to that defined by Shieber [115, 116].

The parser accepts, as input, word tags and constituent tags used to annotate the Lancaster Parsed Corpus (LPC). Word tags represent the grammatical class of the

word. Constituent tags represent the syntactic phrases or clauses for a particular group of words and/or phrases.

With this model, the parsing of each sentence commences by passing its word tags sequentially to the Right-to-Left delimiter (RLD) module. The passing of these word tags to the RLD begins with the last tag of the sentence and shifts to the left; this goes on until the output of the RLD triggers to indicate the beginning (left-hand phrase boundary) of the first phrase to be reduced. The word tags are then passed sequentially to the Left-to-Right (LRD) delimiter module, beginning with the left-hand boundary tag, but now shifting to the right; this continues until the output of the LRD triggers to indicate the end (right-hand phrase boundary) of the first phrase to be reduced. The identification of the phrase boundaries provides the Scheduler with enough information to define the position and width of the reduction window; the tags within these boundaries are passed as input to the Phrase Structure Recogniser (PSR) modules. The output of the PSR, a constituent tag, is a reduction which is then substituted for the phrase in the sentence sequence on the Input-stack. After these, delimitation begins again. The delimiters are reset, and the RLD input is drawn once again beginning from the end of the sentence stored on the Input-stack. This time, the input sequence to the RLD will include one non-terminal symbol (constituent tag) amongst the remaining terminal symbols (word tags).

The process of delimitation, followed by reduction and substitution is repeated, thereby continuing the parse, until the reduction produces the sentence symbol. The output of the parser after each shift-reduce processing stage is a phrase or clause represented in the labelled bracketing format. The final parser output is a labelled bracketing structure that encodes the parse tree for the entire sentence.

3.3 Reducing Saturation

The TASRN architecture used for the delimiters, as shown in figure 3.2 [18, 19], has, as elements of its output vector, the phrase boundary indicator and all the elements of the previous hidden and current input vectors. Its target vector (desired response) has a similar composition. The values of the target vector elements need to be kept within the range of the logistic activation function, 0 to 1. This is to curb the tendency of the back-propagation algorithm to drive the free parameters of the delimiter network to infinity, thereby slowing down the learning process by driving the hidden neurons into saturation [94]. The purpose of this experiment was to select a set of input values that would reduce the tendency for the delimiter hidden nodes to be driven into saturation.

The experiment was carried out with two sets of input values. The first set had input vector element values, 0 and 1. The second set involved offsetting the input vector element values from the first set by 0.2. That is, 0 was offset to 0.2 and 1 was offset to 0.8. Training sessions were run with input values of 0.2 and 0.8 and the results were compared with sessions run with input values of 0 and 1. Training sessions were run till the rate of learning (rate of decrease of the RMSE) was empirically observed to have minimised. 2500 epochs were run with the right-to-left delimiter (RLD) while 3500 epochs were run with the left-to-right delimiter (LRD).

The root mean square error (RMSE) for the two sets of inputs was plotted against the number of epochs for the LRD and RLD as shown in figures 3.3 and 3.4. For both delimiters, the 0.2_0.8 inputs produced smoother RMSE curves although the 0_1 inputs produced a lower final RMSE. The high spikes observed on the RMSE curves with the 0_1 inputs, compared to the very low spikes (almost smooth) observed with the 0.2_0.8 inputs can be attributed to the difference between the

low (0 and 0.2) and high (1 and 0.8) values of the input sets and the impact of this difference on the error calculated after each forward pass.

Generalisation tests were also run for the RLD and LRD with the two sets of inputs, using a pure test set (this sample set is disjoint with the training set). For the RLD, after 2500 epochs the 0.2_0.8 input produced a sequence generalisation of 83.4397% compared to 85.5051% produced by the 0_1 inputs.
For the LRD, after 3500 epochs the 0.2_0.8 input produced a sequence generalisation of 90.445% compared to 84.1623% produced by the 0_1 inputs.

From the sequence generalisation results of the experiments, the 0.2_0.8 inputs displayed better performance for the LRD while the 0_1 inputs displayed better performance for the RLD (having also displayed better RMSE values). Because of the difference in generalisation performance of the two input sets on the LRD and RLD, the 0.2_0.8 input was only temporarily chosen for the optimal learning rate/momentum constant experiment (This did not affect the outcome of the investigations as all the parameters used were the same apart from the different learning rate/ momentum constant cases). The 0_1 input was, however, selected after it still came out with better performance when trained with optimal learning rate/momentum constant.

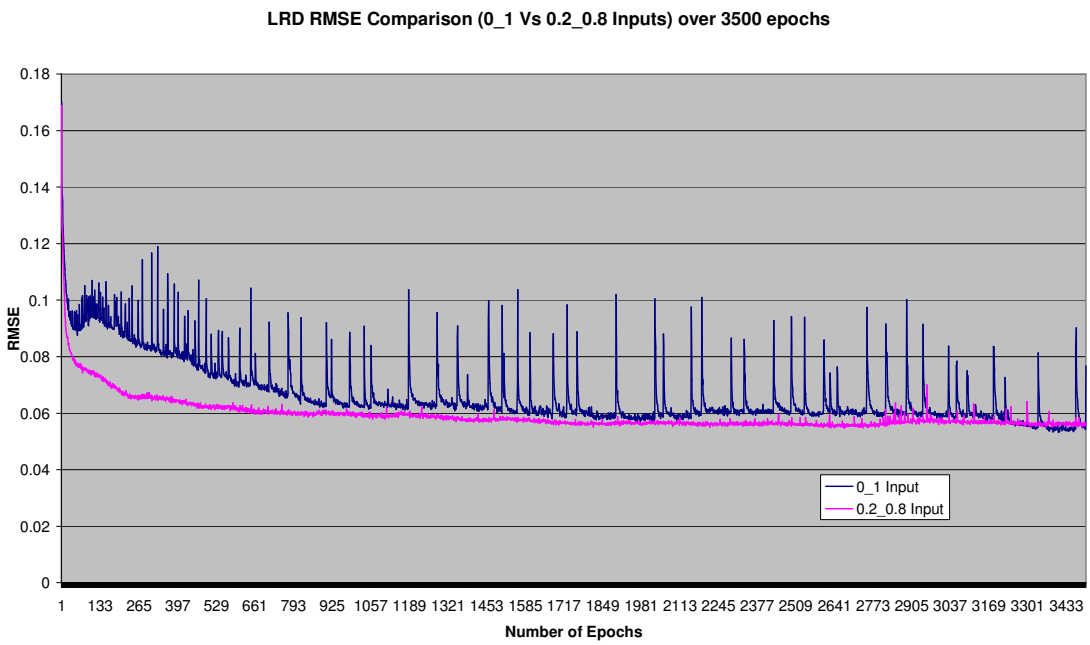

Figure 3.3: Plot of RMSE Against Number of Epochs (LRD)

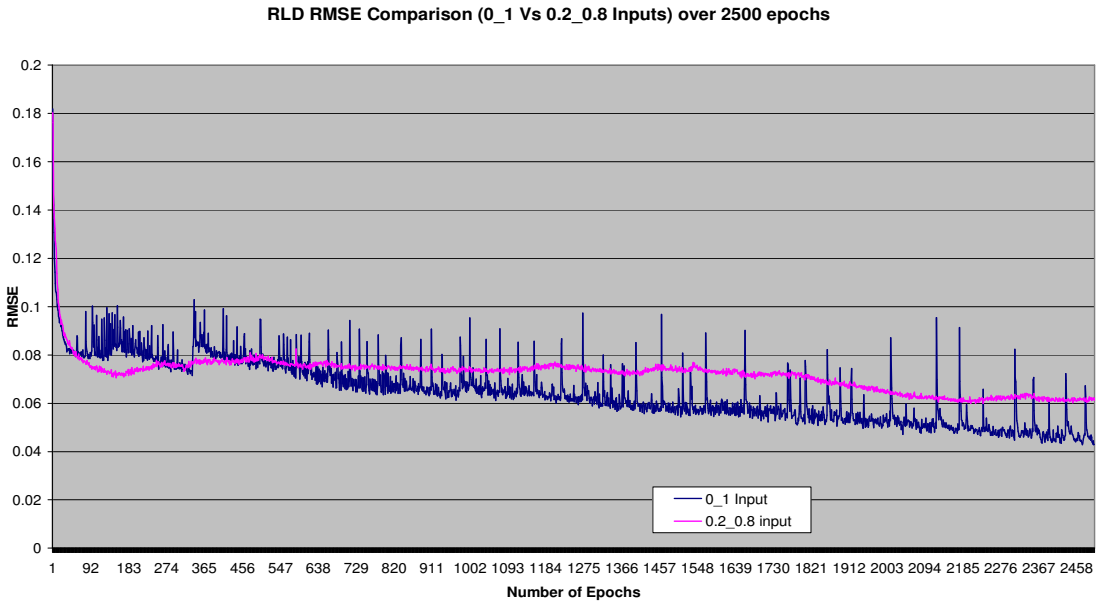

Figure 3.4: Plot of RMSE Against Number of Epochs (RLD)

3.4 Optimal Learning Rate/Momentum Constant

This set of the experiments aims to improve back-propagation learning on the delimiters by selecting the optimal combination of learning rate type, learning rate (or initial learning rate) and momentum term.

Three learning rate types were considered – fixed learning rate, search-then-converge learning rate [91], and delta-bar-delta learning rate [92]. Two momentum constants were used; 0.9 and 0.5. The training data set, which was scaled down (10167 patterns and 1354 sequences) to limit training time, had inputs of 0.2 and 0.8. All training sessions were done with 60 hidden nodes (chosen because of the reduced data set) and run for 2000 epochs (uniform training duration for all the learning rate type/momentum constant cases). With each momentum constant, training sessions were carried out with learning rate, η, fixed at 0.01, 0.1, 0.2, 0.3, 0.35 and 0.4. After that, for each momentum constant, α, training sessions were carried out using the search-then-converge learning rate with initial learning rates of 0.01, 0.1, 0.2, 0.3, 0.35 and 0.4.

Using the fixed learning rate schedule, the learning rate, η in the weight adaptation rule for standard back-propagation remained constant throughout the duration of training. The weight change, at time t+1, for weight ω_{ij} is as follows:

$$\Delta\omega_{ij}(t+1) = \eta\delta_j o_i + \alpha\Delta\omega_{ij}(t)$$

Where δ_j is the local gradient of node j

o_i is the output signal of node i

With the search-then-converge schedule, the learning rate, η in the weight adaptation rule above changes every time step as follows:

$$\eta(t) = \eta_0/(1 + (t/\tau))$$

where η_0 and τ are user-selected constants.

Training sessions were also run using the delta-bar-delta learning rate algorithm. This algorithm comprises a weight update rule and a learning rate update rule, and allows each weight to have its own learning rate. Each weight's learning rate varies with time as training progresses. The direction of learning rate change for each weight depends on the direction of the weight change; if the weight change is in the same direction over several time steps, the learning rate for that weight is increased, otherwise, it is decreased. The new learning rate for each weight at time, t + 1, is given by:

$$\eta_{jk}(t+1) \begin{cases} \eta_{jk}(t) + \kappa & \text{if } \Delta\omega_{jk}(t-1)\,\Delta\omega_{jk}(t) > 0, \\ (1-\gamma)\,\eta_{jk}(t) & \text{if } \Delta\omega_{jk}(t-1)\,\Delta\omega_{jk}(t) > 0, \\ \eta_{jk}(t) & \text{otherwise.} \end{cases}$$

Where $\Delta\omega_{jk}(t)$ is the weight change, at time t, for weight ω_{jk}

and, κ and γ are constants

Apart from requiring more processing time than the other two learning rate types considered, this learning rate type did not produce better generalisation performance than the other two and, so its results were not considered in the final performance ranking. From the plots of RMSE against Number of epochs for the different learning rates/ momentum constants, the learning rate and momentum constant combination that gave the best outcomes (assessed by the curves that converged at the lowest RMSE in fewer epochs and that produced the lowest RMSE) are shown in table 3.1.

Table 3.1: Training performance for combinations of learning rate /momentum term

Momentum Term	Learning Rate Type	(Initial) Learning Rate	RMSE	Epoch	Rank
0.9	Search-then-converge	0.3	0.100639	1301	1
0.9	Search-then-converge	0.4	0.101999	1303	2
0.5	Search-then-converge	0.4	0.100865	1335	3
0.5	Fixed	0.4	0.111686	1895	4

The pattern and sequence generalisations (using a natural test set) for the four listed parameter combinations were also the best. Their generalisation performances are as shown in table 3.2.

Based on the results shown in table 3.2, the search-then-converge learning rate with an initial rate of 0.4 and a momentum term of 0.9 were selected as the optimal combination. The difference in the top two ranking positions for the training and generalisation performance indices is because the RMSE values are calculated on a pattern by pattern basis (on-line mode) rather than at the sequence level.

Table 3.2: Generalisation performance for combinations of learning rate/momentum term

Momentum Term	Learning Rate Type	(Initial) Learning Rate	% Sequence Generalisation	% Pattern Generalisation	Generalisation Performance Rank
0.9	Search-then-converge	0.4	72.5366	95.5343	1
0.9	Search-then-converge	0.3	72.2727	95.3443	2
0.5	Search-then-converge	0.4	71.1718	95.1693	3
0.5	Fixed	0.4	67.3301	94.4783	4

3.5 Optimal Network Size

With the number of input layer neurons already fixed by the representation technique used, the network size of an artificial neural network depends on the number of neurons in its hidden layer. This makes hidden layer neurons responsible for properties of the network such as the ability to learn and to generalise. If the number of hidden layer neurons is too small, the network will be unable to learn. If the number is too large, the network will over-fit its training data and therefore be unable to generalise. The purpose of these experiments was to determine optimal network sizes (this is vital for optimal network performance [93, 94]) for the right-to-left (RLD) and left-to-right (LRD) networks by adopting, for each network, the number of hidden layer nodes that produced the best generalisation performance during training. The number of hidden layer neurons adopted for the RLD and LRD networks used in the existing parsing model [18, 19] were 165 and 110, respectively. These network sizes were chosen ahead of others because they produced the lowest root mean square errors during training [18, 19].

Cross-validation (as described in section 3.6) was used in this set of experiments. Training the delimiter networks involved halting the standard training process every 50 epochs to run generalisation tests with the validation sets during the following three consecutive training epochs to obtain a gradient for generalisation performance. Cross-validation was also used to detect the beginning of over-fitting during training; training was then stopped before convergence to check for over-fitting. Each training session was restricted to 2000 epochs to cater for cases where training was not stopped automatically.

In choosing the optimal number of hidden layer nodes for the delimiter networks, a theoretical "optimal" number of hidden nodes was calculated for each delimiter by equating the number of its weights to the sum of the products of sequence lengths

and their frequencies. Hidden node values were then selected between these "optimal" number of hidden nodes and the number of hidden nodes used in the existing parser. 77 and 61 hidden nodes were derived for the RLD and LRD, respectively. For the RLD, six sizes were considered: 77, 88, 107, 126, 145 and 165 hidden nodes. The maximum test generalisation and training performance exhibited by each hidden layer size in the course of training was recorded and plotted as shown in fig. 3.5.

For the LRD, three sizes were considered: 61, 85 and 110 hidden nodes. The maximum test generalisation and training performance exhibited by each hidden node in the course of training was recorded and plotted as shown in fig. 3.6.

For the RLD, as shown in fig. 3.5, the network size with 145 hidden nodes produced the best test generalisation (87.91%) and was therefore chosen as the optimal network size for the RLD.

For the LRD, as shown in fig. 3.6, the network size with 85 hidden nodes produced the best test generalisation (89.18%) and was therefore chosen as the optimal network size for the RLD.

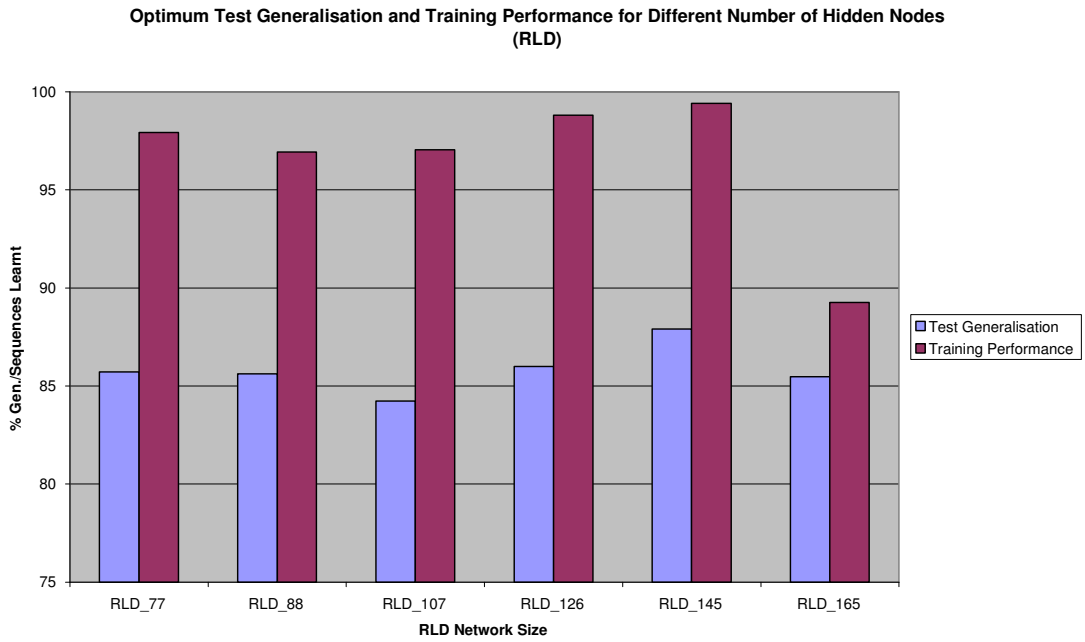

Figure 3.5: Training/Generalisation performance for different RLD network sizes

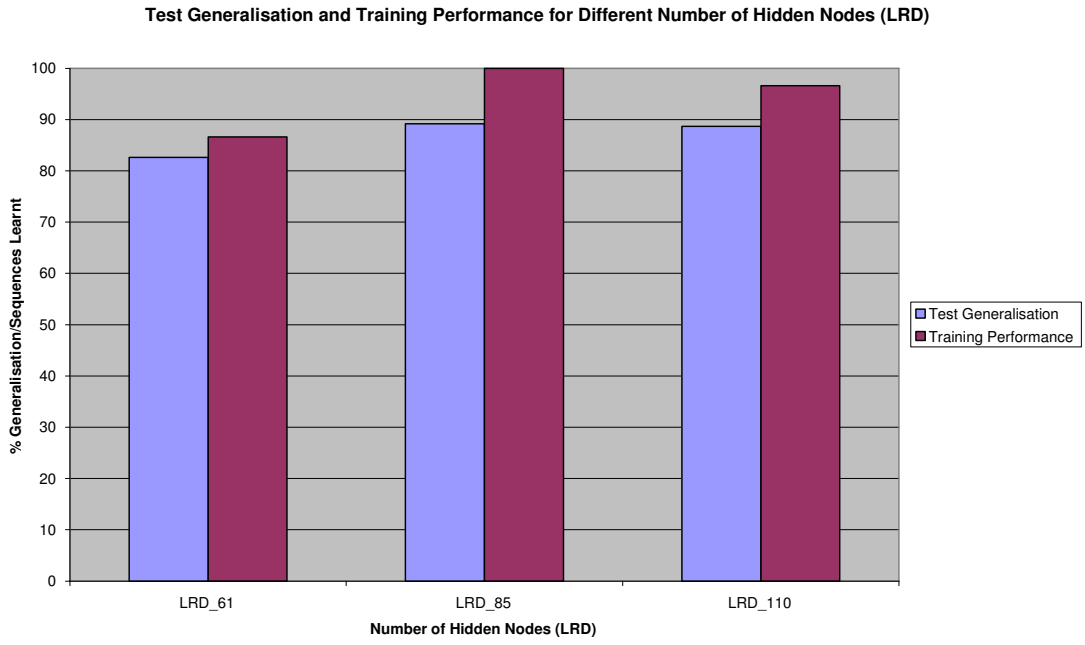

Figure 3.6: Training/Generalisation performance for different LRD network sizes

3.6 Using Cross-validation for Automatic Early Stopping

During supervised training of neural networks, the objective is usually to achieve optimal generalisation performance. Generalisation performance is the performance of the network when presented with examples it has not seen before. However, with training factors such as large parameter space, trained networks stand the risk of over-fitting [95]. This is a situation during network training where the training performance gets better while the generalisation performance gets worse.

Over-fitting can be checked with the use of cross-validation, which is a standard technique in statistics [96]. An approach of cross-validation, known as the hold-out method [94], involves dividing the available data set into two sets, a training set and a test set. The training set is further split into two disjoint sets, an estimation set and a validation set. The idea is to train the network only on the estimation set and occasionally evaluate it on the validation set.

To curb over-fitting, a cross-validation procedure known as the early stopping method of training [94] is used. This procedure is used to detect when over-fitting starts during training; training is then stopped before convergence to check the over-fitting. In carrying out this procedure, the synaptic weights of the neural network are fixed after a period of estimation (training). A forward pass of the network is then run, using the validation set. After error measurements have been taken for all examples of the validation set, the training resumes for another period and the process continues. Training is then stopped based, not on the performance of the training data but on the performance of the validation data.

For this work, network training and test samples were drawn from the Lancaster Parsed Corpus [20]. Data from the LPC was passed through some complexity constraint [18, 19] before the training and test data were chosen. After the

complexity constraint was applied, every 8^{th} sentence was extracted for the training (estimation) set. To ensure that the validation and test sets do not have sentences that are present in the training set (this is essential), every 8^{th} + 1 sentence was extracted for the validation set and every 8^{th} + 2 sentence was extracted for the test set. After the training and test data had been processed to RLD and LRD training and test sequences, the estimation set was compared with the validation set and any sequence that occurred in both sets was removed from the validation set, making both sets disjoint. The same process was carried out for the estimation and test sets and the validation and test sets. All three sets, were, therefore, disjoint. For the Left-to-right delimiter (LRD), the estimation set had 4068 sequences, while the validation set had 2292 sequences and the test set, 1871 sequences. For the Right-to-left delimiter (RLD), the estimation set had 4060 sequences while the validation set had 2663 sequences and the test set, 2262 sequences.

Training the delimiters involved halting standard training every 50 epochs to run generalisation tests (with the validation sets) for a further three consecutive training epochs. Again, the purpose here is to determine a gradient for the generalisation performance. This implies that generalisation tests were carried out at epochs 50, 51, 52, 100, 101, 102, 150, 151, 152 and so on. If the cross-validation test generalisation decreased over 5 successive generalisation tests, training was stopped automatically. The result of the training would be the set of weights that came out with the best test generalisation; only one duplicate weight set is needed for this [102].

3.7 Sentence Parse Performance

On completion of the experiments to optimise the temporal sequence processing modules of the original parsing model [18, 19], the delimiter module and sentence level performances for the original and refined models were compared.

For the delimiter module performance, the optimal weight sets obtained for the RLD and LRD, were used to run a test on a test set not seen by the delimiters during training. This same test set was used to test the original delimiter modules, using the original weight sets.

The results obtained from these tests are as shown in table 3.3.

Table 3.3: Module-level performance for refined parser

Module	**Original Model Delimiter (Test Generalisation)**	**Refined Model Delimiter (Test Generalisation)**
Right-to-Left Delimiter	84.8806%	83.8638%
Left-to-Right Delimiter	88.5088%	87.6537%

For the sentence level tests, the optimal weight sets for the delimiters were plugged into the whole parser. PARSEVAL measures, a widely used standard for assessing the performance of statistical broad coverage parsing models [103], was used. This involved measuring the parsers' labelled precision and labelled recall. Labelled precision is the ratio of the number of correct constituents output by the parser to the number of constituents output by the parser. Labelled recall is the ratio of correct constituents output by the parser to the number of constituents in the Treebank parse.

Labelled Precision = (no. of correct constituents)/(no. of constituents output by parser)

Labelled Recall = (no. of correct constituents output by parser)/(no. of constituents in Treebank parse)

The average labelled precision/recall obtained for the original and refined parsers are shown in table 3.4.

Table 3.4: Sentence-level performance for refined parser

Original Parser (Average Labelled Precision/Recall)	**Refined Parser (Average Labelled Precision/Recall)**
73.3%	72.5%

From the module and sentence level performance results obtained, the refinements on the temporal sequence processing modules of the parser did not yield any significant improvement in the parsers performance. However, the parser has maintained its ability to learn and it is envisaged that it is adaptable to other corpora (given its representation and modular architecture which should make it independent of individual corpora). To confirm its adaptability to other corpora, the parser will be extended to the widely used Wall Street Journal Corpus in the next chapter.

4. THE CORPUS-BASED PARSING MODEL: ADAPTED TO THE WALL STREET JOURNAL CORPUS

4.1 Introduction

As part of the aims of this work, investigations have been carried out into the generic nature of the corpus-based, connectionist parsing model [18, 19, 111] used for this project. This investigation has been with a view to demonstrating that the parsing model is adaptable to other corpora and for other syntactic and semantic annotations without its architecture or its algorithm being changed. This parser has been trained and successfully evaluated on the Lancaster Parsed Corpus (LPC) [20]. However, the widely used Wall Street Journal (WSJ) sections of the Penn Treebank Corpus [101] have become the internationally accepted benchmark corpus for parsing models [5, 6, 7, 98, 99, 100]. This has informed the need to adapt the parser to the WSJ corpus. The WSJ corpus used is the BLLIP (Brown Laboratory for Linguistic Information Processing) 1987-89 Corpus [90] which overlaps the WSJ portion of the Penn Treebank Corpus.

Adapting the existing parsing model to the WSJ Corpus required the extraction and syntactic grouping of all tags used in the corpus. Binary input representations were then designed for the tags to make them compatible with the parser. Training, cross-validation and test data were generated from the 1989 section of the corpus for the left-to-right delimiter (LRD), right-to-left delimiter (RLD) and phrase structure recogniser (PSR) modules of the parser. These modules, which are the connectionist modules of the hybrid parser, were then trained and the optimal weight sets obtained for the sentence level evaluation of the parser.

Section 4.2 presents the nature of the BLLIP 1987-89 WSJ Corpus; its content and tagging convention. Section 4.3 deals with the input representations designed for

the WSJ tags. In section 4.4, a parsing example of a sentence from the WSJ Corpus is presented. Section 4.5 focuses on the training, cross-validation and test data sets generated for the left-to-right delimiter (LRD), right-to-left delimiter (RLD) and phrase structure recogniser (PSR) modules of the parser. In section 4.6, the training and generalisation performances of the delimiter networks are assessed. The training and generalisation performances of the phrase recognition network are assessed in section 4.7 while the sentence level performance of the parsing model on the WSJ Corpus is dealt with in section 4.8. The outcome of adapting the parsing model to the WSJ Corpus is discussed in section 4.9.

4.2 The BLLIP 1987-89 Wall Street Journal CORPUS

4.2.1 Corpus Content

The BLLIP 1987-89 Wall Street Journal (WSJ) Corpus [90] is a pre-parsed newswire corpus which contains a complete, Penn Treebank II-style [101, 119] parsing of the three-year Wall Street Journal archive (provided by Dow Jones, Inc.) from the ACL/DCI (Association for Computational Linguistics/ Data Collection Initiative) Corpus of American English. This corpus contains about thirty million words of text, and its parsing and part-of-speech (POS) annotation were done using statistically-based methods developed by Eugene Charniak, Don Blaheta, Niyu Ge, Keith Hall, John Hale and Mark Johnson [90] of the Brown Laboratory for Linguistic Information Processing. All the processing for this corpus was implemented by machine. The processing comprised basic parsing, grammatical/functional tag assignment, full noun-phrase co-reference identification, pronoun reference identification, and empty node insertion.

The BLLIP 1987-89 WSJ Corpus both overlaps and supplements the one million-word, 1989 Wall Street Journal section of the Penn Treebank Corpus. In a bid to save on parsing time, sentences of length greater than 70 words (including punctuations) were not included in this corpus. The developers report that in about one news story in a thousand, there was some parser error. These parser errors imply that stories in which they occur get cut short; errors led to partial parses.

4.2.2 Tagging Convention

4.2.2.1 The Penn Treebank II Convention

The Penn Treebank II bracketing convention was implemented during the second phase of the Penn Treebank Project at the University of Pennsylvania, U.S.A. The syntactic annotation scheme used is designed to allow the extraction of simple predicate/argument structure.

In addition to the standard syntactic constituent tags (e.g. NP, PP, VP, etc.) functional tags are also assigned to constituents under this scheme. These functional tags denote text categories (list markers, titles, headlines and datelines), grammatical functions (surface subject, logical subjects in passives, true clefts, non NPs that function as NPs, clausal and NP adverbials, non VP predicates, topicalized and fronted constituents, closely related – adjuncts -) and semantic roles (vocatives, direction and trajectory, location, manner, purpose and reason, temporal phrases). For this work, as with other reported work on the WSJ Corpus [5, 6, 7, 98, 99, 100], only the standard syntactic constituent tags are used; this is all that is needed for skeletal syntactic analysis.

This scheme also annotates null elements in a wide range of cases.

4.2.2.2 Exceptions to the Penn Treebank II Convention

All parsing in the BLLIP 1987-89 WSJ Corpus is done using the Penn Treebank II conventions with four exceptions. The first exception is that certain auxiliary verbs (e.g. "have", "been", etc.) are deterministically labelled AUX or AUXG (e.g., "having").

The next exception to the Penn Treebank II scheme in this corpus is that root nodes are given the new non-terminal label S1 (as opposed to the empty string in the Penn Treebank).

Another exception is that numbers attached to non-terminals indicating co-reference are preceded by "#" (as opposed to "-" in the Penn Treebank).

The fourth exception is that two new grammatical function tags, PLE (denoting pleonastic, a form of non-coreferential pronouns) and DEI (denoting deictic, a form of non-coreferential pronouns) have been added.

In setting up this corpus, sentences of length greater than 70 words (including punctuations) were ignored.

4.2.3 The BLLIP 1987-89 WSJ Corpus Vs The Lancaster Parsed Corpus

Like the Lancaster Parsed Corpus (LPC), the syntactic part of the Penn Treebank-II tagset (used in tagging the Wall Street Journal – WSJ -) is based on that of the Brown Corpus. However, the annotation scheme used for the WSJ Corpus is an extended and somewhat modified form of that used for the LPC [119].

Whereas word tags in the LPC are quite detailed and unique to particular lexical items, the Penn Treebank tag-set is designed in such a way to eliminate lexical redundancy. For example, the LPC distinguishes five different forms of main verbs (VB – base form of lexical verb (uninflected present tense, infinitive); VBD – past tense of lexical verb; VBG – present participle or gerund of lexical verb; VBN – past participle of lexical verb; VBZ – 3rd person singular of verb). This same paradigm is also used in the LPC for the word, *have*, irrespective of whether it is used as a main or auxiliary verb (i.e. HV, HVD, HVG, HVN, HVZ). The LPC also provides tags for three forms of *do* (DO – base form; DOD – past tense; DOZ – third person singular present) and eight forms of *be* (BE - be; BED - were; BEDZ - was; BEG - being; BEM - am; BEN - been; BER – are, 're; BEZ – is, 's). On the contrary, since the distinctions between the forms of VB on the one hand and the forms of HV, DO and BE on the other hand are lexically recoverable, they are eliminated in the tag-set for the WSJ; only the five forms of VB are used as shown in table 4.1 below.

Table 4.1: Elimination of lexically recoverable distinctions in verbs

Word	Word tag
Drink	VB
Drinks	VBZ
Drank	VBD
Drinking	VBG
Drunk	VBN
Be	VB
Is	VBZ
Was	VBD
Being	VBG
Been	VBN
Do	VB
Does	VBZ
Did	VBD
Doing	VBG
Done	VBN
Have	VB
Has	VBZ
Had	VBD
Having	VBG
Had	VBD

Another example of the elimination of lexical redundancy in the WSJ Corpus, as opposed to the LPC, is the case of tagging words that precede articles in noun phrases. In the LPC, the tags ABL, ABN and ABX are used to denote pre-qualifiers

(quite, rather, such), pre-quantifiers (all, half, many, nary) and both, respectively. However, in the WSJ Corpus, a single tag, PDT is used to denote all these words (all categorised as pre-determiners).

Null tags are used in the WSJ Corpus in cases such as WH-movement, topicalization, indicating which lexical NP is to be interpreted as the null subject of an infinitive complement clause and aiding the interpretation of other grammatical structure where constituents do not appear in their default positions. Null tags are not used in the LPC. Also, the tags, AUX and AUXG are used for auxiliary verbs in the BLLIP WSJ Corpus. Auxiliary verbs are not denoted in the LPC.

Compared to the 184 tags (143 tags for words and punctuations; 41 tags for constituents) used in the LPC, 84 (57 tags for words and punctuations; 27 tags for constituents – excluding the functional tags -) are used in the BLLIP WSJ Corpus. The Penn Treebank II tags, therefore represent coarser syntactic categories, compared to the syntactic categories represented by the LPC tags.

The BLLIP WSJ Corpus consists of longer sentences than the LPC. Sentences of length greater than 70 words (including punctuations) were not included in the BLLIP WSJ Corpus. Most sentences over 20-25 words in length found in the LOB corpus were omitted from the LPC.

4.3 Tag Representations

The first step in adapting the original parsing model [18, 19, 111] (which was trained on the Lancaster Parsed Corpus) to the BLLIP WSJ Corpus was to design binary input representations for the word and constituent tags used in the corpus. The same technique used for LPC tag representation [18, 19, 111] in the existing parser was adopted because of its success. This technique sees the creation of

input representations that aid the training process by segmenting the input space into different regions that correspond to different word and constituent tag types.

Each segment has an associated signalling bit (its first bit) that is only active when an input symbol belongs to a syntactic group represented by that sub-section. The remaining bits in the segment are used to represent the particular input symbol. This ensures that the representation of any two symbols in different syntactic groups will be orthogonal to each other.

The fifty-seven word tags encoded were placed into five syntactic groups: punctuations, co-ordinate conjunction, preposition/sub-ordinate conjunction, nouns and verb groups. 19 bits of the 46-bit input space were used to represent word tags. As an example, the word tag, **NNP** (Proper noun, singular) is represented as follows:

0000000100011000000000000000000000000000000000

Punctuations, which were removed in the processing of LPC sentences are included here, and treated the same as words. Although they add to the complexity of the parsing task, they are expected to provide linguistic cues. This should aid decision making during parsing.

The constituent tags were placed into thirteen groups according to their syntactic categories: adjective phrase, adverb phrase, conjunction phrase, fragment, phrase containing an interjection, noun phrase, prepositional phrase, phrase within parentheses, reduced relative clause, sentence/clause, unlike co-ordinated phrase, verb phrase, and unknown/uncertain category. 27 bits of the 46-bit input space were used to represent constituent tags. As an example, the constituent tag, **NP** (Noun phrase) is represented as follows:

0000000000000000000000000000000001001000000000

The organisation of the input representation space is as shown in figure 4.1.

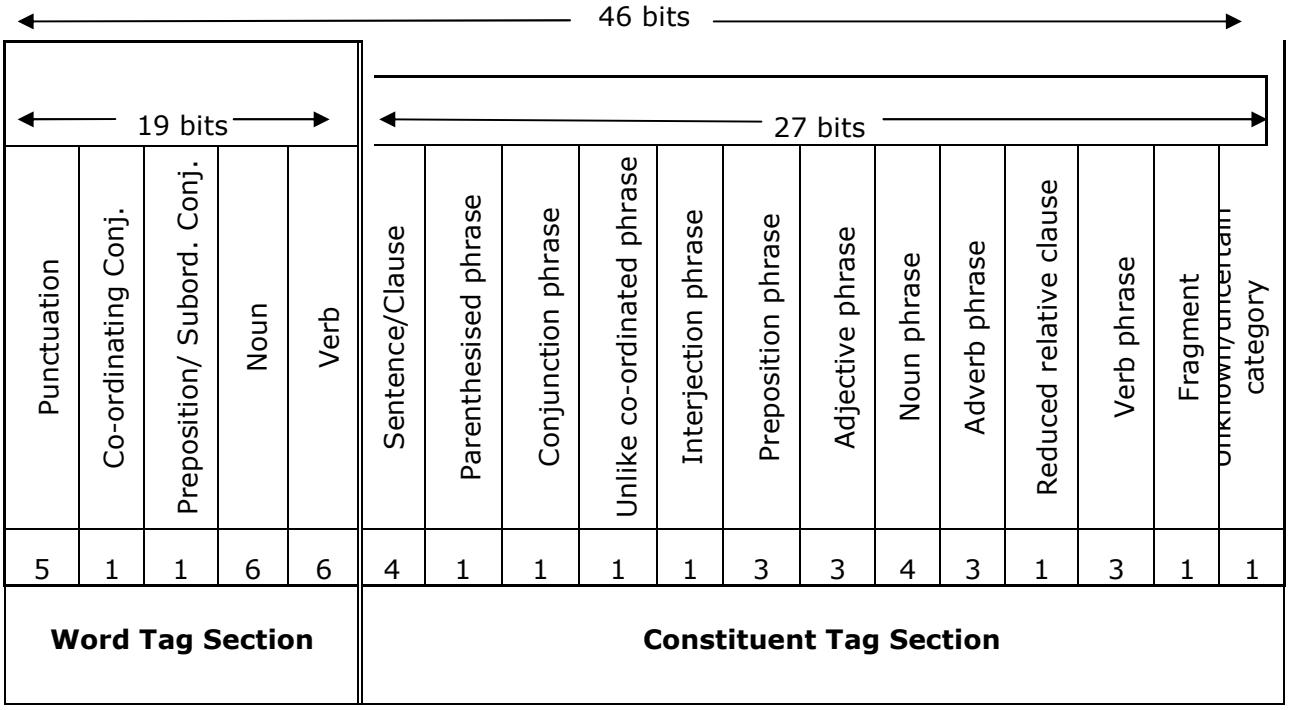

Figure 4.1: Word tag and constituent tag representation

4.4 A Parsing Example

A high level description of the parsing process, using the Wall Street Journal Corpus is presented in this section. The following sentence is used to facilitate this description:

The rate will increase 0.25 point each quarter beginning in the third quarter of the financing.

The input to the parser is a sequence of word tags that correspond to the words of the sentence. Therefore, the input to the parser is as follows:

DT NN MD VB CD NN DT NN VBG IN DT JJ NN IN DT NN .

For clarity, the actual words of the sentence will be used for description in this section.

On being presented with each sentence for parsing, the Scheduler first delimits the sentence with asterisks. These asterisks are used as begin and end markers before all the input symbols are pushed onto the Input-stack from left-to-right. For this sentence the initial content of the Input-stack is:

*** The rate will increase 0.25 point each quarter beginning in the third quarter of the financing. ***

Table 4.2: A parsing example

	Input-stack	**Extracted Phrase**	**Reduction**	**Parse-stack Entries**
1	* The rate will increase 0.25 point each quarter beginning in the third quarter of the financing. *	the financing	NP	(NP the financing)
2	* The rate will increase 0.25 point each quarter beginning in the third quarter of NP. *	of NP	PP	(PP of (NP the financing))
3	* The rate will increase 0.25 point each quarter beginning in the third quarter PP. *	the third quarter	NP	(NP the third quarter) (PP of (NP the financing))
4	* The rate will increase 0.25 point each quarter beginning in NP PP. *	NP PP	NP	(NP (NP the third quarter) (PP of (NP the financing)))
5	* The rate will increase 0.25 point each quarter beginning in NP. *	in NP	PP	(PP in (NP (NP the third quarter) (PP of (NP the financing))))
6	* The rate will increase 0.25 point each quarter beginning PP. *	beginning PP	PP	(PP beginning (PP in (NP (NP the third quarter) (PP of (NP the financing)))))
7	* The rate will increase 0.25 point each quarter PP. *	each quarter	NP	(NP each quarter) (PP beginning (PP in (NP (NP the third quarter) (PP

				of (NP the financing)))))
8	* The rate will increase 0.25 point NP PP. *	0.25 point	NP	(NP 0.25 point) (NP each quarter) (PP beginning (PP in (NP (NP the third quarter) (PP of (NP the financing)))))
9	* The rate will increase NP NP PP. *	increase NP NP PP	VP	(VP increase (NP 0.25 point) (NP each quarter) (PP beginning (PP in (NP (NP the third quarter) (PP of (NP the financing))))))
10	* The rate will VP. *	will VP	VP	(VP will (VP increase (NP 0.25 point) (NP each quarter) (PP beginning (PP in (NP (NP the third quarter) (PP of (NP the financing)))))))
11	* The rate VP. *	The rate	NP	(NP The rate) (VP will (VP increase (NP 0.25 point) (NP each quarter) (PP beginning (PP in (NP (NP the third quarter) (PP of (NP the financing)))))))
12	* NP VP. *	NP VP.	S	(S (NP The rate) (VP will (VP increase (NP 0.25

				point) (NP each quarter) (PP beginning (PP in (NP (NP the third quarter) (PP of (NP the financing))))))) .)
13	* S *	S	S1	(S1 (S (NP The rate) (VP will (VP increase (NP 0.25 point) (NP each quarter) (PP beginning (PP in (NP (NP the third quarter) (PP of (NP the financing))))))) .))

As indicated in Table 4.2 above, at stage one of the parsing process for the given sentence, the RLD and LRD have extracted *the financing* as the first valid syntactic phrase. This phrase consists of two symbols; the PSR network requires a phrase length of ten symbols, in addition to the six look-back and one look-ahead symbols. An additional eight null symbols are therefore added to the extracted phrase to pad it out to the ten symbol requirement. The input to the recogniser is therefore:

beginning in the third quarter of **the financing ^ ^ ^ ^ ^ ^ ^ ^** .

The actual phrase to be recognised is emphasized in bold and underlined as well. The six symbols (seen above as words, but presented to the parser as part-of-speech tags) to the left of the actual phrase are the look-back symbols, as extracted by the RLD. The symbol to the right of the actual phrase is the look-ahead symbol, as extracted by the LRD. The PSR network performs a forward-pass computation and the corresponding Euclidean distances between the resulting

output vector and the binary representations of all the constituent tags in the tag database indicates *NP* to be the nearest match. *NP* is attached to *the financing* and the following bracketing sequence is pushed onto the Parse-stack:

(NP the financing)

The Input-stack is then updated to reflect this 'reduction' by pushing back on the six look-back symbols, the constituent tag, NP, and the look-ahead symbol. The Input-stack now holds the following sequence:

* The rate will increase 0.25 point each quarter beginning in the third quarter of NP. *

As shown in table 4.1, at processing stage two, the RLD and LRD have extracted *of NP* as the next valid phrase. Again, an additional eight null symbols are required to pad out the phrase. At this stage, the input to the recogniser is:

quarter beginning in the third quarter **of NP ^ ^ ^ ^ ^ ^ ^ ^** .

The Recogniser network performs a forward-pass computation and PP is selected as the nearest matching constituent tag. The content of the Parse-stack now becomes:

(PP of (NP the financing))

The Input-stack is then updated to reflect this 'reduction' by pushing back on the six look-back symbols, the constituent tag, PP, and the look-ahead symbol. The Input-stack now holds the following symbols:

* The rate will increase 0.25 point each quarter beginning in the third quarter PP. *

At the third processing stage, the RLD and LRD extract the phrase, the third quarter. This phrase is padded out with seven additional null symbols to meet the ten-symbol input requirement for the Recogniser. The input to the Recogniser is:

0.25 point each quarter beginning in **the third quarter ^ ^ ^ ^ ^ ^ ^** PP

The Recogniser network performs a forward-pass computation and NP is selected as the nearest matching constituent tag. The content of the Parse-stack now becomes:

(NP the third quarter) (PP of (NP the financing))

The Input-stack is then updated by pushing back on the six look-back symbols, the constituent tag, NP, and the look-ahead symbol. The reduction has now been implemented and the Input-stack now holds the following symbols:

* The rate will increase 0.25 point each quarter beginning in NP PP. *

The fourth phrase to be extracted from the sentence, by the RLD and LRD, is *NP PP*. This phrase is padded out with eight null symbols before the six look-back and one look-ahead symbols are added to make up the Recogniser input. The Recogniser input at this stage is:

0.25 point each quarter beginning in **NP PP ^ ^ ^ ^ ^ ^ ^ ^** .

The Recogniser network then performs a forward-pass computation in response to the input above. NP is selected as the nearest match constituent tag. The content of the Parse-stack therefore changes to the following:

(NP (NP the third quarter) (PP of (NP the financing)))

The Input-stack is then updated to reflect this 'reduction' by pushing back on the six look-back symbols, the constituent tag, NP, and the look-ahead symbol. The Input-stack now holds the following symbols:

* The rate will increase 0.25 point each quarter beginning in NP. *

At the fifth processing stage, the phrase, *in NP* is extracted by the RLD and LRD. Again, an additional eight null symbols are required to pad out the phrase. At this stage, the input to the recogniser is:

increase 0.25 point each quarter beginning **in NP ^ ^ ^ ^ ^ ^ ^ ^** .

The Recogniser network then performs a forward-pass computation in response to the input above. PP is selected as the nearest match constituent tag. The content of the Parse-stack therefore changes to the following:

(PP in (NP (NP the third quarter) (PP of (NP the financing))))

The Input-stack is then updated by pushing back on the six look-back symbols, the constituent tag, PP, and the look-ahead symbol. The reduction has now been performed and the Input-stack now holds the following symbols:

* The rate will increase 0.25 point each quarter beginning PP. *

The sixth phrase to be extracted from the sentence, by the RLD and LRD, is *beginning PP*. This phrase is padded out with eight null symbols before the six look-back and one look-ahead symbols are added to make up the Recogniser input. The Recogniser input at this stage is:

will increase 0.25 point each quarter **beginning PP ^ ^ ^ ^ ^ ^ ^ ^** .

The Recogniser network then performs a forward-pass computation in response to the input above. PP is selected as the nearest match constituent tag. The content of the Parse-stack therefore changes to the following:

(PP beginning (PP in (NP (NP the third quarter) (PP of (NP the financing)))))

The Input-stack is then updated by pushing back on the six look-back symbols, the constituent tag, PP, and the look-ahead symbol. The reduction has now been performed and the Input-stack now holds the following symbols:

* The rate will increase 0.25 point each quarter PP. *

At the seventh processing stage, the phrase, *each quarter* is extracted by the RLD and LRD. Again, an additional eight null symbols are required to pad out the phrase. At this stage, the input to the recogniser is:

The rate will increase 0.25 point **each quarter ^ ^ ^ ^ ^ ^ ^ ^** PP

The Recogniser network then performs a forward-pass computation in response to the input above. NP is selected as the nearest match constituent tag. The content of the Parse-stack therefore changes to the following:

(NP each quarter) (PP beginning (PP in (NP (NP the third quarter) (PP of (NP the financing)))))

The Input-stack is then updated by pushing back on the six look-back symbols, the constituent tag, NP, and the look-ahead symbol. The reduction has now been performed and the Input-stack now holds the following symbols:

* The rate will increase 0.25 point NP PP. *

The eighth phrase extracted by the RLD and LRD is *0.25 point*. Eight additional null symbols are required to pad out the phrase. As there were not enough available input symbols on the left of the phrase to make up the required six look-back

symbols, the RLD network would have used an additional null symbol to pad the number of look-back symbols to the desired length. At this stage, the input to the recogniser is:

* The rate will increase ^ **0.25 point ^ ^ ^ ^ ^ ^ ^ ^** NP

The Scheduler arranges the null padding to the right of the look-back symbols. This ensures that the look-back symbols are positioned further away from the phase. The Recogniser network then performs a forward-pass computation in response to the input above. NP is selected as the nearest match constituent tag. The content of the Parse-stack therefore changes to the following:

(NP 0.25 point) (NP each quarter) (PP beginning (PP in (NP (NP the third quarter) (PP of (NP the financing)))))

The Input-stack is then updated by pushing back on the look-back symbols, the constituent tag, NP, and the look-ahead symbol. The reduction has now been performed and the Input-stack now holds the following symbols:

* The rate will increase NP NP PP. *

At the ninth processing stage, the phrase, *increase NP NP PP* is extracted by the RLD and LRD networks. This time, an additional six null symbols are required to pad out the phrase. Again, there were not enough available input symbols on the left of the phrase to make up the required six look-back symbols; the RLD network would have used two additional null symbols to pad the number of look-back symbols to the desired length. At this stage, the input to the recogniser is:

* The rate will ^ ^ **increase NP NP PP ^ ^ ^ ^ ^ ^** .

The Recogniser network then performs a forward-pass computation in response to the input above. VP is selected as the nearest match constituent tag. The content of the Parse-stack therefore changes to the following:

(VP increase (NP 0.25 point) (NP each quarter) (PP beginning (PP in (NP (NP the third quarter) (PP of (NP the financing))))))

The Input-stack is then updated by pushing back on the look-back symbols, the constituent tag, VP, and the look-ahead symbol. The reduction has now been performed and the Input-stack now holds the following symbols:

* The rate will VP. *

At the tenth processing stage, the phrase, *will VP* is extracted by the RLD and LRD networks. Additional eight null symbols are required to pad out the phrase. Again, there were not enough available input symbols on the left of the phrase to make up the required six look-back symbols; the RLD network would have used three additional null symbols to pad the number of look-back symbols to the desired length. At this stage, the input to the recogniser is:

* The rate ^ ^ ^ **will VP ^ ^ ^ ^ ^ ^ ^ ^** .

The Recogniser network then performs a forward-pass computation in response to the input above. VP is selected as the nearest match constituent tag. The content of the Parse-stack therefore changes to the following:

(VP will (VP increase (NP 0.25 point) (NP each quarter) (PP beginning (PP in (NP (NP the third quarter) (PP of (NP the financing)))))))

The Input-stack is then updated by pushing back on the look-back symbols, the constituent tag, VP, and the look-ahead symbol. The reduction has now been performed and the Input-stack now holds the following symbols:

* The rate VP. *

The eleventh phrase to be extracted by the RLD and LRD networks is *The rate*. Eight additional null symbols are required to pad out the phrase. As there were not enough available input symbols on the left of the phrase to make up the required six look-back symbols, the RLD network would have used five additional null symbols to pad the number of look-back symbols to the desired length. At this stage, the input to the recogniser is:

* ^ ^ ^ ^ ^ **The rate ^ ^ ^ ^ ^ ^ ^ ^** VP

The Recogniser network then performs a forward-pass computation in response to the input above. NP is selected as the nearest match constituent tag. The content of the Parse-stack therefore changes to the following:

(NP The rate) (VP will (VP increase (NP 0.25 point) (NP each quarter) (PP beginning (PP in (NP (NP the third quarter) (PP of (NP the financing)))))))

The Input-stack is then updated by pushing back on the look-back symbol, the constituent tag, VP, and the look-ahead symbol. The reduction has now been performed and the Input-stack now holds the following symbols:

* NP VP. *

The twelfth phrase to be extracted by the RLD and LRD networks is *NP VP.*. Seven additional null symbols are required to pad out the phrase. As there were not enough available input symbols on the left of the phrase to make up the required six look-back symbols, the RLD network would have used five additional null symbols to pad the number of look-back symbols to the desired length. At this stage, the input to the recogniser is:

* ^ ^ ^ ^ ^ **NP VP . ^ ^ ^ ^ ^ ^ ^** *

The Recogniser network then performs a forward-pass computation in response to the input above. S is selected as the nearest match constituent tag. The content of the Parse-stack therefore changes to the following:

(S (NP The rate) (VP will (VP increase (NP 0.25 point) (NP each quarter) (PP beginning (PP in (NP (NP the third quarter) (PP of (NP the financing))))))))

The Input-stack is then updated by pushing back on the look-back symbol, the constituent tag, S, and the look-ahead symbol. The reduction has now been performed and the Input-stack now holds the following symbols:

* S *

At the thirteenth processing stage, the phrase *S* is extracted by the RLD and LRD networks. Nine additional null symbols are required to pad out the phrase. Again, there were not enough available input symbols on the left of the phrase to make up the required six look-back symbols; the RLD network would have used five

additional null symbols to pad the number of look-back symbols to the desired length. At this stage, the input to the recogniser is:

* ^ ^ ^ ^ ^ **S ^ ^ ^ ^ ^ ^ ^ ^ ^ ^** *

The Recogniser network then performs a forward-pass computation in response to the input above. S1 is selected as the nearest match constituent tag. The content of the Parse-stack therefore changes to the following:

(S1 (S (NP The rate) (VP will (VP increase (NP 0.25 point) (NP each quarter) (PP beginning (PP in (NP (NP the third quarter) (PP of (NP the financing)))))))))

As the constituent tag, S1 represents the entire sentence structure (the 'root' of the parse tree), its selection by the Recogniser denotes the end of the 'shift-reduce' parsing process. The last state of the Parse-tree becomes the final parse state. This state corresponds to the traditional tree representation illustrated in figure 4.2.

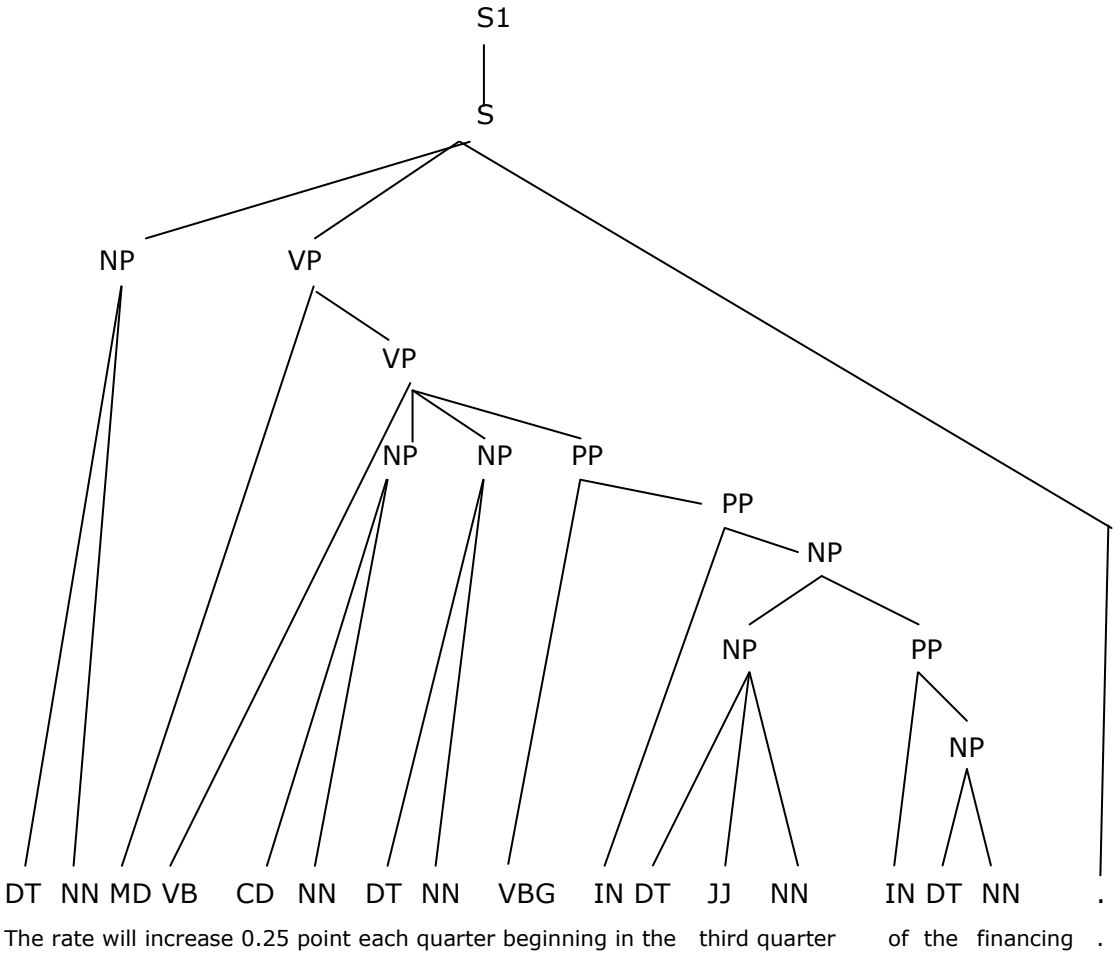

Figure 4.2: Parse tree denoting an example parse from the WSJC

4.5 Training, Validation and Test Samples

Training, cross-validation and test data sets have been selected from the 1989 section of the corpus (The BLLIP WSJ Corpus consists of 3 sections: 1987, 1988 and 1989). This has been done to align them with the WSJ section of the Penn Treebank which consists of material from the 1989 archive of the journal.

The 1989 section of the BLLIP WSJ Corpus comprises 32 sub-sections (10 – 41). Pre-parsed sentences from all the sub-sections have been used to provide data to the different sets; all the sub-sections were collapsed into one file. There are 40,043 pre-parsed sentences in the 1989 section of the BLLIP WSJ Corpus.

In sampling data for the training set, every 800^{th} sentence, beginning from the first sentence, was extracted from the corpus. 206 sentences (0.51% of the 1989 section of the BLLIP WSJ Corpus) were selected for the training data sets. The number of sentences in the training set was arrived at after attempts to incorporate more sentences led to impractically long training times, given the computer resource used (Intel Pentium 4 CPU 1.70 GHz; 1.70GHz, 1 GB of RAM). Every 160^{th} sentence, beginning from the second sentence (i.e. the 160^{th} + 1 sentence), was extracted for the test set. 1059 sentences (2.64% of the 1989 section of the BLLIP WSJ Corpus) were selected for the test data sets. Every 2400^{th} sentence, beginning from the third sentence (i.e. the 2400^{th} + 2 sentence), was picked for the cross-validation set. 74 sentences (0.18% of the 1989 section of the BLLIP WSJ Corpus) were selected for the cross-validation data sets. This provides the needed generalisation test for the cross-validation and test data sets, as the data are extracted in such a way that the different data sets are independent samples of the corpus. Complexity constraints were placed on these data to make them compatible

with data used in reported work [5]. To restrict the complexity of the data, only sentences of 40 words or less have been picked for all 3 data sets. Another complexity constraint imposed has been to restrict the valence of the right-to-left delimiter to 17 words; if the RLD has to process more than 17 symbols before the beginning of a phrase is found, the sentence containing that phrase is not used in training, cross-validation or testing.

Table 4.3: RLD, LRD and PSR data generation

	RLD Input Sequence	**LRD Input Sequence**	**Recogniser Input Pattern**	**Recogniser Output Pattern**
1	* **. CD , CD NNP** VBD NNS NNP CC , VBD	NNP NNS VBD **NNP CD , CD** . * ^	VBD , CC NNP NNS VBD **NNP CD , CD** ^ ^ ^ ^ ^ ^ .	NP
2	* . **NP VBD** NNS NNP CC , VBD RB	CC NNP NNS **VBD NP** . * ^	RB VBD , CC NNP NNS **VBD NP** ^ ^ ^ ^ ^ ^ ^ ^ .	VP
3	* . VP **NNS NNP** CC , VBD RB NN DT	VBD , CC **NNP NNS** VP . *	DT NN RB VBD , CC **NNP NNS** ^ ^ ^ ^ ^ ^ ^ ^ VP	NP
4	* . **VP NP** CC , VBD RB NN DT	VBD , CC **NP VP** . * ^	DT NN RB VBD , CC **NP VP** ^ ^ ^ ^ ^ ^ ^ ^ .	S
5	* . S CC , **VBD** RB NN DT CC * ^	DT NN RB **VBD** , CC S	* CC DT NN RB ^ **VBD** ^ ^ ^ ^ ^ ^ ^ ^ ,	VP
6	* . S CC , VP **RB** NN DT CC * ^ ^	CC DT NN **RB** VP , CC	* CC DT NN ^ ^ **RB** ^ ^ ^ ^ ^ ^ ^ ^ ^ VP	ADVP
7	* . S CC , VP ADVP **NN DT** CC * ^ ^ ^ ^	^ * CC **DT NN** ADVP VP ,	* CC ^ ^ ^ ^ **DT NN** ^ ^ ^ ^ ^ ^ ^ ^ ADVP	NP
8	* . S CC , **VP ADVP NP CC** * ^ ^ ^ ^ ^	^ ^ * **CC NP ADVP VP** , CC S	* ^ ^ ^ ^ ^ **CC NP ADVP VP** ^ ^ ^ ^ ^ ^ ,	S
9	* **. S CC , S** * ^ ^ ^ ^ ^	^ ^ * **S , CC S .** * ^ ^	* ^ ^ ^ ^ ^ **S , CC S .** ^ ^ ^ ^ ^ ^ *	S
10	* **S** * ^ ^ ^ ^ ^	^ ^ * **S** * ^ ^	* ^ ^ ^ ^ ^ **S** ^ ^ ^ ^ ^ ^ ^ ^ ^ *	S1
11	* **S1** * ^ ^ ^ ^ ^	^ ^ * **S1** * ^ ^		

Pre-processing the sentences from the corpus involved generating data sets for the left-to-right delimiter (LRD), right-to-left delimiter (RLD) and phrase structure recogniser (PSR) modules. So, the training, cross-validation and test sets each had LRD, RLD and PSR data sets generated from it (an example is shown in table 4.3). Each of these generated data sets had any replicated sequences within it removed. Structural replication would occur in cases where sentences shared the same phase structure. For example, all the sentences will generate *S1*^^^^^ for the RLD network and ^^*S1*^^ for the LRD network. Most sentences will also generate *S*^^^^^ for the RLD network and ^^*S*^^ for the LRD network. The occurrence of replicated sequences is enhanced by the use of word tags, rather than the words themselves, as input representation to the different parser networks. The presence of these replicated sequences creates an imbalance with some sequences occurring more frequently than others in the training set. If the training set is used in this state, network learning would be skewed in favour of the replicated sequences; at the expense of the less frequently occurring sequences.

The data sets with non-replicated sequences were processed to remove any conflicting sequences. An input sequence for the RLD or LRD network is considered to be in conflict with another if it is a sub-set of that other sequence. An input pattern for the Recogniser network would be in conflict if it occurred more than once in a training set and one or more occurrences have different target outputs associated with it. Conflicts within the data sets can be reduced by adding further contextual information to resolve the sequence ambiguity. This implies adding look-back symbols to the RLD data and, look-back and look-ahead data to the LRD data. The cost of reducing the number of conflicts with further contextual information is an increase in the length of input sequences which increases training times. It is therefore necessary to determine the optimum number of look-back and look-ahead symbols for the RLD and LRD networks.

Optimum Number of Look-back and Look-ahead

Prior to pre-processing the data sets, experiments were carried out to determine the optimum number of look-back and look-ahead for the LRD, and look-back for the RLD (the RLD only requires look-back because sentences are processed from right to left and the RLD always starts processing sequences from the end of a sentence). Look-back symbols are those symbols to the left of a phrase. Look-ahead symbols are the symbols to the right of a phrase. Although look-back and look-ahead symbols are not part of a phrase, they play an essential role in the phrase delimitation process by providing context, which enables the phrase delimitation networks of the parser to resolve sequence ambiguities. RLD and LRD data sets were extracted from 652 sentences (1989 section of the BLLIP WSJ Corpus) for these experiments.

In determining the optimum number of look-back and look-ahead symbols for the LRD, different look-back/look-ahead combinations were considered (Table 4.4) and the combination (3 look-back symbols, 3 look-ahead symbols) with the lowest level of conflicting sequences was chosen as the optimum combination. The other combinations that had the same (or slightly lower) levels of conflicting sequences were not considered because they would make the sequences longer, thereby adding complexity to them with little gain in terms of conflicting sequence reduction.

Table 4.4: Optimising LRD Look-back and Look-ahead Symbols

Look-Back Symbols	Look-Ahead Symbols	Conflicting Sequences	Training Sentences Affected	% Training Corpus Affected
1	1	83	265	40.6
1	2	17	59	9.0
1	3	18	57	8.7
1	4	18	57	8.7
2	1	33	75	11.5
2	2	8	17	2.6
2	3	7	12	1.8
2	4	7	12	1.8
3	1	17	33	5.1
3	2	2	6	0.9
3	3	2	5	0.8
3	4	2	5	0.8
4	1	8	17	2.6
4	2	1	5	0.8
4	3	1	4	0.6
4	4	1	4	0.6

In determining the optimum number of look-back symbols for the RLD, different look-back symbols were considered (Table 4.5) and the number of look-back symbols (6 look-back symbols) with the lowest level of conflicting sequences was chosen as the optimum look-back symbol.

Table 4.5: Optimising RLD Look-Back Symbols

Look-Back Symbols	Conflicting Sequences	Training Sequences Affected	% Training Corpus Affected
1	147	634	97.2
2	91	225	34.5
3	49	90	13.8
4	24	34	5.2
5	9	9	1.4
6	6	6	0.9

Balancing the Training Data

At this stage of pre-processing, the cross-validation and test data sets for the RLD, LRD and PSR networks are ready to be used. The training data for the PSR network is also ready to be used. However, the training data sets for the RLD and LRD networks need to be balanced according to the different sequence lengths (The LRD has 8 different sequence lengths ranging from length 7 to length 13, while the RLD has 10 different sequence lengths ranging from length 8 to length 17). Balancing is done to aid learning by ensuring that the frequency of occurrence of sequence lengths is the same for all lengths.

For each of these two data sets, the frequency of the most frequent sequence length (length 9 for the RLD, and length 8 for the LRD) is used as a standard; all other sequence lengths are made up to this standard using sequence replication. The sequences with sequence length used as a standard for balancing are not

replicated during balancing; this skews training to their disadvantage. To curtail this new imbalance, the unbalanced data set is added to the balanced data set to create the final training data set. This way every sequence is replicated, at least once. In order to determine the number of unbalanced data sets to be added to the balanced data set to create an optimum final data set, experiments were carried out on different combinations of balanced and unbalanced data sets for both delimiter networks. The data sets experimented on comprised an unbalanced set, a balanced set and a combination of a balanced set and one unbalanced set. Also experimented on were a combination of a balanced set and two unbalanced set and a combination of an unbalanced set and three unbalanced set.

The RLD was trained with the five different data sets for 300 epochs. Overall training performance, in terms of sequences learnt, and training performances on sequences of particular sequence lengths were compared for each data set (as shown in table 4.6). Learning progress, using a plot of root mean square error against number of epochs (figure 4.3) was also compared for the different data set combinations. Considering these performance indices, the final training data set for the RLD network is represented as shown in equation 4.1 below.

$$N_f = N_b + N_{ubl} \qquad (4.1)$$

Where N_f = final training data set
N_b = Balanced training data set
N_{ubl} = Unbalanced training data set

Table 4.6: Optimising RLD Training Data Set (Balanced and Unbalanced Sets Combination)

Data Set	Unbalanced Set	Balanced Set	Balanced + UnBalanced Set	Balanced + 2 * UnBalanced	Balanced + 3 * UnBalanced	Number of Unbalanced Sequences
% Sequences Learnt	75.943	93.5848	94.4747	94.1438	91.708	-
Length 8	70.5263	100	100	97.9198	98.9899	95
Length 9	80.9854	78.368	87.9908	92.6097	87.4519	1299
Length 10	71.2247	82.679	86.6355	85.0721	84.4846	841
Length 11	77.7244	86.2972	91.0556	91.7943	88.9625	624
Length 12	80.2469	96.6128	96.9193	95.9938	93.2188	324
Length 13	62.931	98.3064	99.1519	100	99.15	116
Length 14	35.2941	100	100	98.0014	98.0716	51
Length 15	27.2727	100	100	100	100	11
Length 16	0	100	100	100	100	6
% Patterns Learnt	87.5557	91.1249	90.8686	90.6391	90.2659	-

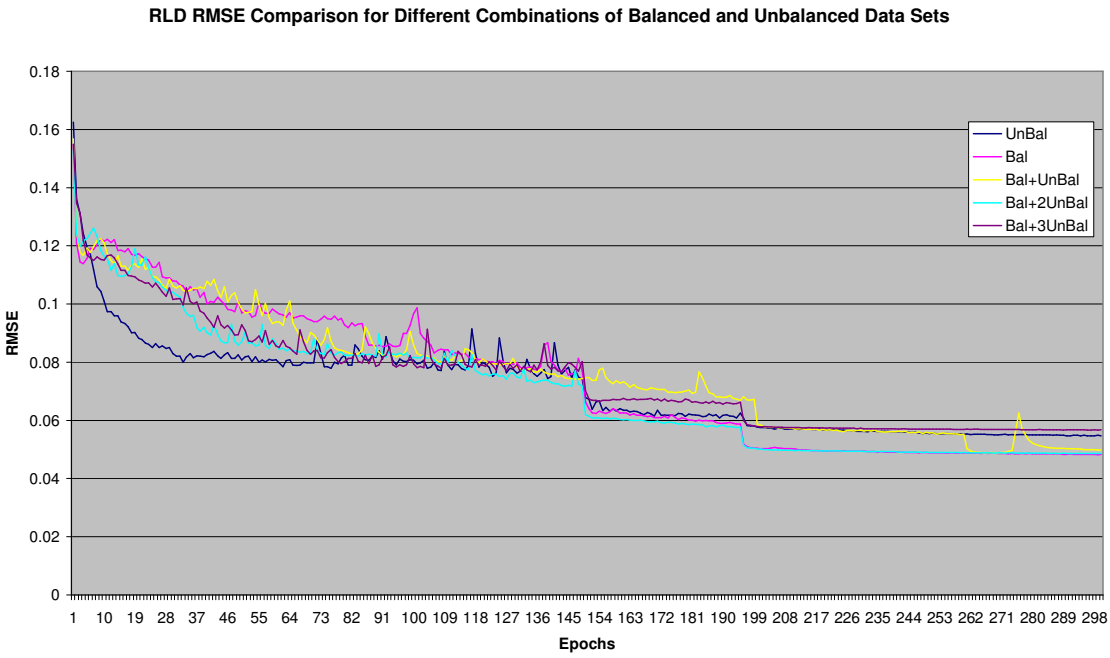

Figure 4.3: RLD RMSE for Combinations of Balanced and Unbalanced Data Sets

The LRD was trained with the five different data sets for 500 epochs. As with the RLD, overall training performance, in terms of sequences learnt and training performances on sequences of particular sequence lengths were compared for each data set (as shown in table 4.7). Learning progress, using a plot of root mean square error against number of epochs (figure 4.4) was also compared for the different data set combinations. Considering these performance indices, the final training data set for the LRD network is represented as shown in equation 4.2 below.

$$N_f = N_b + 2 \text{ X } N_{ubl} \qquad (4.2)$$

Where N_f = final training data set

N_b = Balanced training data set

N_{ubl} = Unbalanced training data set

Table 4.7: Optimising LRD Training Data Set (Balanced and Unbalanced Sets Combination)

Data Set	Unbalanced Set	Balanced Set	Balanced + UnBalanced Set	Balanced + 2 * UnBalanced	Balanced + 3 * UnBalanced	Number of Unbal. Seq.s
% Seq. Learnt	93.8917	95.5078	94.5029	97.6048	94.2593	-
Length 7	90.2439	91.0464	90.5976	96.3992	91.8629	656
Length 8	96.548	85.2211	89.9173	97.0874	93.1499	1854
Length 9	91.3655	92.2869	90.5612	94.4912	88.7993	498
Length 10	86.4286	100	97.1916	98.5942	95.6904	140
Length 11	78.9474	100	100	100	100	19
Length 12	100	100	100	100	100	7
Length 13	100	100	100	100	100	2
% Patterns Learnt	86.8371	89.55	89.0158	89.0615	88.4673	-

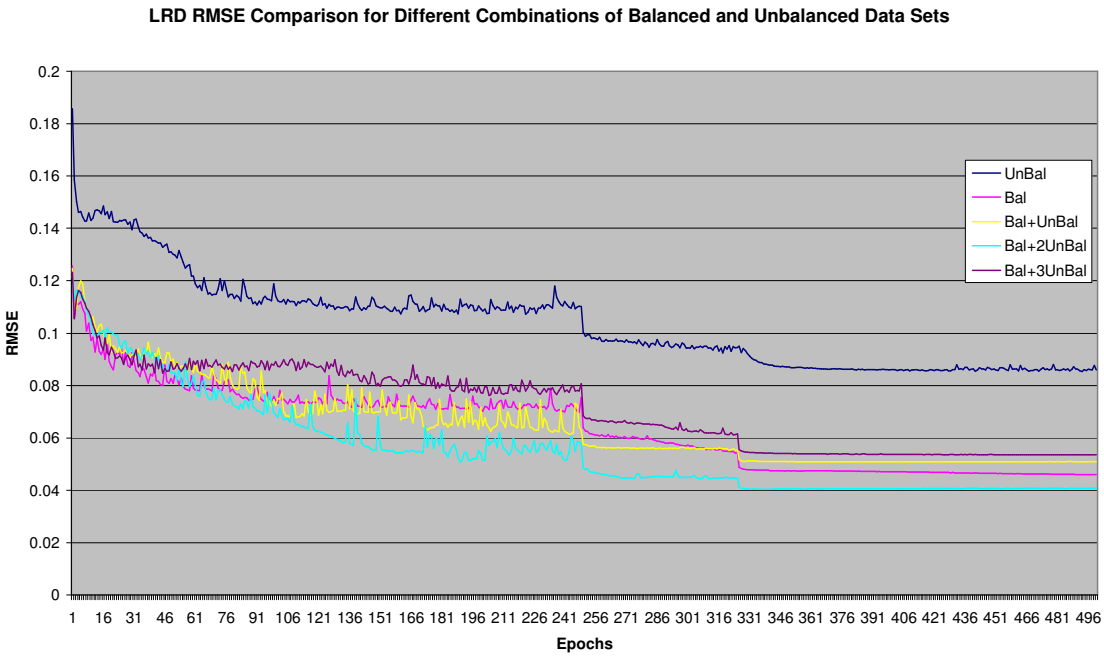

Figure 4.4: LRD RMSE for Combinations of Balanced and Unbalanced Data Sets

On completion of pre-processing, the training data set for the LRD consisted of 14,839 sequences and 142,840 patterns. The training data set for the RLD had 14,268 sequences and 175,115 patterns. The training data set for the PSR was made up of 3,103 patterns.

The cross-validation data set for the LRD had 1169 sequences and 9584 patterns, while that for the RLD comprised 1201 sequences and 13,846 patterns. The cross-validation data set for the PSR had 1191 patterns.

The test data set generated for the LRD was made up of 13,383 sequences and 109,494 patterns. The test data set for the RLD had 15,605 sequences and 178,507 patterns. The test data set for the PSR consisted of 15,317 patterns.

4.6 Phrase Segmentation Performance

The training data generated were trained and tested (for training and generalisation performances) to determine whether the delimiter modules of the parser could be successfully adapted to the Wall Street Journal Corpus. The sentences used in generating these data included punctuations, which were treated the same as words (punctuations were removed in the processing of LPC sentences). Although including punctuations add to the complexity of the parsing task, they are expected to provide linguistic cues. Corresponding training data were therefore generated from the same sentences used in generating the main training data, but with punctuations removed. These corresponding training data (from sentences with punctuations removed) were used to investigate if the inclusion of punctuations in the training data aids decision making during parsing

The LRD was trained, with an empirically determined network size of 105 hidden nodes for 500 epochs. 96.97% of the 14,839 sequences were learnt. Details of the LRD training result are displayed in table 4.8. Table 4.9 indicates the training performance of the LRD network on sequences of different lengths. When the corresponding training data without punctuations were used in training the LRD (using the same network configuration and number of epochs), 94.50% of sequences were learnt.

A plot (figure 4.5) of the root mean square error (RMSE) for the LRD at each of the 500 epochs is observed. Also plotted, in this figure, is the RMSE for the LRD at these 500 epochs, using corresponding training data with punctuations removed from the training sentences. These plots detail the learning process for the LRD, showing RMSE curves (in both cases) with negative gradients; this indicates that the network is learning with each periodic presentation of the given data. The plots

also show that the LRD performed better when punctuations were included in the sentences that when punctuations were removed.

Table 4.8: Training Results for the LRD and RLD

	Hidden Nodes	Connections	No. of Patterns	No. of Sequences	Epochs	RMS Error	% Pat. Learnt	% Seq. Learnt
RLD	165	70,172	175,115	14,268	360	0.0407269	91.55	96.31
LRD	105	32,072	142,840	14,839	500	0.0457812	89.29	96.97

Table 4.9: Training Results (for sequences of different lengths) for the LRD

Sequence Length	**7**	**8**	**9**	**10**	**11**	**12**	**13**
No. of Sequences	2290	3384	2179	1839	1731	1712	1704
% Sequences Learnt	96.68	92.38	95.27	99.35	100	100	100

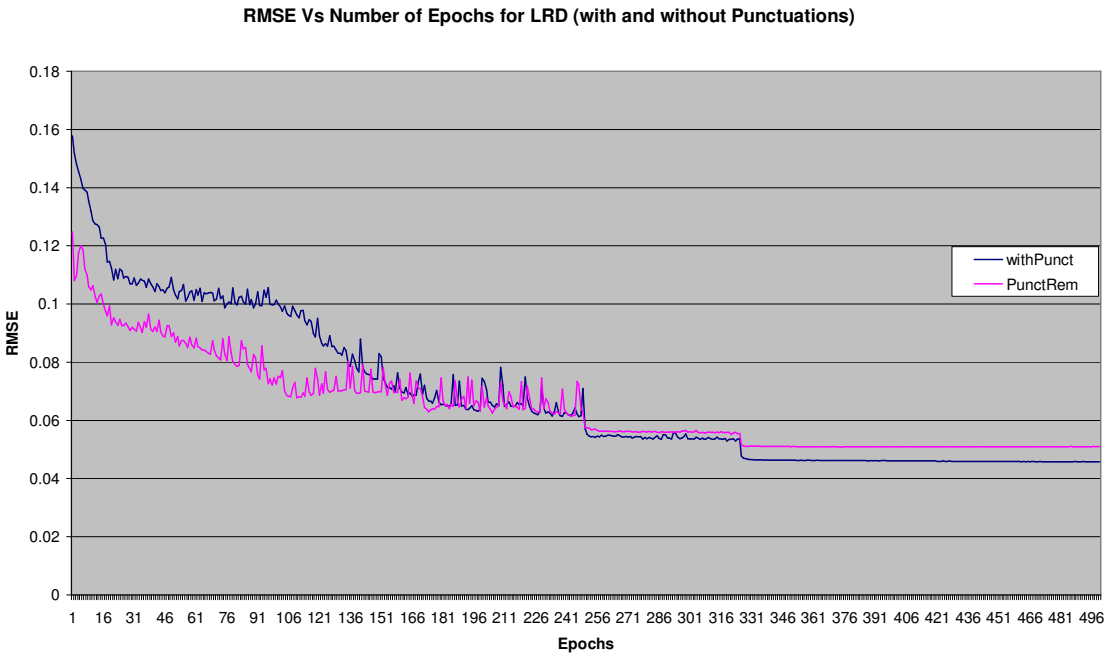

Figure 4.5: Plot of RMSE Against Number of Epochs for the LRD (WSJ Data)

To validate the RMSE training result for the LRD network, the network was tested after the 500th epoch. The test was carried out using both the cross-validation (containing 1169 sequences generated from 74 sentences) and the test (containing 13383 sequences generated from 1059 sentences) data sets. Results from this test indicate that the network came up with a sequence generalisation performance of 84% and 80.05% on the cross-validation and test data sets, respectively.

The RLD was trained, with an empirically determined network size of 165 hidden nodes for 360 epochs. 96.31% of the 14,268 sequences were learnt. Details of the RLD training result are displayed in table 4.8. Table 4.10 indicates the training performance of the LRD network on sequences of different lengths. When the corresponding training data without punctuations were used in training the RLD (using the same network configuration and number of epochs), 94.68% of sequences were learnt.

A plot (figure 4.6) of the root mean square error (RMSE) for the RLD at each of the 360 epochs is observed. Also plotted, in this figure, is the RMSE for the RLD at these 360 epochs, using corresponding training data with punctuations removed from the training sentences. These plots detail the learning process for the RLD, showing RMSE curves (in both cases) with negative gradients; this indicates that the network is learning with each periodic presentation of the given data. The plots also show that the RLD performed better when punctuations were included in the sentences that when punctuations were removed.

Table 4.10: Training Results (for sequences of different lengths) for the RLD

Length of Sequence	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**
No. of Sequences	1119	1119	2222	1780	1585	1460	1386	1231	1159	1127
% Sequences Learnt	100	100	92.44	90.62	95.14	93.50	98.56	100	100	100

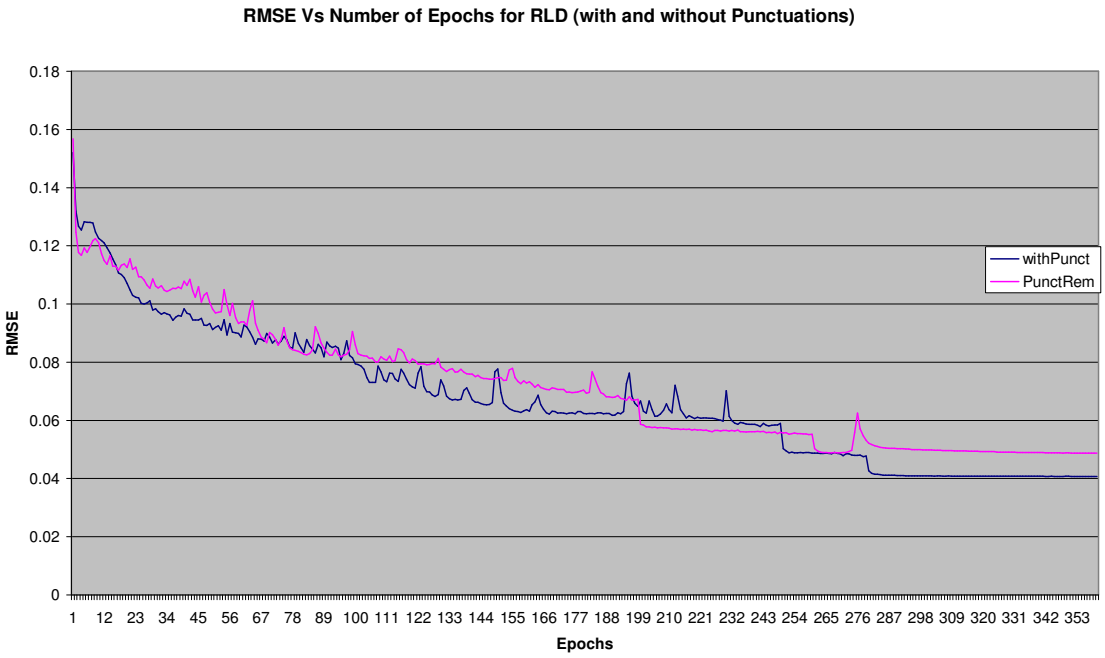

Figure 4.6: Plot of RMSE Against Number of Epochs for the RLD (WSJ Data)

To validate the RMSE training result for the RLD network, the network was tested after the 360th epoch. The test was carried out using both the cross-validation (containing 1201 sequences generated from 74 sentences) and the test (containing 15605 sequences generated from 1059 sentences) data sets. Results from this test indicate that the network came up with a sequence generalisation performance of 73.02% and 74.44% on the cross-validation and test data sets, respectively.

4.7 Phrase Recognition Performance

Training the PSR, with a network size of 50 hidden nodes for 500 epochs resulted in 99.84% of the 3103 patterns presented to the network being learnt. When the corresponding training data without punctuations were used in training the PSR (using the same network configuration and number of epochs), 99.70% of patterns were learnt.

A plot (figure 4.7) of the root mean square error (RMSE) for the PSR at each of the 500 epochs is observed. Also plotted, in this figure, is the RMSE for the PSR at these 500 epochs, using corresponding training data with punctuations removed from the training sentences. The comparison of these two cases (performance of the LRD on training sentences with punctuations and those with punctuations removed) is to highlight the effect of punctuations in the learning process. These plots detail the learning process for the PSR, showing RMSE curves (in both cases) with negative gradients; this indicates that the network is learning with each periodic presentation of the given data. The plots also show that the PSR performed better when punctuations were included in the sentences that when punctuations were.

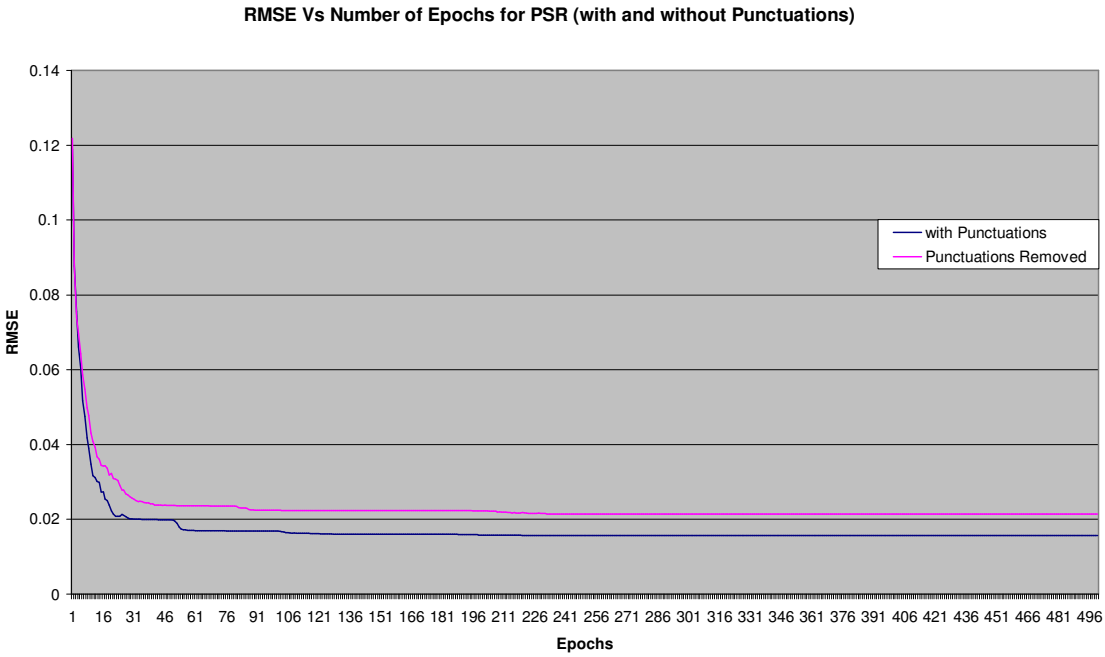

Figure 4.7: Plot of RMSE Against Number of Epochs for the PSR (WSJ Data)

To validate the RMSE training result for the PSR network, the network was tested after the 500th epoch. The test was carried out using both the cross-validation (consisting of 1191 patterns generated from 74 sentences) and the test (consisting

of 15317 patterns generated from 1059 sentences) data sets. Results from this test indicate that the network came up with a pattern generalisation performance of 95.05% and 93.36% on the cross-validation and test data sets, respectively.

4.8 Sentence Level Performance

Weights derived from the network training of the LRD, RLD and PSR were used in the parsing model to parse the three sentence sets: the training, cross-validation and test sets.

In addition to the PARSEVAL measures, Labelled Precision and Labelled Recall, used to assess sentence level performance, the F-Measure (the harmonic mean of labelled precision and labelled recall) is also used.

$$\text{F-Measure} = \frac{(2 \times \text{Labelled\_Precision} \times \text{Labelled\_Recall})}{(\text{Labelled\_Precision} + \text{Labelled\_Recall})}$$

Details of the sentence level results derived from the parser are shown in table 4.9.

Table 4.11: Sentence Level Results for the WSJ Corpus

Sentence Level Results							
	Sentences	**Words**	**Parsed**	**Exact Matches**	**Labelled Precision**	**Labelled Recall**	**F-Measure**
Training Set	202	3572	88.12%	20.30%	76.73	74.81	75.76%
Cross-validation Set	74	1382	87.84%	8.11%	60.16%	57.99%	59.06%
Test Set	1059	18722	85.74%	5.85%	60.21%	58.83%	59.51%

4.9 Discussion

After this parsing model was refined (in Chapter 3), trained and tested (with sentences that were not used in training) on the LPC, the parser performed with an average labelled precision/recall of 72.5%. On adapting this parser to the WSJ Corpus, it performed with an average labelled precision/recall of 59.52% when presented with test sentences not used during training (this set of test sentences being five times the size of the set of training sentences). It performed with an average labelled precision/recall of 75.77% when presented with the sentences used during training.

In digesting these results, the composition of longer sentences in the WSJ Corpus, compared to those in the LPC, is considered; on the average, sentences from the WSJ corpus generate nine times the number of LRD/RLD sequences generated from LPC sentences. Besides, syntactic tags used in annotating the WSJ corpus are of a coarser nature than those used for the LPC. Also considered is the fact that whereas punctuations were included in the WSJ Corpus sentences during training and testing, the data used from the LPC had all punctuations removed to simplify the parsing problem. Including punctuations add to the complexity of the parsing task. However, training results (as shown in sections 4.6 and 4.7) indicate that punctuations actually improve the performances of the three connectionist modules (LRD, RLD and PSR), thereby aiding decision making during parsing.

The reduced performance can therefore be attributed to the longer sentences in the WSJ corpus and the less coarse nature of the LPC tags, compared to the WSJ Corpus tags. This finer-grained nature of the LPC tags implies that they presented the parser with more information to make decisions with, during the parsing process. With these factors in mind, the parsing results for the WSJ Corpus

demonstrate that this parsing model is adaptable to the Wall Street Journal Corpus without its architecture or algorithm being changed.

By this adaptation, the parsing model takes advantage of its connectionism. Given its representation and modular architecture, which makes it independent of individual corpora, this parsing model should be adaptable to other corpora.

5. THE CORPUS-BASED PARSING MODEL: INTRODUCING LEXICAL SEMANTIC INFORMATION FOR NOUNS

5.1 Introduction

Semantic information is important in the resolution of common syntactic ambiguities during syntactic parsing [120]. However, contrasting theories on the use of semantic information during syntactic parsing exist. One body of research adopts the two-stage 'Fodorian' approach whereby semantic information (with other linguistic information) is considered during a second independent post-processing stage, after the syntactic information-only stage [10, 11, 12]; the other body of research, citing psycholinguistic evidence, adopts the multiple constraint-satisfaction process whereby syntactic and semantic information (as well as other linguistic information) are allowed to simultaneously interact (to varying degrees) during online syntactic processing[13, 14, 15, 16]. Considering the importance of semantic information during sentence processing and to gain an insight into these two contrasting theories of syntactic parsing, this chapter reports on the effect on the performance of full syntactic parsers, of integrating lexical semantic information with syntactic information in the parsing process. This integration of lexical semantic information with syntactic information is thought to be necessary for large-scale parsing of unconstrained natural language to be truly realisable and useful for practical applications.

In a bid to integrate lexical semantic information with syntactic information during parsing, it is necessary to extract lexical semantic features from large-scale resources. A host of lexical semantic resources exist. These include the Longman Dictionary of Contemporary English, the Longman Lexicon of Contemporary English, the Core Lexical Engine, Euro WordNet, CYC, EDL, WordNet, Cycorp, etc. WordNet [17], a generic lexical semantic network developed at Princeton University, U.S.A,

is used for the extraction of lexical semantic information needed for the investigation in this project. WordNet was chosen because of its availability, large coverage and taxonomy. In the course of this work, an algorithm has been developed to annotate nouns in the BLLIP (Brown Laboratory for Linguistic Information Processing) 1987-89 Wall Street Journal Corpus [90] with lexical semantic information extracted from WordNet. This algorithm defines a semi-automatic semantic tagging process.

The adaptability of the connectionist shift-reduce parsing model used in this project has been exploited in the process of integrating lexical semantic information with syntactic information. The parser (which has been shown in previous chapters to have been used successfully on both the Lancaster Parsed Corpus and the BLLIP 1987-89 Wall Street Journal Corpus [90]) has been extended to allow the combination of lexical semantic and syntactic representation in its input. When integrating the new lexical semantic tag representations (developed from the word sense tagging process) with the existing syntactic representations of the parser, the architecture of the parser has remained unchanged. Its three connectionist modules (Left-to-Right Delimiter network, Right-to-Left Delimiter network and Phrase Structure Recogniser network) remain in conjunction with its symbolic modules.

Data was generated for the different connectionist modules, with the combined linguistic information. The modules were then trained and tested. Linguistic knowledge, in the form of network weights, garnered from the network training processes was then used by the parser to syntactically analyse sentences.

Section 5.2 provides a description of WordNet, the on-line lexical reference system used for lexical semantic tagging. The new algorithm for the extraction of lexical semantic information from WordNet and the semantic annotation of nouns in the

BLLIP WSJ Corpus is described in section 5.3. In section 5.4, the tag representation scheme developed for the combination of semantic and syntactic knowledge and used as input to the parser is presented. Section 5.5 focuses on the performance of the connectionist modules of the parser, given the combination of lexical semantic and syntactic information. This performance is compared with the performance of the modules using only syntactic information as input in section 5.6. This comparison is done in a bid to assess the impact of each different piece of linguistic knowledge. Section 5.7 shows the performance of the phrase structure recogniser module of the parser, given the combination of lexical semantic and syntactic information. Again, the performance is compared with that obtained for the module without the combination. In section 5.8, the sentence-level performance of the parser is detailed. A comparison is made between this performance and the parser's performance before the combination of semantic and syntactic information. Section 5.9 presents a comparison of this work with other work that parses the Wall Street Journal corpus. The various performances are discussed in section 5.10.

5.2 WordNet

Wordnet [17] is an on-line lexical reference system which organises lexical information in terms of word meanings, rather than word forms. Word forms refer to the physical utterances or inscriptions of words; they are represented in WordNet in their familiar orthography. Word meanings refer to the lexicalised concept that a word can be used to express; they are represented in WordNet by synonym sets (synsets). In this reference system, English nouns, verbs, adjectives and adverbs are organised into synonym sets, each representing an underlying lexical concept.

Each synset in WordNet contains synonymous word forms, relational pointers, and other information. Different relations link the synonym sets. The relations

represented by the pointers in the synsets include hypernymy/hyponymy, antonymy, entailment, and meronymy/holonymy. Words that make up a synset are words which are equal or very close in meaning, for example, {plant, flora}. Hyperonyms are synsets which are the more general class of other synsets, for example, {mouth, muzzle} is a hyperonym of {beak, bill, neb}. Hyponyms are synsets which are particular kinds of a synset, for example, {beak, bill, neb} is an hyponym of {mouth, muzzle}. Antonyms are synsets which are opposite in meaning, for example, {man, adult man} and {woman, adult woman} are antonyms. Holonyms are synsets which are the whole of which another synset is a part, for example, {face, countenance} is a holonym of {mouth, muzzle}. Meronyms are synsets which are the parts of a synset, for example {flower, bloom, blossom} is a meronym of {angiosperm, flowering plant}.

The hypernymy/hyponymy relation provides the basis for the hierarchical semantic organisation of nouns in WordNet. This organisation takes into consideration the fact that definitions of common nouns typically provide a super-ordinate term and distinguishing features. Hyponyms are linked to their super-ordinates, and vice versa, in the WordNet database. WordNet is, therefore, a lexical inheritance system.

At the inception of the WordNet project about sixteen years ago, there were approximately 57,000 noun word forms organised into approximately 48,800 synsets. These numbers have grown since then; WordNet being an online database.

Nouns in WordNet are partitioned into a set of 25 generic semantic concepts, each treated as the unique beginner of a separate semantic hierarchy. These 25 hierarchies correspond to relatively distinct semantic fields, each having its own vocabulary. They vary widely in size and cover distinct conceptual and lexical

domains. The unique beginners are: {act, action, activity}, {animal, fauna}, {artefact}, {attribute, property}, {body, corpus}, {cognition, knowledge}, {communication}, {event, happening}, {feeling, emotion}, {food}, {group, collection}, {location, place}, {motive}, {natural object}, {natural phenomenon}, {person, human being}, {plant, flora}, {possession}, {process}, {quantity, amount}, {relation}, {shape}, {state, condition}, {substance}, and {time}. A description of these unique beginners is given in table 5.1.

All the nouns belonging to each of the 25 unique beginners are placed in one file. Three of these unique beginners: {animal, fauna}; {person, human being}; and {plant, flora}, are concerned with living things and can be grouped under {living thing, organism}. Four others: {artefact}; {food}; {natural object}; and {substance}, are concerned with non-living things and can be grouped under {non-living thing, object}. These two semantic concepts could be further converged under {thing, entity}. A 26th 'Tops' file is created in WordNet to include this extended semantic hierarchy.

Table 5.1: Unique Beginners for Nouns in WordNet

Unique Beginners	**Category**
Act	Nouns denoting acts or actions
Animal	Nouns denoting animals
Artefact	Nouns denoting man-made objects
Attribute	Nouns denoting attributes of people and objects
Body	Nouns denoting body parts
Cognition	Nouns denoting cognitive processes and contents
Communication	Nouns denoting communicative processes and contents
Event	Nouns denoting natural events
Feeling	Nouns denoting feelings and emotions
Food	Nouns denoting foods and drinks
Group	Nouns denoting groupings of people or objects
Location	Nouns denoting spatial position
Motive	Nouns denoting goals
Object	Nouns denoting natural objects (not man-made)
Person	Nouns denoting people
Phenomenon	Nouns denoting natural phenomena
Plant	Nouns denoting plants
Possession	Nouns denoting possession and transfer of possession
Process	Nouns denoting natural processes
Quantity	Nouns denoting quantities and units of measure
Relation	Nouns denoting relations between people or things or ideas
Shape	Nouns denoting two or three dimensional shapes
State	Nouns denoting stable states of affairs
Substance	Nouns denoting substances
Time	Nouns denoting time and temporal relations

5.3 Semantic Annotation of Nouns in The BLLIP WSJ Corpus

An important part of the process of investigating the effect on syntactic parsing, of providing shallow semantic information for nouns, is the semantic annotation of these nouns. In annotating the nouns in the BLLIP WSJ corpus, advantage was taken of the lexical inheritance system provided by WordNet: each noun in the WSJ sentence sets had information extracted from WordNet as to which of the topmost generic levels (unique beginners) it belonged.

A lot of the nouns are polysemous, and could have senses that belong to more than one of the unique beginners. In such cases, a maximum of four of the most frequently used senses are extracted from WordNet. The frequency of use for each sense (determined by the number of times a sense is tagged in the various semantic concordance texts built up as part of the WordNet project) is also extracted.

An algorithm, as depicted in the flow chart in figure 5.1, has been developed for the semantic annotation of nouns in the BLLIP WSJ corpus. Nouns in all the three sentence sets (training, cross-validation and test sets) that were previously set aside from the BLLIP WSJ Corpus have been semantically tagged. This annotation algorithm maps out a semi-automatic semantic tagging procedure.

As shown in figure 5.1, for each set of WSJ corpus sentences, one sentence is dealt with at a time. For each sentence, each word (and its tag) is picked up and

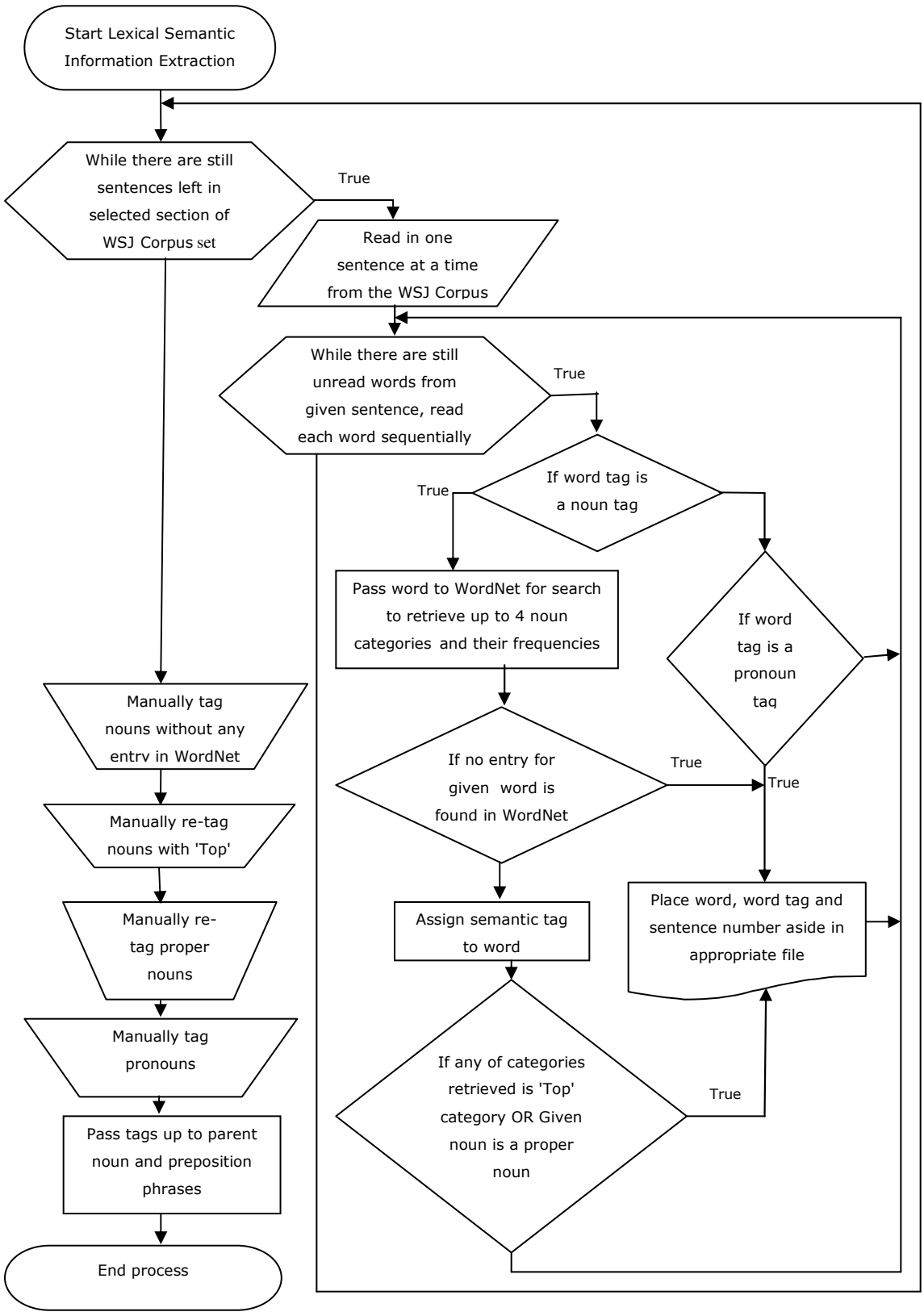

Figure 5.1: A flow-chart depicting the semantic annotation process for nouns from the BLLIP WSJ Corpus, using WordNet

evaluated sequentially until all the words in the sentence have been looked up. The syntactic tag for each word picked up is first checked to determine if the word is a noun or a pronoun. If a word is neither a noun nor a pronoun, it is ignored, and the next word in the sentence is evaluated. If the word looked up is a pronoun, it is stored (together with its tag and sentence number) in a dedicated file for manual tagging after the automatic phase of the annotation. If the word looked up is a noun, it is passed through a program to WordNet. This noun undergoes a search in WordNet, culminating in WordNet returning the unique beginner(s) (the topmost category where the noun's sense(s) is placed) for that noun. This retrieval of information from is implemented with the aid of a program which has been created in the course of this work to interface WordNet's library of functions, which in turn interface the WordNet database. Up to four unique beginners (senses) can be returned for each noun. WordNet also returns the frequency of use of each sense associated with the noun. Nouns with more than one sense have their categories returned in order of frequency (from the most frequent to the least frequent). If there is no entry in WordNet for a particular WSJ noun, it is stored (together with its tag and sentence number) in a dedicated file for manual tagging after the automatic phase of the annotation. Nouns with no corresponding senses in WordNet could be names of people, places or establishments. They could also be abbreviations. Wrongly spelt words would also not have any entries in WordNet (unless, the wrong spellings turned out to be correct spellings for some other word). Besides, words in combination words like "New" in "New York" have no senses in the noun section of WordNet, as they are not nouns. Most of these words (set aside to be tagged manually) can be easily tagged and the most common categories they fall into are "person", "location" or "group".

At the next processing stage in the semantic annotation procedure, nouns for which unique beginners are returned by WordNet have these unique beginners checked. If

any of the unique beginners returned for the noun is the "Top" category, it is stored (together with its syntactic tag and sentence number) in a dedicated file for manual re-tagging after the automatic phase of the annotation. The "Top" category is the "26^{th} category" for nouns in WordNet; it is a vague abstraction that attempts to pull all nouns into a single hierarchical memory structure. It has as its immediate hyponyms, senses (synsets) that are at the top of the other categories (unique beginners), like: {natural object}; {artefact}; {plant}; and {food}. The "Top" tag for each word is converted to any of the 25 categories which is the immediate hyponym in each case.

As part of the automatic phase of processing, all the proper nouns passed to WordNet are stored (together with their tags and sentence numbers) in a dedicated file. This enables them to be later manually assessed (and re-tagged, if need be). This is because proper nouns like names of people, places or organisations (e.g. Mr. Bank, Miss Stone) could easily be assigned wrong categories by WordNet.

After the automatic phase of semantic tagging, the four files containing nouns that need re-assessing and possible re-tagging are used to manually assess and tag some nouns. These four files contain, separately, nouns from the WSJ corpus (together with their tags and sentence numbers) without any entries in WordNet, nouns with the "Top" tag assigned as their semantic category, proper nouns and pronouns.

At this stage, the semantic tags (as shown in table 5.2) would be attached to the syntactic tags in the form "NN_act*5*_art*2*_pos*1*_qua*0*". This is the annotation for the noun "yield". It indicates that "yield" is polysemous and has the following senses: {act, action, activity} – frequency of 5; {artefact} – frequency of

2; {possession} – frequency of 1; and {quantity} – no occurrence in the semantic concordance texts. The tags are attached in order of frequency.

The semantic tags derived for the nouns are then ready to be converted to a binary form; to provide compatibility with the connectionist parser.

Table 5.2: Semantic Tags Used to Represent WordNet Unique Beginners

Unique Beginners	Tags
Act	act
Animal	ani
Artefact	art
Attribute	att
Body	bod
Cognition	cog
Communication	com
Event	eve
Feeling	fee
Food	foo
Group	gro
Location	loc
Motive	mot
Object	obj
Person	per
Phenomenon	phe
Plant	pla
Possession	pos
Process	pro
Quantity	qua
Relation	rel
Shape	sha
State	sta
Substance	sub
Time	tim

5.4 Tag Representation

The semantic tags assigned to the nouns from the implementation of the annotation algorithm described in section 5.3 are symbolic tags. To enable the integration of the semantic information inherent in these tags into the existing syntactic information used by the connectionist parser, these symbolic word sense tags need to be converted to binary vector representations. The parser's input also needs to be adapted to accommodate the semantic representation together with the syntactic information it already receives.

5.4.1 Semantic Tag Representation

In designing binary input representations for the word senses (unique beginners), the number of high-level sense categories to be represented (twenty-five) and the frequency of each sense (for polysemous words) are considered. 25 bits are used to represent this lexical semantic information. Each of the 25 high-level sense categories is assigned its "bit space", as shown in figure 5.2.

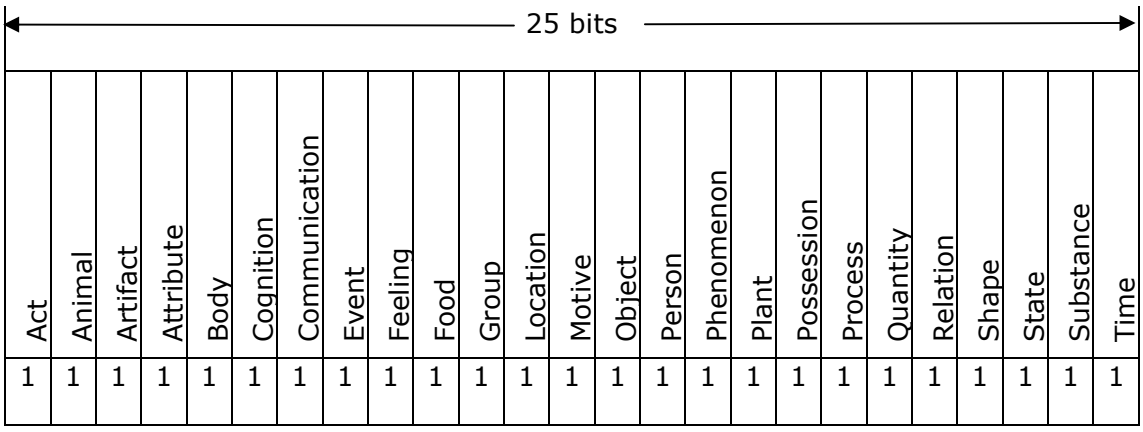

Figure 5.2: Bit Space Allocation to WordNet Senses

For each noun, the value for the bit corresponding to each sense is 0, unless the sense is one of the senses that define the noun. The values for the senses that define the given noun range from 0.25 to 1, depending on the frequency of the sense for the given noun. The values are, however, initially represented with the letters 'A', 'B', 'C', and 'D' (with 'A' indicating the most frequent category and 'D', the least frequent). The values that these bit representation symbols denote are shown in table 5.3. The actual values, as shown in table 5.3, are fed as input to the neural networks.

Table 5.3: Semantic Representation Values

Semantic Bit Representation Symbol	Value
A	1
B	0.75
C	0.5
D	0.25

Before assigning the 'A' – 'D' symbols, the extracted frequency value for each sense is compared with the value for the next most frequent sense. A frequency difference measure, FDM is calculated as follows:

$$FDM = (1 - f_v / f_{next}) \times 100\%$$

where f_v = frequency value

and f_{next} = frequency value of next most frequent sense

This measure has been designed to capture cases where the frequencies of occurrence of different senses for a word are very close to each other. If the FDM is

less than 10%, the sense with the frequency value being assessed is assigned the same frequency symbol, say 'A', as that assigned the next most frequent sense.

For the "yield" example first given in section 5.3, the semantic tag annotation is then changed from the form, "NN_act*5*_art*2*_pos*1*_qua*0*" into the form NN_A0B00000000000000C0D00000.

The semantic representation is, at this stage, ready to be integrated with the syntactic representation for the nouns.

5.4.2 Integrating Syntactic and Semantic Representation

The binary input representation adopted for denoting the BLLIP WSJ syntactic tags in this work uses 46 bits with bit spaces created for terminal and non-terminal tags. The first 19 bits of the 46-bit input space are used to represent word tags, while the next 27 bits are used to represent constituent tags. To incorporate the extracted lexical semantic information into this word tag representation, 25 additional bit spaces were added to the input bit representation. These 25 bit spaces (placed immediately after the 27 bit spaces used for constituent tags) cater for the 25-bit semantic representation adopted in sub-section 5.4.1 above. This results in a 71-bit input space for all word and constituent tags. Non-noun tags have all 0's as their last 25 bits.

The representation for the word tag, NN_A0B00000000000000C0D00000 for "yield" would then be:

00000001000010000000000000000000000000000000000A0B00000000000000C0D00000

The semantic tags for nouns, pronouns, noun phrases and preposition phrases that constitute noun (NP) or preposition phrases (PP) are then incorporated into parent

noun and preposition phrases. This is to ensure that the semantic properties are not lost during the initial stages of shift-reduce parsing. It is envisaged that this stage of the process will help resolve preposition attachment cases in sentences such as "the boy ate the pasta with the sauce" and "the boy ate the pasta with the fork". In the first sentence, the {food} category returned by WordNet for "sauce" is transferred to the noun phrase, "the sauce" and further on to the preposition phrase, "with the sauce". There would, therefore, be a difference in input representations to the parser for the prepositional phrases, "with the sauce" and "with the fork".

In semantically tagging a noun or preposition phrase, the configuration of the 25-bit section of its tag representation depends on the configurations of the semantic sections of the tag representations that denote the nouns, pronouns, noun phrases and preposition phrases that constitute the given noun or preposition phrase. For each bit space in the semantic tag representation of the given noun or preposition phrase, if the values of all the constituting tags in a similar bit position are all '0', the bit value for that position would be '0'. If the value for any of the constituting tags in a similar position is 'A', 'B', 'C' or 'D', the highest value ('A' > 'B' > 'C' > 'D' > '0') becomes the value for that position.

To further aid decision making during parsing, the head nouns of all noun phrases are attached to the noun phrases during the shift-reduce parsing process. In cases where the head noun of a noun phrase is itself a noun phrase, the head noun of the head noun constituent is carried along.

With the semantic representations integrated into the syntactic representations for the WSJ corpus, the left-to-right and right-to-left delimiter and phrase structure recogniser data sets were generated, trained and tested.

5.5 Phrase Segmentation Performance

The training data generated from the integration of lexical semantic and syntactic tag representations were trained and tested (for training and generalisation performances) for the delimiter modules of the parser. Given that the delimiter modules handle the most complex aspects of the parsing problem, temporal sequence processing, their performance with the integrated input is critical to the ability of the parser to analyse the new data.

The LRD network was initially trained for 1800 epochs, with an empirically determined network size of 165 hidden nodes. In determining the network size, training was initially attempted with the same number of hidden nodes (105 hidden nodes) that was used to train the LRD with an input representation that contained only syntactic information. As expected, given the increased complexity of the training data, the network with these initial hidden nodes could not learn. The number of hidden nodes was then successively increased by 20 until the network started learning. The initial number of hidden nodes (165 hidden nodes) used for training is therefore the minimum number of hidden nodes at which the fully-connected TASRN network started learning. After 1800 epochs of training the LRD network, 95.53% of the 16,492 sequences in the training set had been learnt. To improve the training performance of the network, it was trained for 200 more epochs. After 2000 epochs, 95.66% of the 16,492 sequences in the training set had been learnt. However, generalisation performance on the cross-validation set was 79.94% and 80.17% for both training times.

In a bid to search for the optimal network size for the LRD network, the number of hidden nodes was increased to 205. The content of the LRD data set was also

reconfigured using equation 4.2 (a combination of one balanced set and two unbalanced sets), set out in section 4.5. The enlarged network was initially trained for 600 epochs, and then to 800 epochs in an attempt to achieve some improved training performance. Of the 19,845 sequences in the reconfigured training set, 94.69% and 95.13% were learnt by the LRD after 600 and 800 epochs, respectively.

A plot (figure 5.3) of the root mean square error (RMSE) for the LRD network is observed for training carried out with the two different network sizes. This plot shows the network needing fewer epochs to converge with the larger network size (205 hidden nodes). However, in determining the best weight vector to be used as the LRD component of the parser, the performance measure used was the sequence generalisation performances of the five weight vectors obtained on the cross-validation set. These generalisation performances are shown in figure 5.4. The weight vector obtained from the network trained with 205 hidden nodes after 700 epochs was the best available weight vector with a sequence generalisation of 82.3949%; it was therefore chosen as the LRD component of the parser.

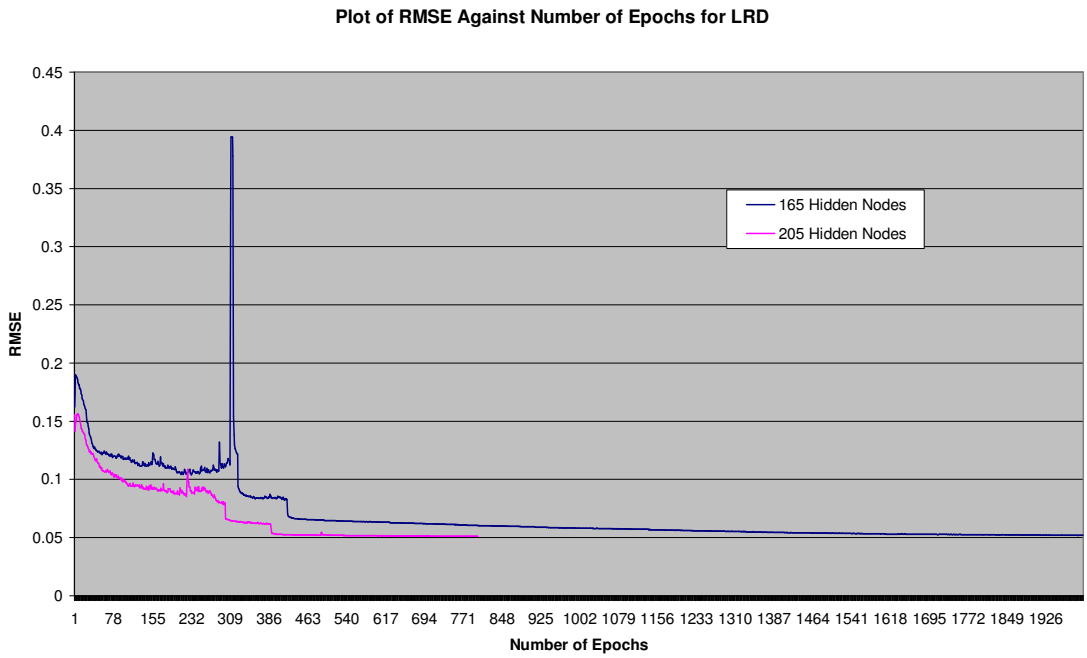

Figure 5.3: Plot of RMSE Against Number of Epochs for the LRD (WSJ Data)

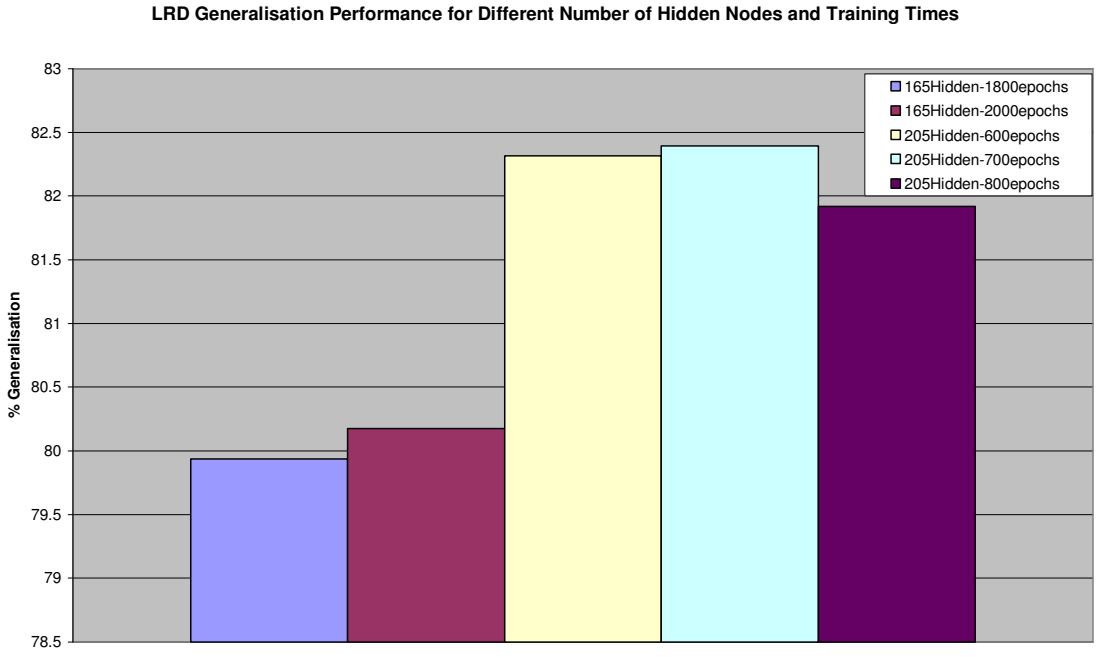

Figure 5.4: RLD Generalisation Performance for Different Number of Hidden Nodes and Training Times (WSJ Data)

Details of the training results (including number of connections, hidden nodes and % sequences learnt) for the chosen LRD network, are displayed, alongside those of the RLD, in table 5.7. Table 5.4 indicates a breakdown of the training performance of the LRD network according to sequences of different lengths. While the LRD network learnt all of the sequences of length 11 to length 13, it had short-comings in learning sequences of length 7 to length 10. Sequences of length 7 were the ones where the LRD network had the greatest learning challenge, with only 86.99% of the 3137 of such sequences learnt.

Apart from being tested on the cross-validation data set (containing 1261 sequences generated from 74 sentences), LRD network was also tested on the test data set (containing 16786 sequences generated from 1059 sentences). Results from this test indicate that the network came up with a sequence generalisation of 82.04% on the test data sets, compared to the sequence generalisation of 82.39% it achieved on the smaller cross-validation data set. Considering the difference in size of the two sentence samples used in generating the data sets, this difference in performance of only 0.35% indicates that the network could possibly generalise at the same level to any data set not used in training but generated from the corpus used in training, irrespective of the size of the data set.

Table 5.4: Training Results (for sequences of different lengths) for the LRD

Sequence Length	**7**	**8**	**9**	**10**	**11**	**12**	**13**
No. of Sequences	3137	5631	3017	2225	2005	1921	1909
% Sequences Learnt	86.99	93.23	94.90	98.92	100	100	100

Further attempts at optimising the number of hidden nodes for the LRD network would require experimenting with higher number of hidden nodes than 205. However, due to the computational complexity of training the Temporal Auto-Associative Simple Recurrent Network used for the LRD (given the size of the training data sets, number of parameters and the case that the rate of convergence in back-propagation learning tends to be relatively slow; making it computationally excruciating [94, 134]) and the computer resource (Intel Pentium 4 CPU 1.70 GHz; 1.70GHz, 1 GB of RAM) available for this project, it is not feasible to exhaustively optimise the network. Attempts to further increase the number of synaptic weights for this network would lead to impractical training times (table 5.5).

Table 5.5: Training Times for the LRD Network

Number of Hidden Nodes	Number of Network Connections (Weights)	Number of Training Sequences	Number of Training Patterns	Time needed for training (days)	Approximate Number of Epochs
165	78447	16492	158862	95	2000
205	113847	19845	186334	285*	2000
225	133947	19845	186334	335†	2000

* Estimated from training carried out with 205 hidden nodes

† Estimated for training yet to be undertaken (using training times for other network configurations)

The RLD network was initially trained, with an empirically determined network size of 265 hidden nodes for 500 epochs. As with the LRD network, in determining the initial network size for the RLD network, training was firstly attempted with the same number of hidden nodes (165 hidden nodes) that was used to train the RLD network with an input representation that contained only syntactic information. As

envisaged, given the increased complexity of the training data, the network with these initial hidden nodes could not learn. The number of hidden nodes was then successively increased by 20 until the network started learning. The number of hidden nodes used (265 hidden nodes) is therefore the initial minimum number of hidden nodes at which the fully-connected TASRN network started learning the right-to-left delimitation task.

After 500 epochs of training the RLD network, 91.61% of the 15,308 sequences were learnt. In a bid to determine the optimal network size for the RLD network, the number of hidden nodes was increased twice to 285 and 305. The network training with 285 hidden nodes had to be truncated early after 152 epochs when it was observed that its training performance was not significantly different from that of the network with 265 hidden nodes. Training of the RLD network with 305 epochs led to 94.78% of the 15,308 sequences being learnt, after 400 epochs of training.

A plot (figure 5.5) of the root mean square error (RMSE) for the RLD network is observed for training carried out with the three different network sizes. This plot shows the networks displaying better training ability (lower RMSE) the larger the network size. However, in determining the best weight vector to be used as the RLD component of the parser, the performance measure used was the sequence generalisation performances of the two weight vectors (for hidden node sizes of 265 and 305) obtained on the cross-validation set; the network with 285 hidden nodes was not trained for long enough to be considered. These generalisation performances are shown in figure 5.6. The weight vector obtained from the network trained with 305 hidden nodes after 400 epochs was the best available weight vector with a sequence generalisation of 74.29%; it was therefore chosen as the RLD component of the parser.

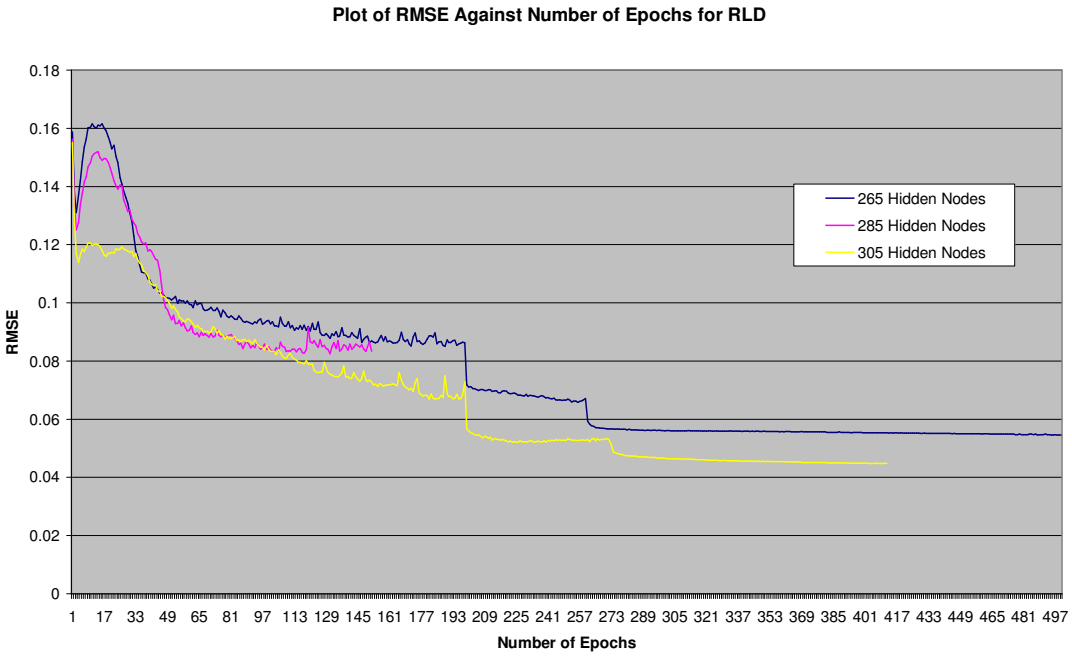

Figure 5.5: Plot of RMSE Against Number of Epochs for the RLD (WSJ Data)

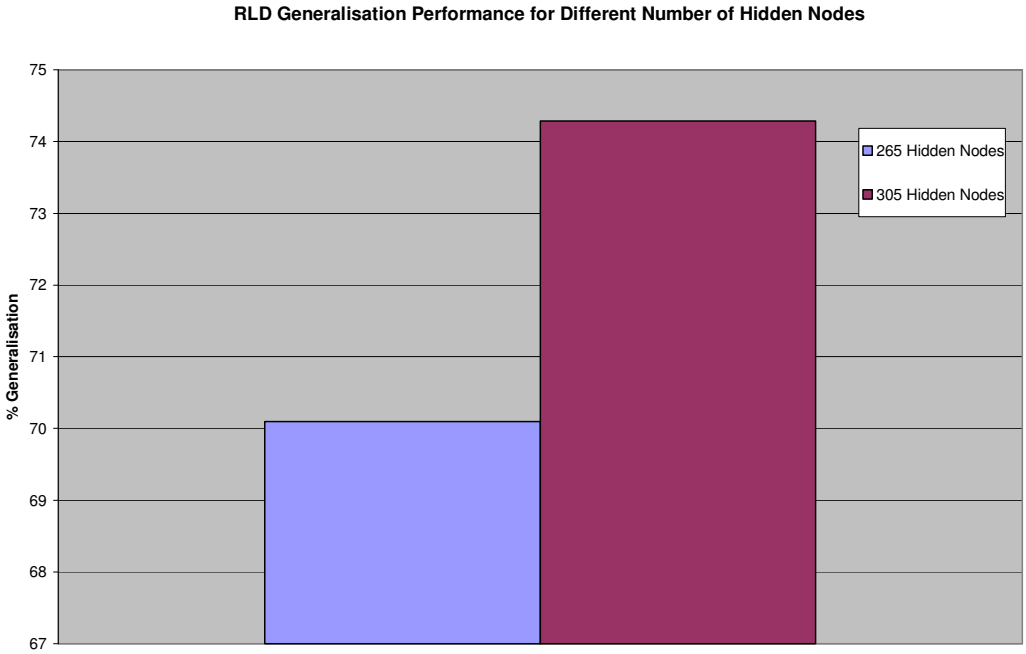

Figure 5.6: RLD Generalisation Performance for Different Number of Hidden Nodes

Details of the training results (including number of connections, hidden nodes and % sequences learnt) for the chosen RLD network, are displayed, alongside those of the LRD, in table 5.7. Table 5.6 indicates a breakdown of the training performance of the RLD network according to sequences of different lengths. While the RLD network learnt all of the sequences of lengths 8, 16 and 17, it had short-comings in learning sequences of length 9 to length 15. Sequences of lengths 10 and 11 were the ones where the RLD network had the greatest learning challenge; only 86.84% of the 2386 sequences with length 10 were learnt while 88.93% of the 1915 sequences with length 11 were learnt.

Besides being tested on the cross-validation data set (containing 1264 sequences generated from 74 sentences), the RLD network was also tested on the test data set (containing 17221 sequences generated from 1059 sentences). Results from this test indicate that the network came up with a sequence generalisation of 72.67% on the test data sets, compared to the sequence generalisation of 74.29% it achieved on the smaller cross-validation data set. Considering the difference in size of the two sentence samples used in generating the data sets, this difference in performance of 1.62% indicates that the network could possibly generalise at the same level to any data set not used in training but generated from the corpus used in training, irrespective of the size of the data set.

Table 5.6: Training Results (for sequences of different lengths) for the RLD

Length of Sequence	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**
No. of Sequences	1207	1281	2386	1915	1699	1575	1482	1313	1241	1209
% Sequences Learnt	100	97.74	86.84	88.93	94.11	94.86	97.23	98.32	100	100

Table 5.7: Training Results for the LRD and RLD

	Hidden Nodes	Connections	No. of Patterns	No. of Sequences	Epochs	RMS Error	% Pat. Learnt	% Seq. Learnt
RLD	305	230,347	187,825	15,308	400	0.0381275	91.42	94.78
LRD	205	113,847	186,334	19,845	800	0.048689	88.83	95.13

As with the LRD network, further attempts at optimising the number of hidden nodes for the LRD network would require experimenting with higher number of hidden nodes than 305. However, due to the computational complexity of training the network and the computer resource (Intel Pentium M CPU 1.73 GHz; 1.73GHz, 1 GB of RAM) available for this project, it is not feasible to exhaustively optimise the network. Attempts to further increase the number of synaptic weights for this network would lead to impractical training times (table 5.8).

Table 5.8: Training Times for the RLD Network

Number of Hidden Nodes	Number of Network Connections (Weights)	Number of Training Sequences	Number of Training Patterns	Time needed for training (days)	Approximate Number of Epochs
265	178947	15308	187825	43	500
285	203847	15308	187825	78*	500
305	230347	15308	187825	118*	500
325	258447	15308	187825	133†	500

* Estimated from training carried out with 285 and 305 hidden nodes

† Estimated for training yet to be undertaken (using training times for other network configurations)

5.6 Effect of Integrating Lexical Semantic and Syntactic Representation on Phrase Segmentation Performance

Table 5.9 shows a comparison of training results for the delimiter networks (LRD and RLD). This compares training results achieved using a combination of lexical semantic and syntactic input representation with training results achieved using only syntactic input representation. Larger network sizes were required to train the delimiter networks using the combined linguistic information because of the additional information they had to process.

For both delimiters, the networks trained on the combination of lexical semantic and syntactic input representation (compared to the networks trained on only syntactic input representations) dealt with more complex tasks (using a greater number of network connections) and had to be trained for longer periods in terms of number of epochs and actual training time. They also learnt slightly lower proportions of the sequences presented to them, with both the LRD and RLD networks learning over 94.5% of these sequences in both input representation cases. The differences in proportions of patterns learnt between the networks trained on the combination of semantic and syntactic input representation and those trained on only syntactic input representation was very slight (less than 0.5% in both cases – 0.46% for the LRD and 0.13% for the RLD). The training data sets (with the integrated semantic and syntactic input representation) for both delimiter networks were larger than the sets generated with only syntactic input representation; the integration of lexical semantic and syntactic information having resulted in fewer sequence replications and conflicts.

Table 5.9: Comparison – Training Results for the Delimiter Networks for Input Representations with Syntactic-only and a Combination of Semantic and Syntactic Information

	LRD		RLD	
	Syntactic-Only	**Semantic + Syntactic**	**Syntactic-Only**	**Semantic + Syntactic**
No. of Hidden Nodes	105	205	165	305
No. of Connections	32,072	113,847	70,172	230,347
No. of Patterns	142,840	186,334	175,115	187,825
No. of Sequences	14,839	19,845	14,268	15,308
No. of Epochs	500	800	360	400
RMS Error	0.0457812	0.0486.89	0.0407269	0.0381275
% Pattern Learnt	89.29	88.83	91.55	91.42
% Sequence Learnt	96.97	95.13	96.31	94.78

A breakdown (based on sequence lengths) of training comparison between the two different input representations fed to the LRD (as shown in table 5.10) indicate that with sequence lengths of 8 to 13, both input representations achieve about the same training performances. However, with sequences of length 7, there is 9.69% difference in training performance; the LRD network with a combination of lexical semantic and syntactic input representation learns only 86.99% of these sequences. Given that LRD sequences make use three look-back symbols and three

look-ahead symbols, sequences of length 7 would normally be phrases with only one word or constituent.

Table 5.10: Comparison – LRD Training Results (for sequences of different lengths) for the Delimiter Networks for Input Representations with Syntactic-only and a Combination of Semantic and Syntactic Information

Sequence Length	No. of sequences		% Sequences Learnt	
	Syntactic-Only	Semantic + Syntactic	Syntactic-Only	Semantic + Syntactic
7	2290	3137	96.68	86.99
8	3384	5631	92.38	93.23
9	2179	3017	95.27	94.90
10	1839	2225	99.35	98.92
11	1731	2005	100	100
12	1712	1921	100	100
13	1704	1909	100	100

A plot (figure 5.7) of the root mean square error (RMSE) for the LRD network is observed for training carried out with the two different input representations. This plot shows the network needing fewer epochs to converge when syntactic-only input is used. This indicates that the LRD network with syntactic-only input has parameters that enable it to deal more comfortably with its task, compared to the network with a combination of lexical semantic and syntactic input representation.

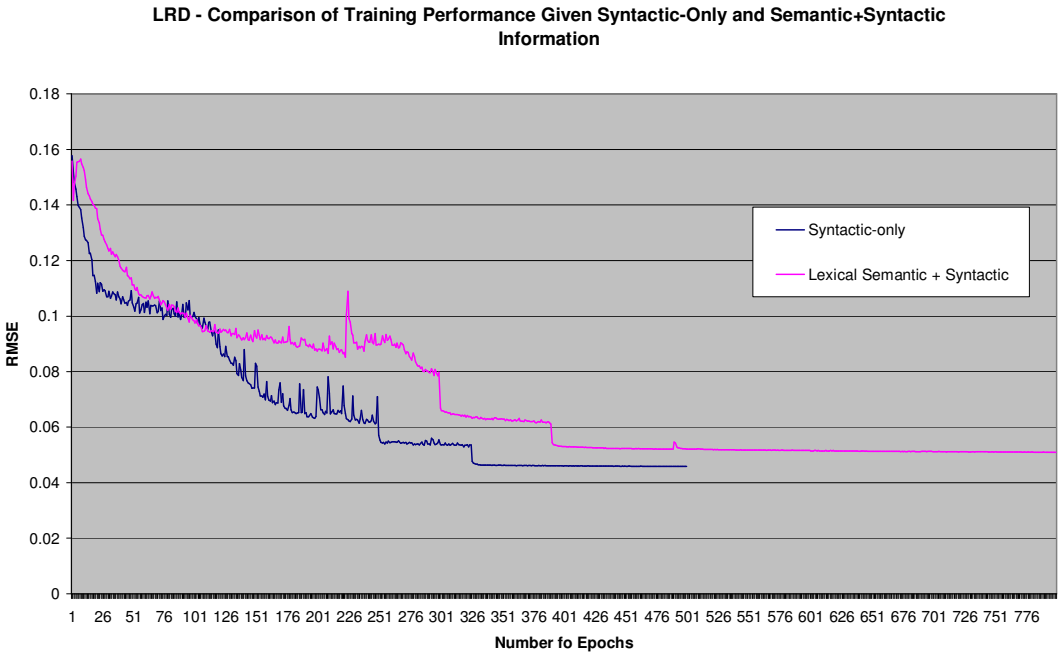

Figure 5.7: Plot Comparison of Training Performance for the LRD, Given Syntactic-Only and Semantic + Syntactic Information

A breakdown (based on sequence lengths) of training comparison between the two different input representations fed to the RLD (as shown in table 5.11) shows that learnt sequences of length 8, 16 and 17 irrespective of the input representation. Apart from sequences of these three sequence lengths, the RLD network trained on syntactic-only input performed better on sequences of all but one (sequence length 13) sequence lengths. On the whole, the performances of the RLD network on sequences of different lengths given the two input representations followed a very similar trend.

Table 5.11: Comparison – RLD Training Results (for sequences of different lengths) for the Delimiter Networks for Input Representations with Syntactic-only and a Combination of Semantic and Syntactic Information

Sequence Length	No. of sequences		% Sequences Learnt	
	Syntactic-Only	Semantic + Syntactic	Syntactic-Only	Semantic + Syntactic
8	1119	1207	100	100
9	1119	1281	100	97.74
10	2222	2386	92.44	86.84
11	1780	1915	90.62	88.93
12	1585	1699	95.14	94.11
13	1460	1575	93.50	94.86
14	1386	1482	98.56	97.23
15	1231	1313	100	98.32
16	1159	1241	100	100
17	1127	1209	100	100

A plot (figure 5.8) of the root mean square error (RMSE) for the RLD network is observed for training carried out with the two different input representations. Although the network seems to have a slight edge when syntactic-only input is used, this plot shows the network having a similar learning pattern with both input representations during training.

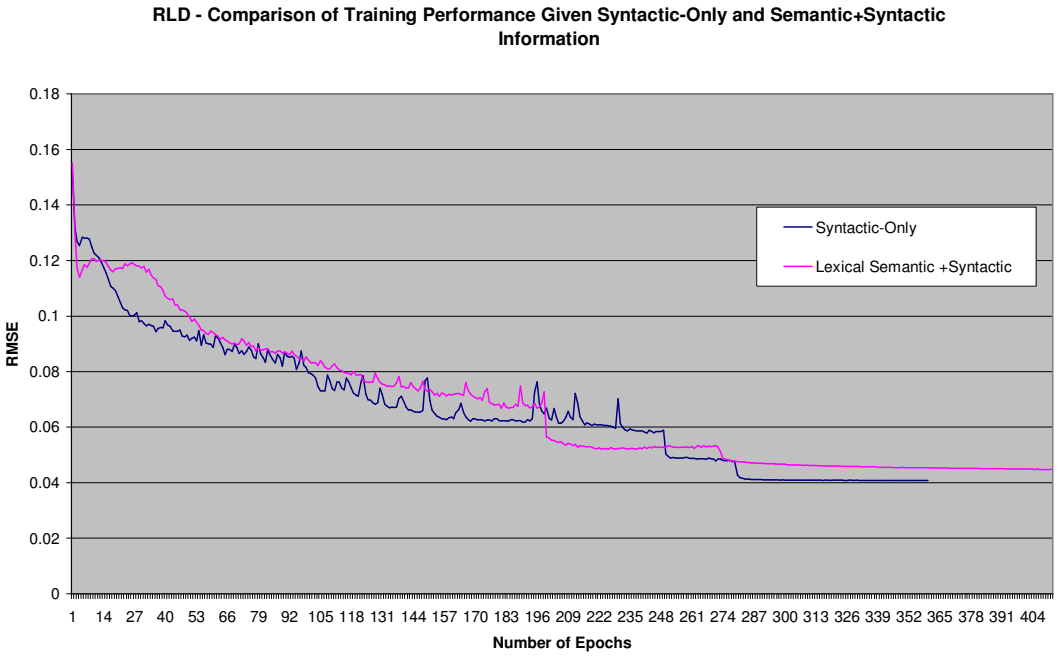

Figure 5.8: Plot Comparison of Training Performance for the RLD, Given Syntactic-Only and Semantic + Syntactic Information

A comparison of the generalisation performances (table 5.12) of the delimiter networks, using two test sets, reveals very close performances when both sets of input representations are used. The LRD network with syntactic-only input has a 1.61% better generalisation performance on the cross-validation set (generated from 74 sentences) than the same network with a combination of lexical semantic and syntactic input representation. However, with the test set (generated from 1059 sentences), the LRD network with a combination of lexical semantic and syntactic input representation produces a 1.99% better performance than the same network with syntactic-only input representation. With the RLD, the network with a combination of lexical semantic and syntactic input representation has a 1.27% better generalisation than the same network with syntactic-only representation when tested on the cross-validation set. On the other and, the RLD network with syntactic-only representation has a 1.77% better performance when the test set is used for testing. From the foregoing, for each of the delimiter networks, the use of

a particular set of input representations produces slightly better generalisation performance depending on which of the two test sets is used. The generalisation performance of each delimiter network when lexical semantic information is added appears to be at par with its performance when only syntactic information is used

Table 5.12: Comparison – Generalisation Performance for the Delimiter Networks for Input Representations with Syntactic-only and a Combination of Semantic and Syntactic Information

	LRD		RLD	
	Syntactic-Only	**Semantic + Syntactic**	**Syntactic-Only**	**Semantic + Syntactic**
% Generalisation on Cross-validation Set (from 74 sentences)	84.00	82.39	73.02	74.29
% Generalisation on Test Set (from 1059 sentences)	80.05	82.04	74.44	72.67

5.7 Phrase Recognition Performance

The training data generated from the integration of lexical semantic and syntactic tag representations were also trained and tested (for training and generalisation performances) for the phrase recognition module of the parser. Training the PSR, with an initial network size of 50 hidden nodes for 2000 epochs resulted in 99.91% of the 3349 patterns presented to the network being learnt. In a search for an

optimal network size for the PSR, three more network sizes (60, 70 and 90 hidden nodes) were experimented with. The PSR networks with 60, 70, and 90 hidden nodes learnt 99.88, 99.94, and 99.91 of the 3349 patterns presented to them, respectively. A plot (figure 5.9) of the root mean square error (RMSE) for the four PSR networks shows a similar convergence, with the network made of 90 hidden nodes exhibiting lower root mean square error. However, in determining the best weight vector to be used as the PSR component of the parser, the performance measure used was the pattern generalisation performances of the four weight vectors obtained on the cross-validation set. These generalisation performances are shown in figure 5.10. The weight vector obtained from the network trained with 70 hidden nodes produced the best pattern generalisation of 90.71%; it was therefore chosen as the PSR component of the parser.

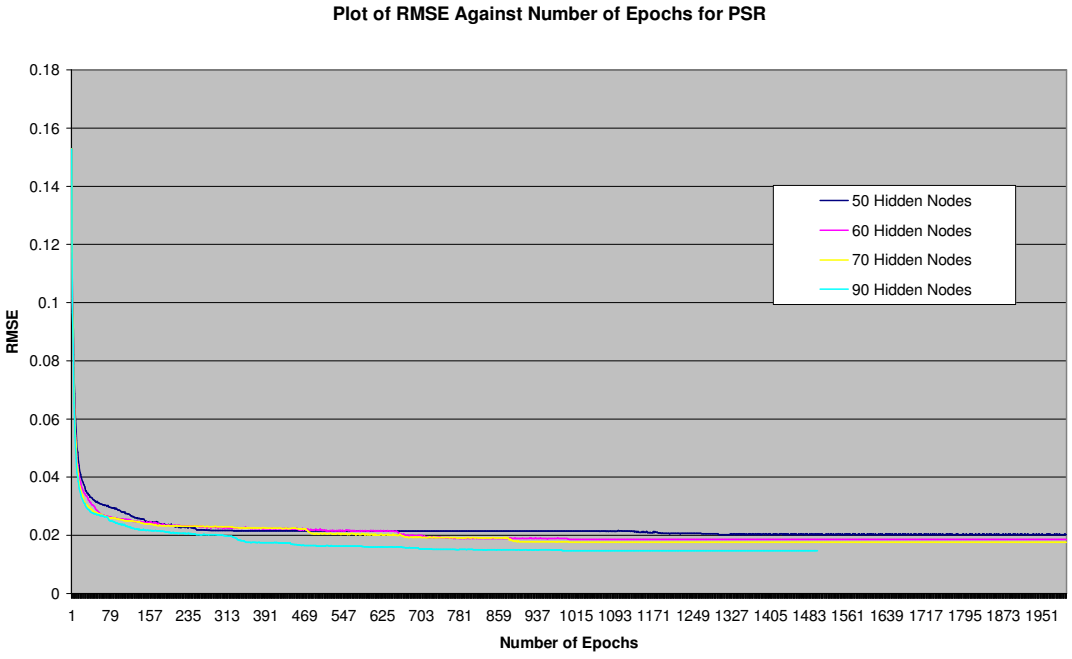

Figure 5.9: Plot of RMSE Against Number of Epochs for the PSR (WSJ Data)

Besides being tested on the cross-validation data set (consisting of 1259 patterns generated from 74 sentences), the PSR network was also tested on the test data set (consisting of 16890 patterns generated from 1059 sentences). Results from this test indicate that the network came up with a pattern generalisation of 88.93% on the test data set, compared to the pattern generalisation of 90.71% it achieved on the smaller cross-validation data set; a small difference of 1.78% when the differences in sizes of the two data sets are considered.

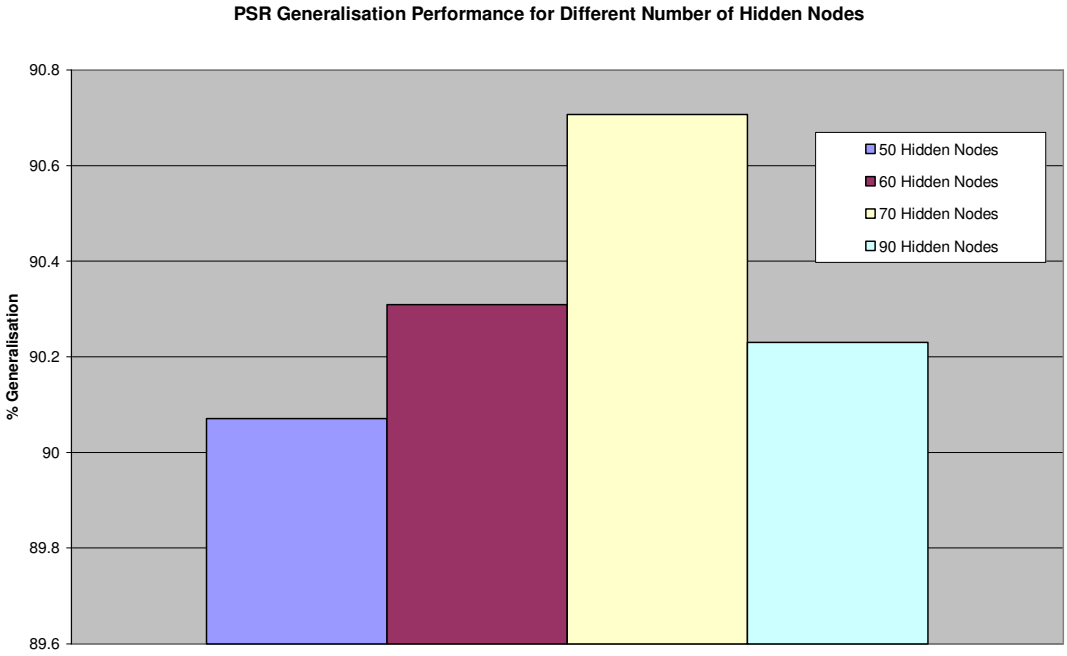

Figure 5.10: PSR Generalisation Performance for Different Number of Hidden Nodes

Figure 5.11 shows a plot comparing the learning curve of the PSR network when fed with a combination of lexical semantic and syntactic input representation and when fed with only syntactic input representation. Although the PSR network with syntactic-only input representation converges much faster (in terms of number of epochs), the network with a combination of lexical semantic and syntactic input representation gradually achieves the same minimum error.

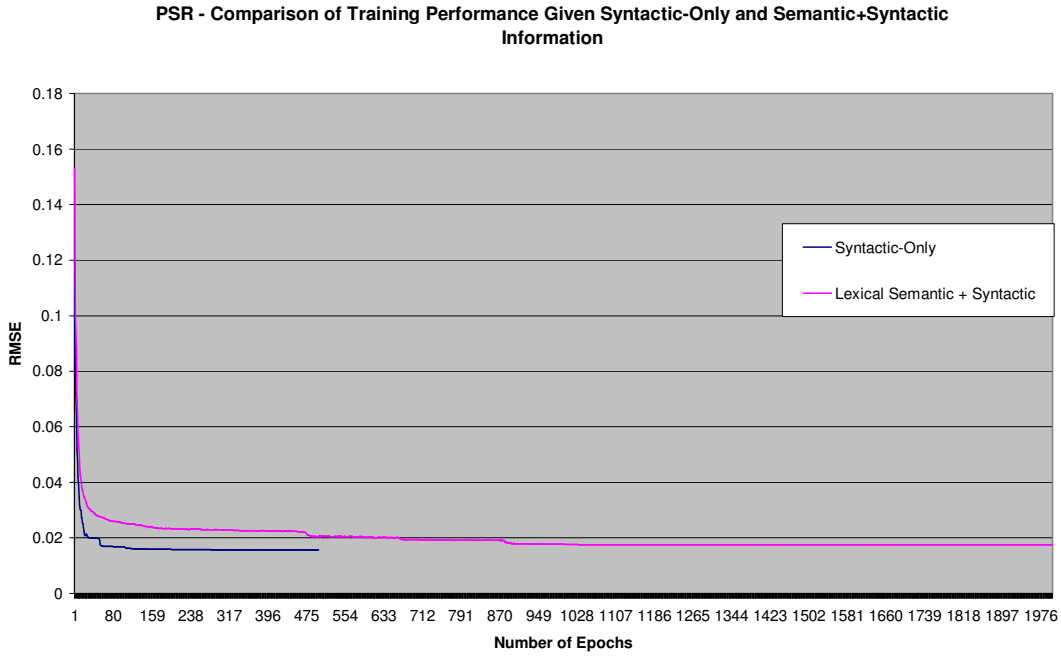

Figure 5.11: Plot Comparison of Training Performance for the PSR, Given Syntactic-Only and Semantic + Syntactic Information

A comparison of the generalisation performance (table 5.13) of the phrase recognition network, using two test sets, shows that the PSR performs better when only syntactic information is used than when a combination of lexical semantic and syntactic information is used.

Table 5.13: Comparison – Generalisation Performance for the PSR Network for Input Representations with Syntactic-only and a Combination of Semantic and Syntactic Information

	PSR	
	Syntactic-Only	**Semantic + Syntactic**
% Generalisation on Cross-validation Set (from 74 sentences)	95.05	90.71
% Generalisation on Test Set (from 1059 sentences)	93.36	88.93

5.8 Sentence Level Performance

Synaptic weights derived from the network training of the LRD, RLD and PSR store the linguistic knowledge acquired by the connectionist parser. They were used in the parsing model to parse the three sentence sets: the training, cross-validation and test sets.

Details of the sentence level results derived from the parser are shown in table 5.13. Table 5.14 shows a comparison of these results with those obtained using only syntactic information.

Table 5.14: Sentence Level Results for the WSJ Corpus (Lexical Semantic + Syntactic Input Representations)

Sentence Level Results							
	Sentences	**Words**	**Parsed**	**Exact Matches**	**Labelled Precision**	**Labelled Recall**	**F-Measure**
Training Set	206	2903	68.45%	17.48%	73.06%	74.14%	73.60%
Cross-validation Set	74	1092	54.05%	8.11%	57.44%	58.93%	58.17%
Test Set	1059	12551	60.72%	5.19%	55.78%	57.76%	56.75%

The sentence level results have kept improving with positive modifications to the component connectionist modules. When sentences from the training set were parsed with a connectionist module configuration that comprised an LRD (165 hidden nodes after 1800 epochs), RLD (265 hidden nodes) and PSR (50 hidden

nodes), an F-Measure of 69.26% was achieved. This rose to 73.45% when the configuration was changed to [LRD (165 hidden nodes after 2000epochs), RLD (305 hidden nodes), PSR (70 hidden nodes)]. The F-Measure improved further to 73.60% with the present connectionist module configuration [LRD (205 hidden nodes), RLD (305 hidden nodes), PSR (70 hidden nodes)]. This indicates a lot of room for improvement of the sentence level performance if the connectionist modules are further optimised.

As with the training set, when sentences from the cross-validation set were parsed with a connectionist module configuration that comprised an LRD (165 hidden nodes after 1800 epochs), RLD (265 hidden nodes) and PSR (50 hidden nodes), an F-Measure of 52.79% was achieved. This went up to 54.57% when the configuration was changed to [LRD (165 hidden nodes after 2000epochs), RLD (305 hidden nodes), PSR (70 hidden nodes)]. The F-Measure improved further to 58.17% with the present connectionist module configuration [LRD (205 hidden nodes), RLD (305 hidden nodes), PSR (70 hidden nodes)].

When sentences from the test set were parsed with a connectionist module configuration that comprised an LRD (165 hidden nodes after 1800 epochs), RLD (265 hidden nodes) and PSR (50 hidden nodes), an F-Measure of 55.06% was achieved. This improved to 55.15% when the configuration was changed to [LRD (165 hidden nodes after 2000epochs), RLD (305 hidden nodes), PSR (70 hidden nodes)]. The F-Measure improved further to 56.75% with the present connectionist module configuration [LRD (205 hidden nodes), RLD (305 hidden nodes), PSR (70 hidden nodes)].

Table 5.15: Sentence Level Comparison for the WSJ Corpus (Syntactic-Only Vs Lexical Semantic + Syntactic Input Representations)

Sentence Level Comparison							
	Sentences	**Words**	**Parsed**	**Exact Matches**	**Labelled Precision**	**Labelled Recall**	**F-Measure**
Training Set(Semantic + Syntactic)	206	2903	68.45%	17.48%	73.06%	74.14%	73.60%
Training Set(Syntactic Only)	202	3572	88.12%	20.30%	76.73%	74.81%	75.76%
Cross-Validation Set(Semantic + Syntactic)	74	1092	54.05%	8.11%	57.44%	58.93%	58.17%
Cross-validation Set(Syntactic Only)	74	1382	87.84%	8.11%	60.16%	57.99%	59.06%
Test Set(Semantic + Syntactic)	1059	12551	60.72%	5.19%	55.78%	57.76%	56.75%
Test Set(Syntactic Only)	1059	18722	85.74%	5.85%	60.21%	58.83%	59.51%

5.9 Comparison with Other WSJ Parsers

Parsing models that have used the Wall Street Journal Corpus have focused on two main tasks: full syntactic parsing [5, 7, 98, 100, 121], which the parsing model presented in this work does, and Semantic Role Labelling [147, 149, 150, 151, 152]. Reported work on full syntactic parsing has mostly involved traditional statistical parsing models which continue to represent the state-of-the-art for broad coverage natural language parsing (table 5.15). While this connectionist parsing

model does not yet compare favourably with the statistical parsers in terms of performance, it has achieved its performance with a far smaller training data set size. Its training to test data set ratio is 1:5.14.

Table 5.16: Comparison of Syntactic Parser Results on the WSJ Corpus

Full Syntactic Parser	Training Data Set	Test Data Set	Training To Test Data Ratio	Precision	Recall	F-Measure
This Parser (Syntactic only input)	202	1059 (74)	1:5.14 (2.73:1)	60.21 (60.16)	58.83 (57.99)	59.51 (59.06)
This Parser (Syntactic + Lexical semantics input)	206	1059 (74)	17.74:1 (2.78:1)	55.78 (57.44)	57.76 (58.93)	56.75 (58.17)
Charniak & Johnson (2005)[121]	39832	2245	17.74:1	91.3	90.6	90.9
Bod (2003)[7]	39832	2245	17.74:1	90.8	90.7	90.7
Charniak (2000)[5]	39832	2245	17.74:1	89.5	89.6	89.5
Collins (2000)[98]	39832	2245	17.74:1	89.9	89.6	89.7
Ratnaparkhi (1997)[100]	39832	2245	17.74:1	87.5	86.3	86.9

5.10 Discussion

To enable the combination of lexical semantic information with syntactic knowledge as input to the parser, an algorithm, involving a semi-automatic semantic tagging procedure, has been developed for the semantic annotation of nouns in the BLLIP WSJ corpus. A manual parse of semantically tagged sentences from the WSJ corpus shows that the noun classes obtained from WordNet provide sufficient information to aid the disambiguation of preposition attachment cases. They have also been found to be sufficient in the preposition attachment resolution for the following sentence pairs (POS tags for the first three pairs are the same; including the lexical semantic information provides useful additional knowledge):

1a) The boy ate the pasta with the sauce.

1b) The boy ate the pasta with he fork.

2a) The boy broke the window with the curtain.

2b) The boy broke the window with the rock.

3a) The policeman chased the boy with a limp.

3b) The policeman chased the boy with a truncheon.

4a) I examined the man with a stethoscope.

4b) I examined the man with a broken leg.

For all three connectionist modules of this parser, the networks trained on the combination of lexical semantic and syntactic input representation (compared to the networks trained on only syntactic input representations) dealt with more complex tasks (using a greater number of network connections) and had to be trained for longer periods in terms of number of epochs and actual training time. The delimiter networks learnt slightly lower proportions of the sequences presented to them when trained on a combination of lexical semantic and syntactic input representation, with both the LRD and RLD networks learning over 94.5% of

sequences in both input representation cases. However, a comparison of the generalisation performances of the delimiter networks, using two test sets, reveals very close performances when both sets of input representations are used. The generalisation performance of each delimiter network when lexical semantic information is added appears to be at par with its performance when only syntactic information is used.

In comparing the performances of the delimiters at sequence length level, there was a similar trend in learning performance for sequences of the same length, apart from one case. This is the case with LRD sequences of length 7, there is 9.69% difference in training performance between both input representation instances; the LRD network with a combination of lexical semantic and syntactic input representation learns only 86.99% of these sequences. Given that LRD sequences make use of three look-back and three look-ahead symbols, sequences of length 7 would normally be phrases with only one word or constituent, for example a single noun forming a noun phrase. The introduction of additional information seems to have made the parsing of phrases like this a more difficult task. This could be solved by fitting the network better to its training examples (longer training times and more optimal networks).

For the RLD, sequences with sequence lengths of 10, 11, 12 and 13 did not perform as well as sequences of other lengths (8, 9, 14, 15, 16 and 17). Sequences with these four least performing sequence lengths are the four most frequent sequences, in terms of sequence length. The RLD sequences include six look-back symbols. They also always include the end-of-sentence symbol, '*' and possibly symbols that do not belong to the phrase for which they set out to find the beginning of. The phrases involved with the sequence lengths of 10, 11, 12 and 13 would therefore be short phrases with the same phrase composition variety and

balancing issues as raised for the LRD. It is worth noting that for the RLD, sequences with the shortest sequence lengths (8 and 9) are among the high performers. Considering the presence of number of look-back symbols (6) and the end-of-sentence symbol, '*', phrases involved here would usually have one or two symbols; there are not very many of these phrases.

Another similarity in results obtained for the two different set of input representations is the consistency in the generalisation result on test sets generated from sentences of very different sample sizes. Generalisation performances were about the same for test data generated from 74 sentences and for those generated from 1059 sentences. This shows that the generalisation performance in these cases is not deeply affected by test sample size.

On the whole, the module level performances of the delimiter networks seemed to be at par, irrespective of the set of input representation used. The module level performance of the phrase recognition network that used only syntactic input representation appeared to perform better than when a combination of lexical semantic and syntactic information was used. This also seemed the case with performances at the sentence level. However, when using a combination of lexical semantic and syntactic information, the sentence level results have kept improving with positive modifications to the component connectionist modules. Its F-Measure has improved from 69.26% to 73.45% and up to 73.60% with training improvement to different component connectionist modules. This indicates a lot of room for improvement of the sentence level performance if the connectionist modules are further optimised. To fully grasp the effect of combining lexical semantic and syntactic input representation on the parser, an examination of the parser's behaviour on the two sets of input representations is necessary. This is done in the next chapter.

6. SENTENCE LEVEL EVALUATION

6.1 Introduction

Having been adapted to syntactically analyse sentences from the Wall Street Journal Corpus, and further adapted to integrate lexical semantic and syntactic representations in this analysis, the sentence level results achieved by the parser are presented in chapters 4 and 5. There is the need to analyse these results in detail in order to get an insight into the parser's behavioural characteristics and structural preferences. This analysis should also reveal the effect of integrating lexical semantic and syntactic representations on the capabilities of the parser.

In section 6.2, the performance of the parser on the Wall Street Journal Corpus, given only syntactic information as input, is examined in detail. A similar analysis is done on the parser, given a combination of lexical semantic and syntactic information as input, in section 6.3. Section 6.3 also contains a comparison of the parser's behaviour and structural preferences when presented with the two sets of input representations. A summary of the findings in the analyses carried out in this chapter is presented in section 6.4.

6.2 Parser with Syntactic-Only Input Representation

As shown in table 5.14 in the last chapter, of the 202 training sentences syntactically analysed by the parser (using only syntactic information in its input representation), 178 (88.12%) were successfully parsed, with an F-measure of 75.76%. 41 (20.30%) of the parse trees produced by the parser from analysing the 202 training sentences were exact matches of the target parse trees. The parser failed to produce complete parse trees for 24 (11.88%) of these sentences.

For each sentence analysed by the parser, the parser could fail to produce a complete parse if, during the process of building up the parse tree for the sentence, any of its connectionist modules (RLD, LRD, and PSR networks) failed to carry out its function. In other words, if the right-to-left delimiter network failed to identify the beginning of a phrase, having seen all the word or constituent tags in the sentence, then the parser would fail for that particular sentence. This is notwithstanding how far the parser had gone in the parsing process for that sentence. Similarly, if the left-to-right delimiter network failed to identify the end of a phrase, having seen all the word or constituent tags to the right of the word/constituent tag identified as the beginning of phrase by the right-to-left delimiter network, the parser would fail for that particular sentence. Also, after the delimiter networks have identified a sequence of tags that constitute a phrase, if the phrase structure recogniser network fails to find a parent for that sequence, the parser would fail for that particular sentence. For the 24 failed parses, 1 (4.17%) failed parse was due to the right-to-left delimiter network's inability to find the beginning of a phrase. 5 (20.83%) failed parses were due to the left-to-right delimiter network's inability to find the ends of certain phrases. 18 (75%) failed parses were due to the phrase recogniser network's inability to create the valid phrase classification from the sequence passed to it by the delimiter networks.

Despite 75% of failed parses being due to the inability of the phrase recogniser network to create a valid phrase from the sequences passed to it by the delimiter networks, most of the failed parses can be attributed to incorrect phrase boundary identifications by the delimiter networks. This is because if the beginning and/or end of a phrase are incorrectly indicated by any or both of the delimiter networks, the phrase recogniser network receives an incorrect sequence of tokens to create a valid phrase for. In most cases, the PSR network creates a valid parent for such sequences; however, the error only accumulates in the shift-reduce process. In the cases where the PSR network failed to create valid phrases from the sequences

passed to it, all of these sequences have been identified to be single constituent tags (14 'NP's, 2 'S1's, 1 'VP and 1'FRAG'). The phrase structure recogniser network is designed to result in a failed parse if, at any point in the parsing process, the 'daughter' (sequence of word/constituent tags input to the PSR network to be classified as a phrase) is the same as the 'mother' (constituent tag to be identified by the PSR network as denoting the phrase for the input sequence of tags). This design is to prevent the parser from shifting and reducing indefinitely at a particular stage in the shift-reduce parsing process where the 'daughter' and 'mother' are the same.

The 24 sentences which the parser failed to completely parse had RLD and LRD data generated from them. These data were then used to test the different delimiter modules to determine how much of this group of sentences they had learnt during training. Results from these tests, shown in table 6.1 (detailed sequence level results are shown in tables 6.4 and 6.7) indicate a strong learning performance at module level. These modular results for the failed sentences compare favourably with those for the mismatched sentences, as shown in table 6.3 (detailed sequence results for the mismatched sentences are shown in tables 6.6 and 6.9). Modular results (shown in tables 6.2, 6.5 and 6.8) for the sentences whose parses matched the target parses indicate, expectedly, that all, but 9 RLD sequences were learnt during training.

The strong modular performances of the failed/mismatched sentences, compared to their sentence level performance highlights a limitation of the modular model that is due to the knock-on effects that occur throughout the shift-reduce parsing process. During the parsing process, the parser's connectionist modules operate in cascade. The RLD first processes the sequences and passes its results, including any errors, if available, to the LRD. The LRD passes its own results, including any errors, if available to the PSR.

Table 6.1: Training Results for the LRD and RLD Data Generated from Failed Sentences

	Hidden Nodes	Connections	No. of Patterns	No. of Sequences	RMS Error	% Pat. Learnt	% Seq. Learnt
RLD	165	70,172	6535	580	0.0489381	90.60	94.31
LRD	105	32,072	4691	580	0.0648378	86.72	92.59

Table 6.2: Training Results for the LRD and RLD Data Generated from Matching Sentences

	Hidden Nodes	Connections	No. of Patterns	No. of Sequences	RMS Error	% Pat. Learnt	% Seq. Learnt
RLD	165	70,172	4507	420	0.027647	90.66	97.76
LRD	105	32,072	3370	420	0.0440867	87.54	100

Table 6.3: Training Results for the LRD and RLD Data Generated from Mismatching Sentences

	Hidden Nodes	Connections	No. of Patterns	No. of Sequences	RMS Error	% Pat. Learnt	% Seq. Learnt
RLD	165	70,172	30,496	2,747	0.0486726	90.45	94.03
LRD	105	32,072	22,193	2,747	0.0577466	87.03	95.30

Table 6.4: Training Results (for sequences of different lengths) for the LRD Data Generated from Failed Sentences

Sequence Length	7	8	9	10	11	12	13
No. of Sequences	148	297	92	28	11	2	2
% Sequences Learnt	95.95	89.56	93.48	100	100	100	100

Table 6.5: Training Results (for sequences of different lengths) for the LRD Data Generated from Matching Sentences

Sequence Length	7	8	9	10	11	12	13
No. of Sequences	148	171	66	22	8	1	4
% Sequences Learnt	100	100	100	100	100	100	100

Table 6.6: Training Results (for sequences of different lengths) for the LRD Data Generated from Mismatching Sentences

Sequence Length	7	8	9	10	11	12	13
No. of Sequences	732	1394	418	122	49	20	12
% Sequences Learnt	98.09	92.97	96.41	98.36	100	100	100

Table 6.7: Trained Results (for sequences of different lengths) for the RLD Data Generated from Failed Sentences

Length of Sequence	8	9	10	11	12	13	14	15	16	17
No. of Sequences	48	23	138	136	93	76	38	17	10	1
% Sequences Learnt	100	100	92.75	89.71	95.70	93.42	100	100	100	100

Table 6.8: Training Results (for sequences of different lengths) for the RLD data Generated from Matching Sentences

Length of Sequence	8	9	10	11	12	13	14	15	16	17
No. of Sequences	87	13	125	60	44	42	30	15	3	1
% Sequences Learnt	100	100	100	100	97.73	100	100	100	100	100

Table 6.9: Training Results (for sequences of different lengths) for the RLD data Generated from Mismatching Sentences

Length of Sequence	8	9	10	11	12	13	14	15	16	17
No. of Sequences	274	51	939	510	360	255	223	86	35	14
% Sequences Learnt	100	100	92.11	90.39	94.72	92.94	98.21	100	100	100

In the course of the training session, the parser would have acquired linguistic knowledge from the training data presented to it. This linguistic knowledge is stored in the synaptic weights for the different connectionist modules of the parser. In order for this parser to be useful in any automatic natural language application, it should be able to syntactically analyse sentences not presented to it during training, using its acquired linguistic knowledge. Two sets of test sentences were presented to the parser; one consisting of 74 sentences, the other comprising 1059 sentences.

As shown in table 5.14 in the last chapter, of the 74 test sentences (from the first test set) syntactically analysed by the parser (using only syntactic information in its input representation), 65 (87.84%) were successfully parsed, with an F-measure of 59.06%. 6 (8.11%) of the parse trees produced by the parser from analysing the 74 test sentences were exact matches of the target parse trees. The parser failed to produce complete parse trees for 9 (12.16%) of these sentences.

As also shown in table 5.14 in the last chapter, of the 1059 test sentences (from the second test set whose sentence composition is different from that of the first test set) syntactically analysed by the parser (using only syntactic information in its input representation), 908 (85.74%) were successfully parsed, with an F-measure of 59.51%. 62 (5.85%) of the parse trees produced by the parser from analysing the 1059 test sentences were exact matches of the target parse trees. The parser failed to produce complete parse trees for 151 (14.26%) of these sentences.

The parser's performance on these test sets shows its ability to generalise to sentences not seen during training. The parser has been able to do this because of linguistic knowledge derived during training. The parser's performance was also

consistent, irrespective of the size of the test set. In most cases, where the parser could not achieve the target parse for a test sentence, it was able to make useful approximations. However, the inadequacies noticed in the parser during training, also affect its generalisation performance. Its performance is also hindered in part by inconsistencies in the pre-parsed corpus.

In examining the behaviour of the parser, a sample of 12 matching parse trees produced by the parser from sentences belonging to the large test set (1059 sentences) has been extracted and presented below. The sentences whose parse trees are shown below are of varying structural complexity. Figures 6.1, 6.2 and 6.3 display the parser's ability to generalise to sentences with simple syntactic structures. The parser is able to analyse the determiner and two nouns that constitute the object noun phrase in figure 6.1. It is also able to deal with the modifying adjectives in the subject noun phrase as well as the case that there is no object noun phrase in figure 6.2. In figure 6.3, the parser is shown to be able to handle the cardinal number and noun that constitute the object noun phrase.

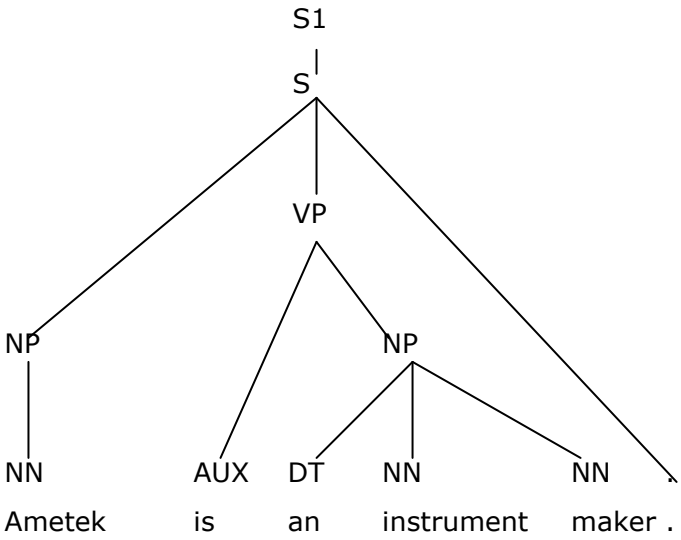

Figure 6.1: Matching parse tree for the sentence, *Amatek is an instrument maker.* (Using only syntactic information)

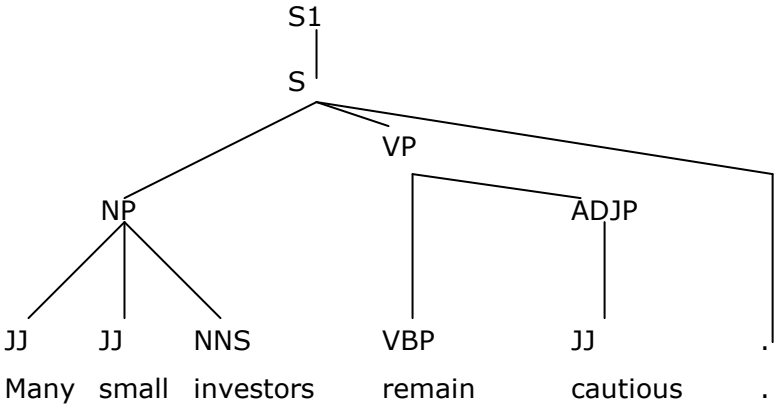

Figure 6.2: Matching parse tree for the sentence, *Many small investors remain cautious.* (Using only syntactic information)

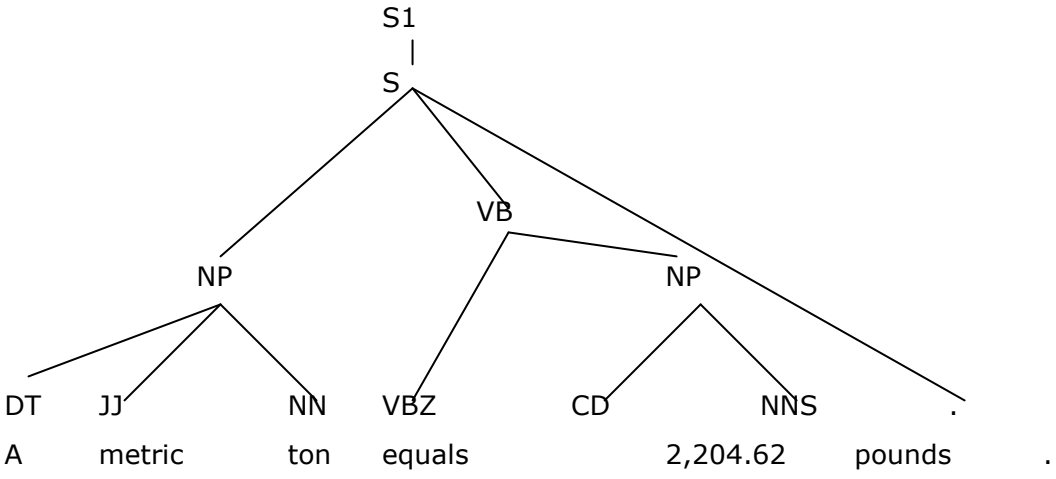

Figure 6.3: Matching parse tree for the sentence, *A metric ton equals 2,204.62 pounds.* (Using only syntactic information)

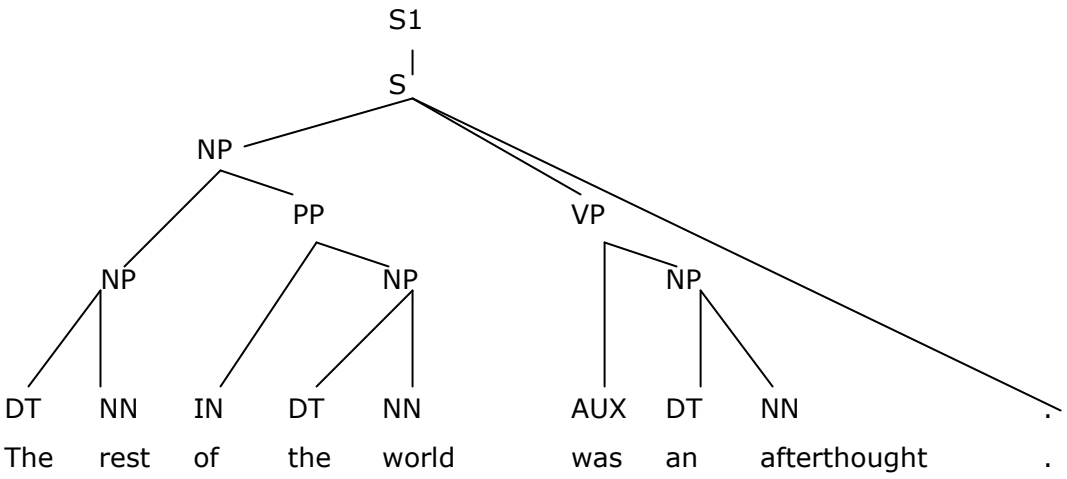

Figure 6.4: Matching parse tree for the sentence, *The rest of the world was an afterthought.* (Using only syntactic information)

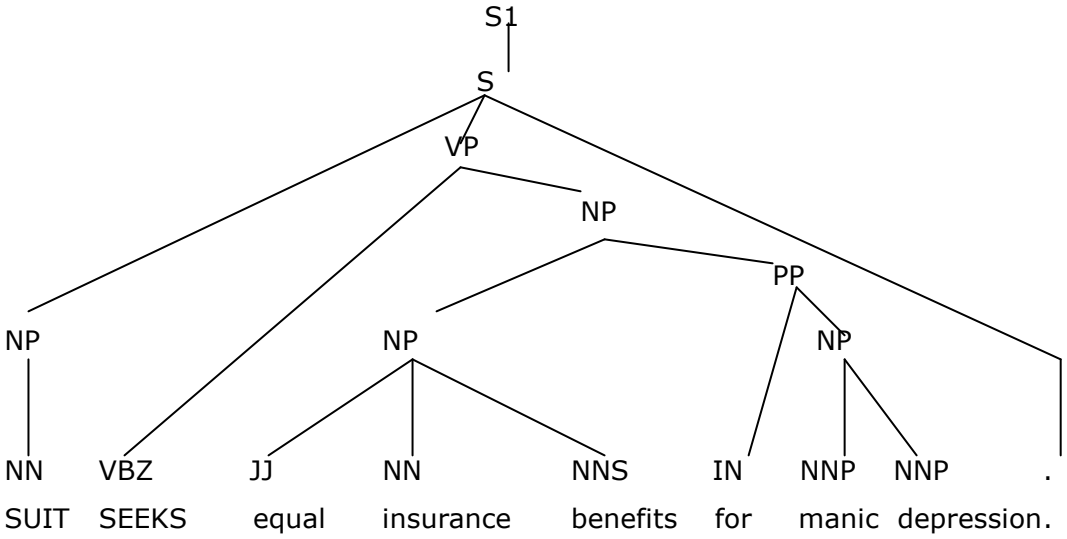

Figure 6.5: Matching parse tree for the sentence, *SUIT SEEKS equal insurance benefits for manic depression.* (Using only syntactic information)

Figures 6.4 and 6.5 show the parser's ability to analyse sentences containing more complex noun phrases, while dealing with cases requiring the attachment of preposition phrases. The parser is also seen to be able to handle recursivity in sentence structure. In figures 6.6 and 6.7 the parser is shown to be able to parse sentence structures containing right-embedded clauses. The parser is also able to

handle centre-embedding as seen in figures 6.8 and 6.9. Figures 6.10, 6.11 and 6.12 show the parser's handling of more structurally complex sentences.

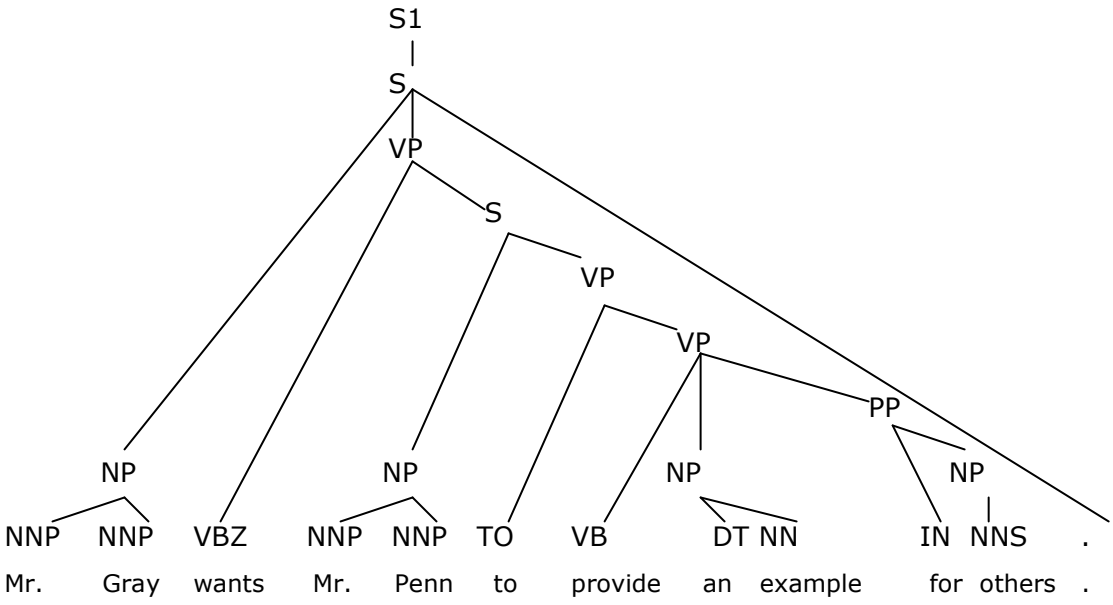

Figure 6.6: Matching parse tree for the sentence, *Mr. Gray wants Mr. Penn to provide an example for others.* (Using only syntactic information)

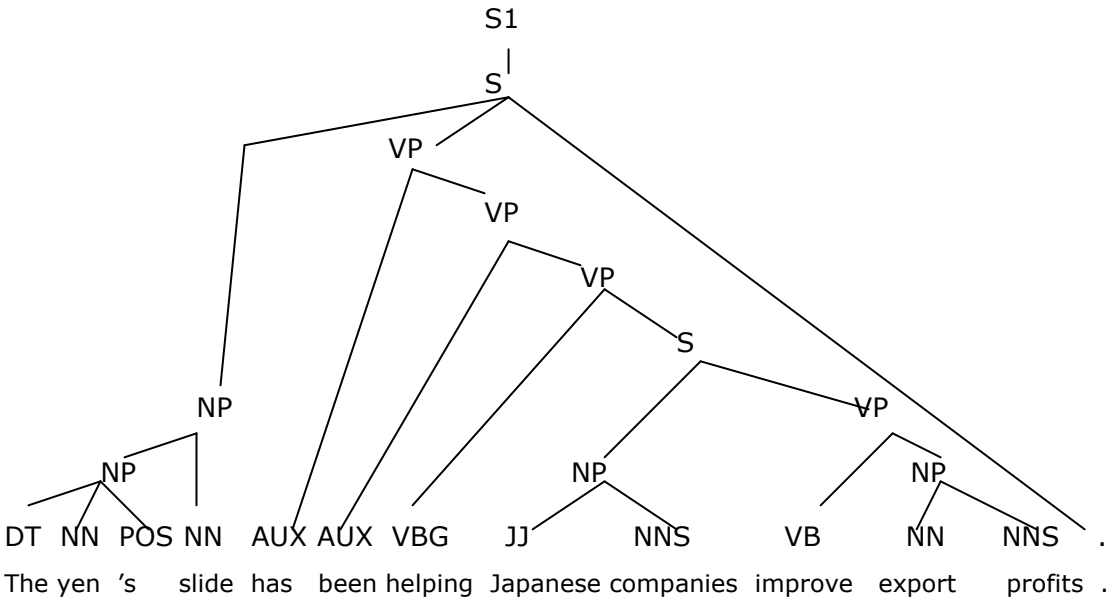

Figure 6.7: Matching parse tree for the sentence, *The yen's slide has been helping Japanese companies improve export profits.* (Using only syntactic information)

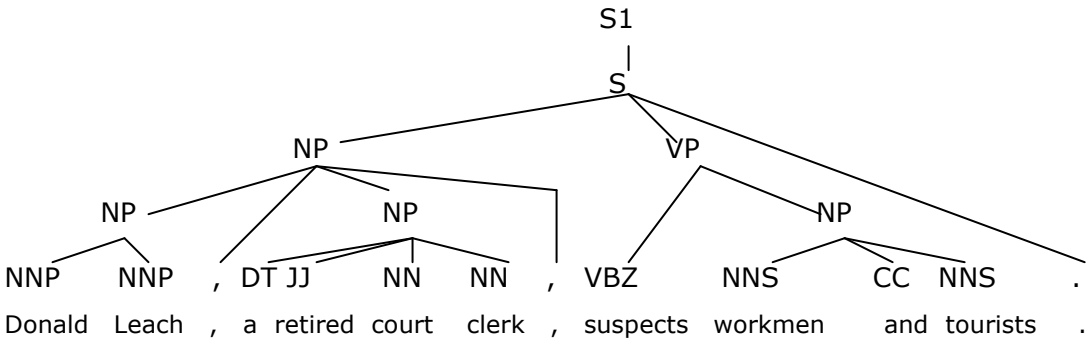

Figure 6.8: Matching parse tree for the sentence, *Donald Leach, a retired court clerk, suspects workmen and tourists.* (Using only syntactic information)

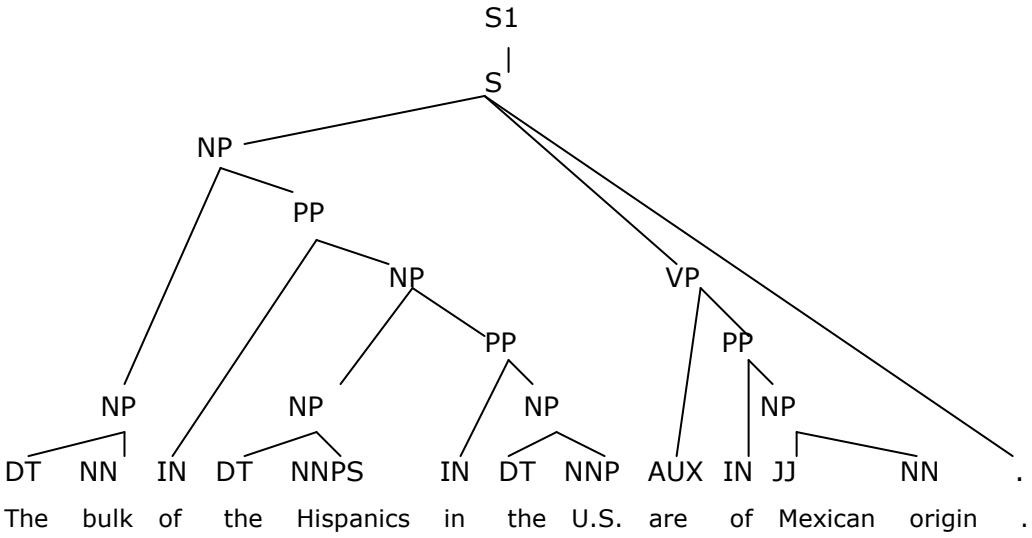

Figure 6.9: Matching parse tree for the sentence, *The bulk of the Hispanics in the U.S. are of Mexican origin.* (Using only syntactic information)

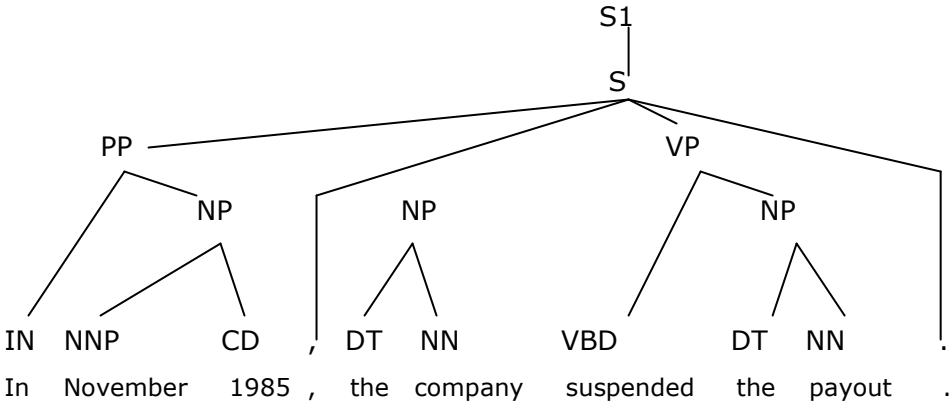

Figure 6.10: Matching parse tree for the sentence, *In November 1985, the company suspended the payout.* (Using only syntactic information)

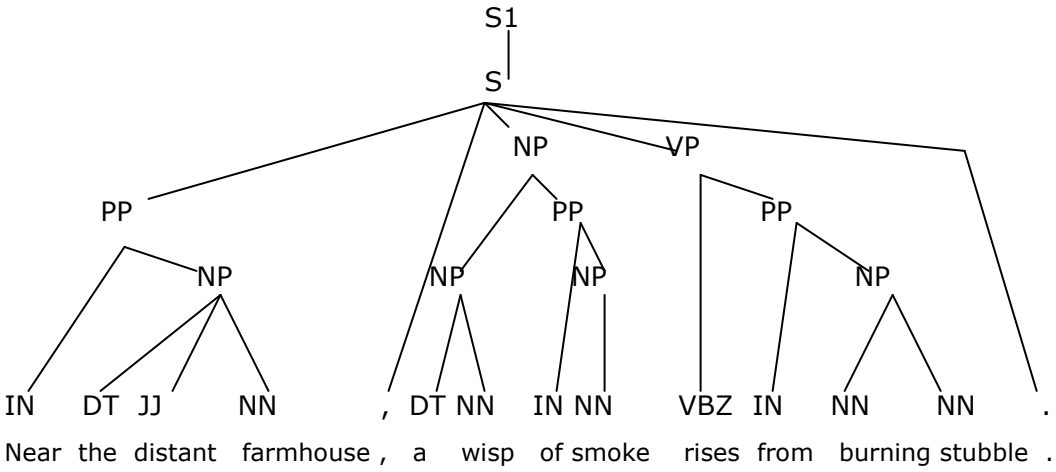

Figure 6.11: Matching parse tree for the sentence, *Near the distant farmhouse, a wisp of smoke rises from burning stubble.* (Using only syntactic information)

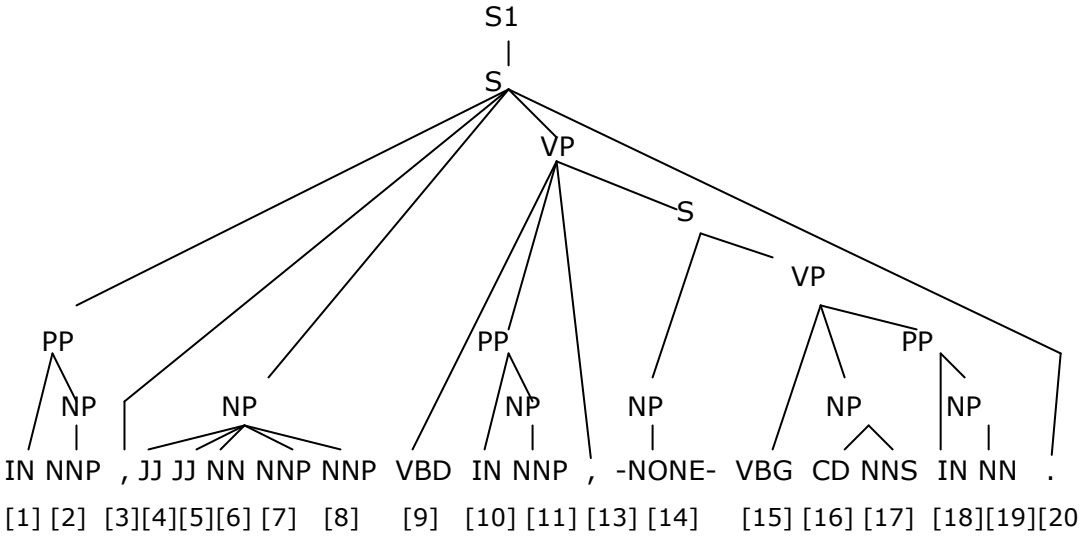

Figure 6.12: Matching parse tree for the sentence, ***In[1] Namibia[2],[3] a[4] black[5] nationalist[6] leader[7] Sam[8] Nujoma[9] arrived[10] in[11] Windhoek[12],[13] *-7[14] ending[15] three[16] decades[17] in[18] exile[19].[20]***

6.3 Parser with a Combination of Lexical Semantic and Syntactic Input Representation

As shown in table 5.14 in the last chapter, of the 206 training sentences syntactically analysed by the parser (using a combination of lexical semantic and syntactic information in its input representation), 141 (68.45%%) were successfully parsed, with an F-Measure of 73.60%. 36 (17.48%) of the parse trees produced by the parser from analysing the 206 training sentences were exact matches of the target parse trees. The parser failed to produce complete parse trees for 65 (31.55%) of these sentences.

For the 65 failed parses, 4 (6.15%) failed parses were triggered off by the right-to-left delimiter network's inability to find the beginnings of some phrases. 20 (30.77%) failed parses were triggered off by the left-to-right delimiter network's inability to find the ends of certain phrases. 41 (63.08%) failed parses were

triggered off by the phrase recogniser network's inability to create a valid phrase from the sequence passed to it by the delimiter networks.

Despite 63.08% of failed parses being triggered off by the inability of the phrase recogniser network to create a valid phrase from the sequences passed to it by the delimiter networks, most of the failed parses can be attributed to incorrect phrase boundary identifications by the delimiter networks. This is a similar situation to that when the parser had only syntactic information in its input representation. In the cases where the PSR network failed to create valid phrases from the sequences passed to it, all of these sequences have been identified to be single constituent tags (30 'NP's, 1 'S1', 2 'VP's, 1 'S', 1 'ADVP', 2 'PRN's, 1 'INTJ', 1 'WHAVDP', 1 'QP' and 1 'UCP'). This is deduced from the fact that the parser's average precision and recall (where phrase 'daughters' are correctly identified but the 'mother' is not) on the training set compares favourably with its average labelled precision and recall (average precision = 73.76%; average recall = 74.85%; average labelled precision =73.06%; average labelled recall = 74.14%).

The parser's "knock-on effect" limitation (due to its connectionist modules operating in cascade), seen when it had only syntactic information in its input representation, still affects its performance with the combined input representation. The number of sentences successfully parsed and F-measure were lower than when the parser had only syntactic information in its input. It is envisaged that further optimising the network sizes for the delimiter modules would optimise this shortfall; although the number of hidden nodes had been increased to cope with the increased complexity of combining lexical semantic and syntactic information, there seems to be some room for improvement.

To test the modified parser's ability to syntactically analyse sentences not presented to it during training, two sets of test sentences were presented to it.

These are the same sets of test sentences earlier presented to the parser when it had only syntactic information in its input representation.

Of the 74 test sentences (from the first test set - as shown in table 5.14 in the last chapter -) syntactically analysed by the parser (using a combination of lexical semantic and syntactic information in its input representation), 40 (54.05%) were successfully parsed, with an F-measure of 58.17%. 6 (8.11%) of the parse trees produced by the parser from analysing the 74 test sentences were exact matches of the target parse trees. The parser failed to produce complete parse trees for 34 (45.95%) of these sentences.

Of the 1059 test sentences (from the second test set whose sentence composition is different from that of the first test set - as also shown in table 5.14 in the last chapter -) syntactically analysed by the parser (using a combination of lexical semantic and syntactic information in its input representation), 643 (60.72%) were successfully parsed, with an F-Measure of 56.75%. 55 (5.19%) of the parse trees produced by the parser from analysing the 1059 test sentences were exact matches of the target parse trees. The parser failed to produce complete parse trees for 416 (39.28%) of these sentences.

The parser's performance on the test set (compared to its performance using only syntactic information in its input representation) reflects its training results. Its generalisation performance on the test sentences was, however, consistent, irrespective of the size of the test set.

In order to observe the effects, if any, of combining lexical semantic and syntactic information in the parser's input representation, it is pertinent to compare the parser's analytic behaviour, given the two sets of input representations. All matching and mismatched parses were examined. A sample of eight pairs of parse

trees is presented here. Each pair comprises a matching (to the target Treebank parse tree) parse tree constructed by the parser using the combined linguistic information and a mismatching parse tree constructed, for the same sentence, by the parser with only syntactic information. Another sample comprising three mismatching parse trees constructed by the parser using the combined linguistic information is presented, in comparison with matching parse trees constructed by the parser for the same sentences using only syntactic information.

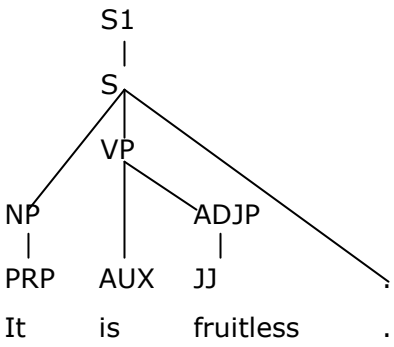

Figure 6.13: Matching Parse tree for the sentence, *It is fruitless.* (Using combined linguistic information)

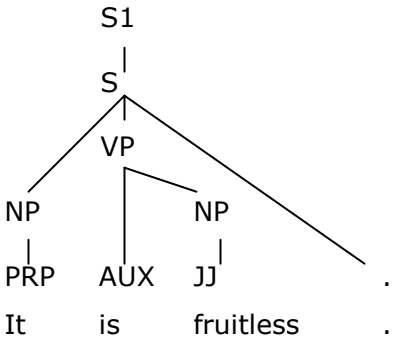

Figure 6.14: Mismatching parse tree for the sentence, *It is fruitless.* (Using only syntactic information)

Figures 6.13 and 6.14 show two parse trees for the sentence, *It is fruitless.*, where the parser used a combination of linguistic information to make better judgement on analysing the given structure. Using a combination of lexical semantic and

syntactic, compared to its use of only syntactic information in its input representation, the parser also made better parsing decisions in the structural analysis of the sentences shown in figures 6.15 to 6.28. Worthy of note is the parser's consistent better decision making (when using a combination of lexical semantic and syntactic information in its input representation) in the attachment of preposition phrases (figures 6.21 to 6.28).

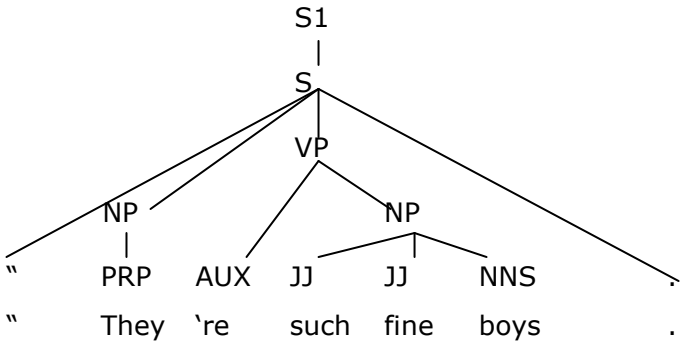

Figure 6.15: Matching Parse tree for the sentence, *"They're such fine boys.* (Using combined linguistic information)

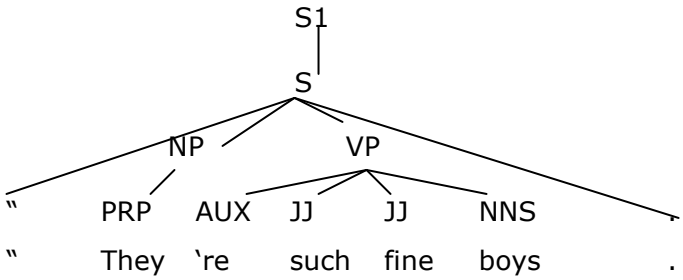

Figure 6.16: Mismatching parse tree for the sentence, "They're such fine boys. (Using only syntactic information)

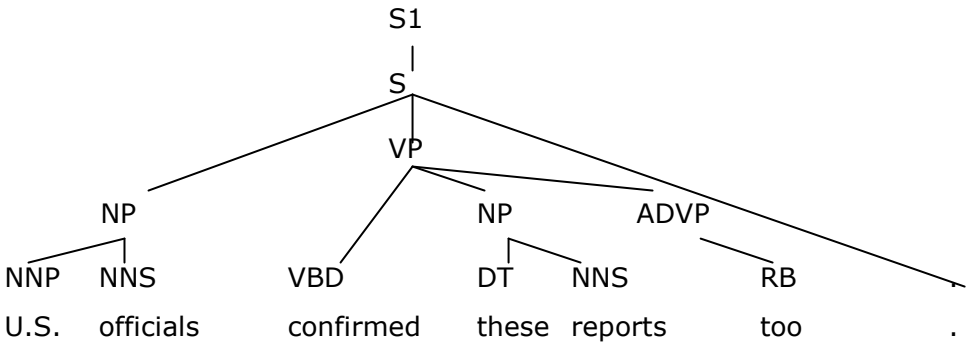

Figure 6.17: Matching Parse tree for the sentence, *U.S. officials confirmed these reports too.* (Using combined linguistic information)

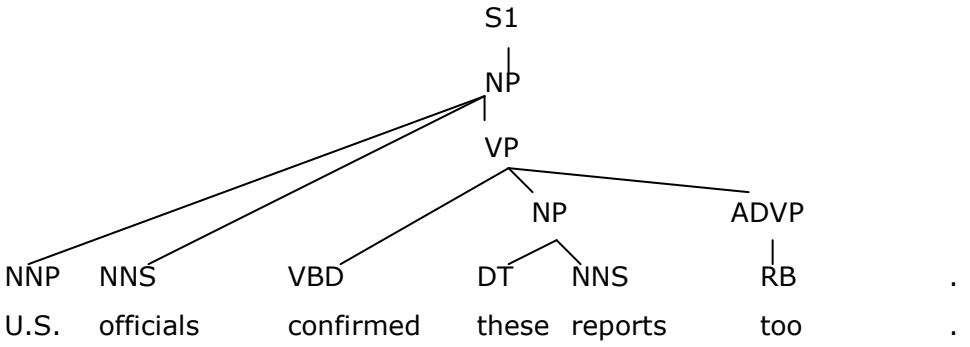

Figure 6.18: Mismatching parse tree for the sentence, U.S. officials confirmed these reports too. (Using only syntactic information)

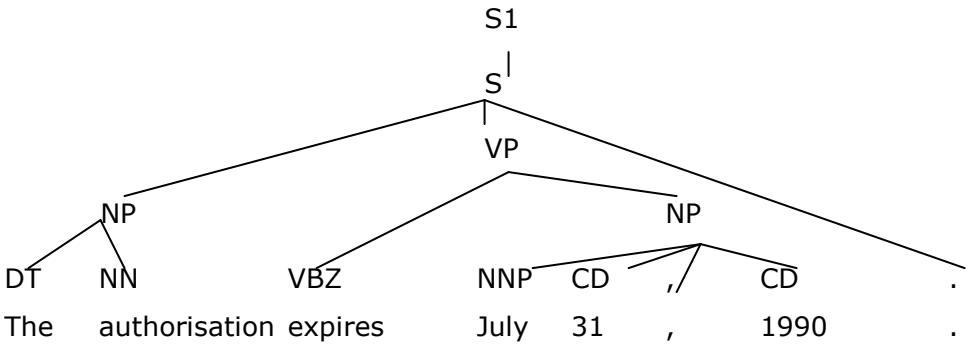

Figure 6.19: Matching Parse tree for the sentence, *The authorisation expires July 31, 1990.* (Using combined linguistic information)

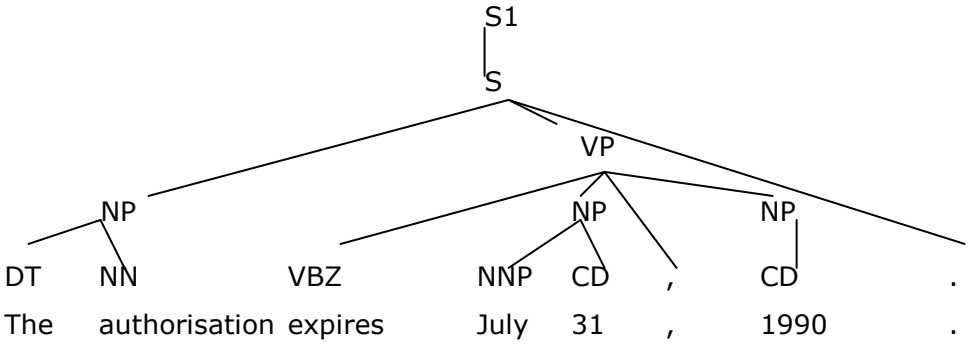

Figure 6.20: Mismatching parse tree for the sentence, *The authorisation expires July 31, 1990.* (Using only syntactic information)

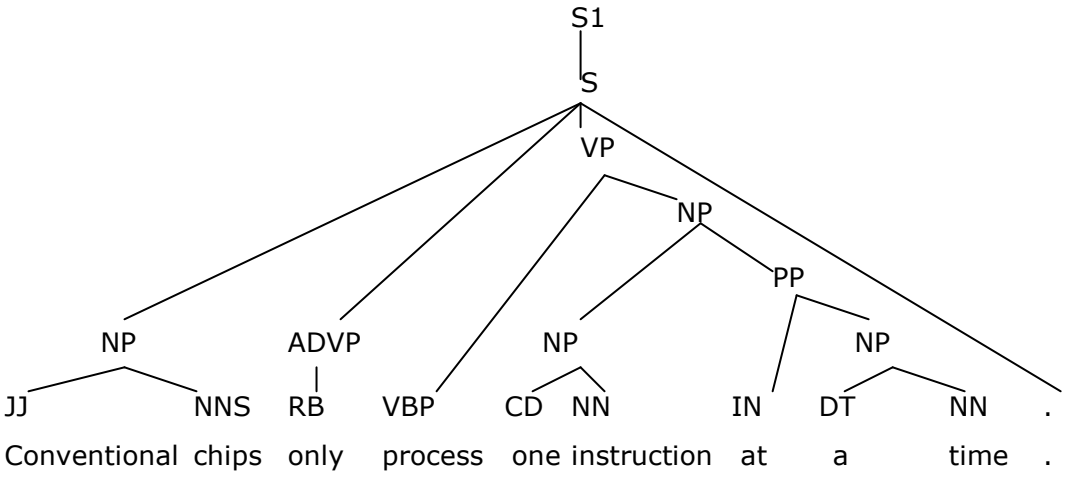

Figure 6.21: Matching Parse tree for the sentence, *Conventional chips only process one instruction at a time.* (Using combined linguistic information)

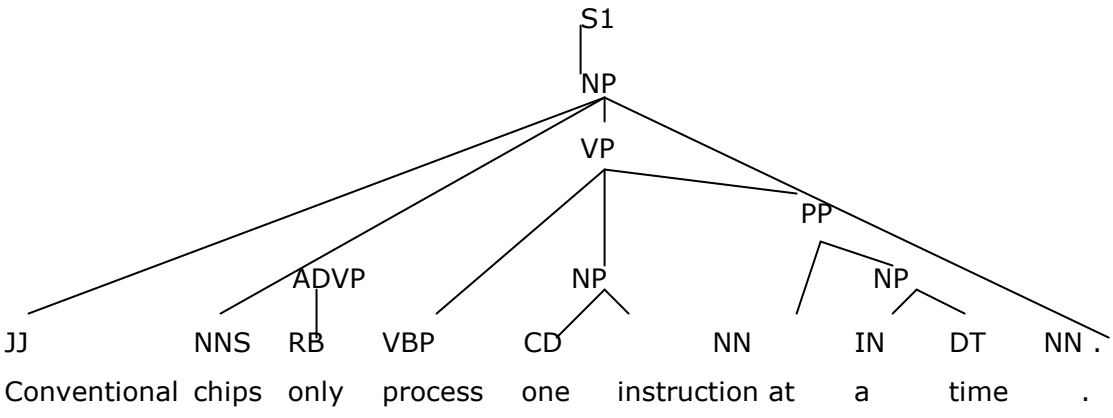

Figure 6.22: Mismatching parse tree for the sentence, *Conventional chips only process one instruction at a time.* (Using only syntactic information)

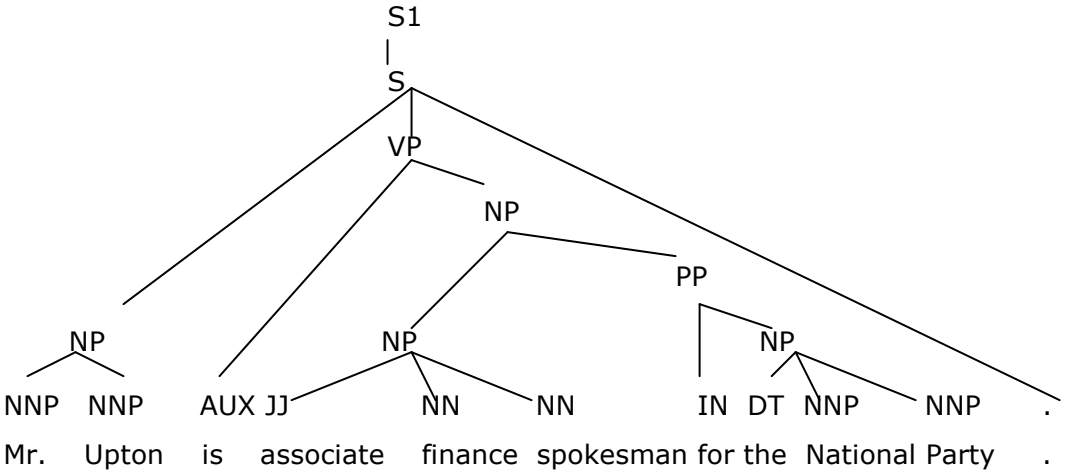

Figure 6.23: Matching Parse tree for the sentence, *Mr. Upton is associate finance spokesman for the National Party.* (Using combined linguistic information)

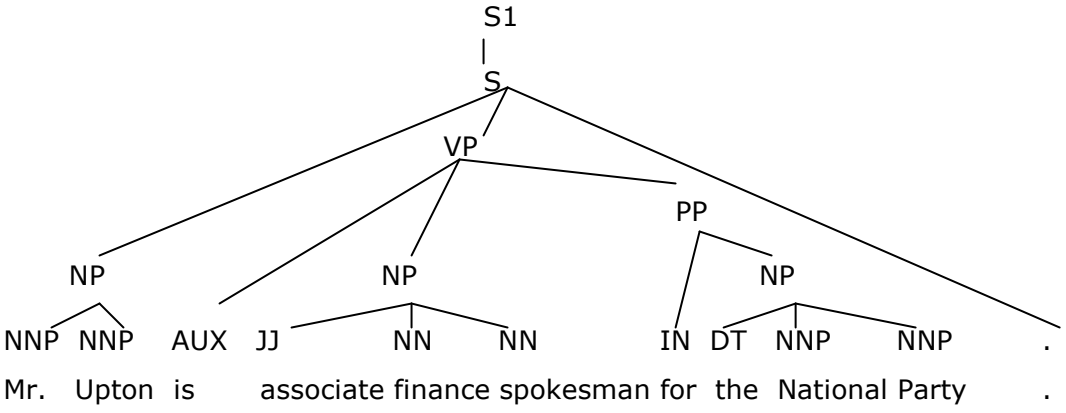

Figure 6.24: Mismatching parse tree for the sentence, *Mr. Upton is associate finance spokeman for the National Party.* (Using only syntactic information)

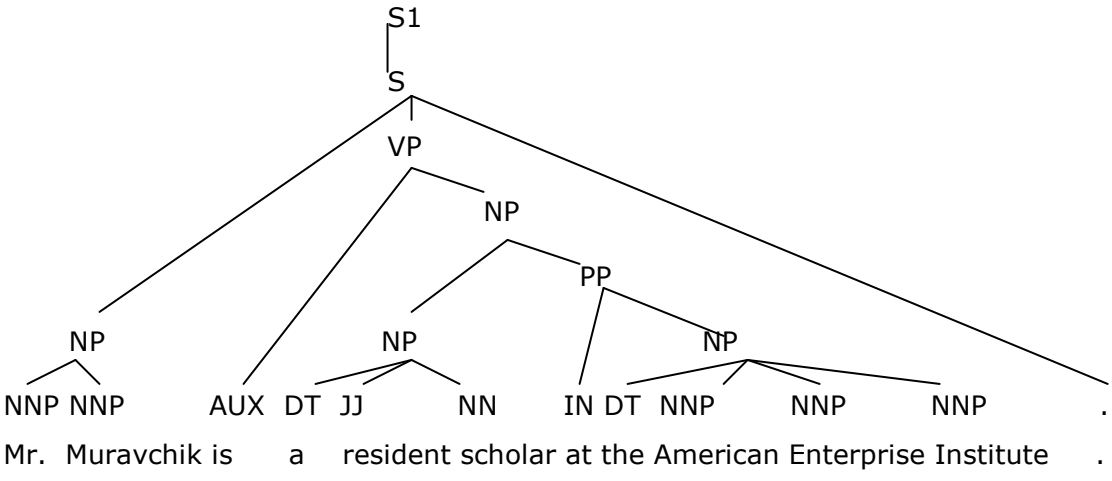

Figure 6.25: Matching Parse tree for the sentence, *Mr. Muravchik is a resident scholar at the American Enterprise Institute.* (Using combined linguistic information)

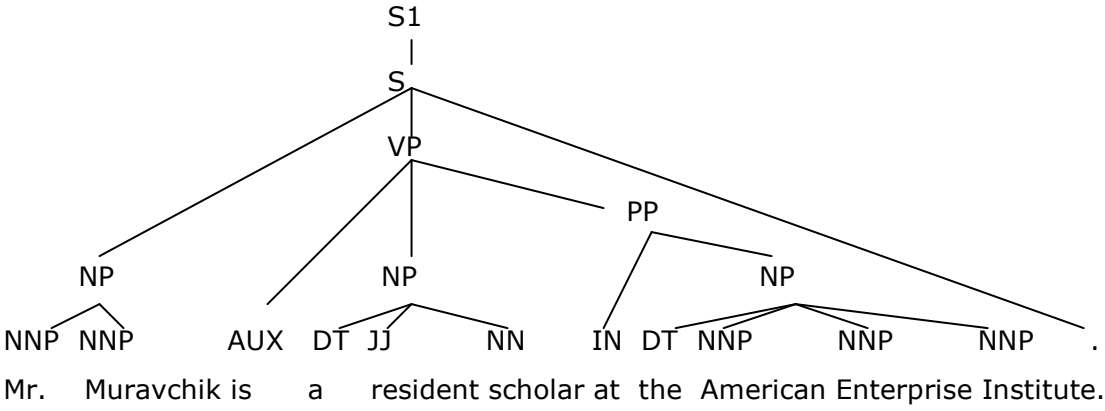

Figure 6.26: Mismatching parse tree for the sentence, *Mr. Muravchik is a resident scholar at the American Enterprise Institute.* (Using only syntactic information)

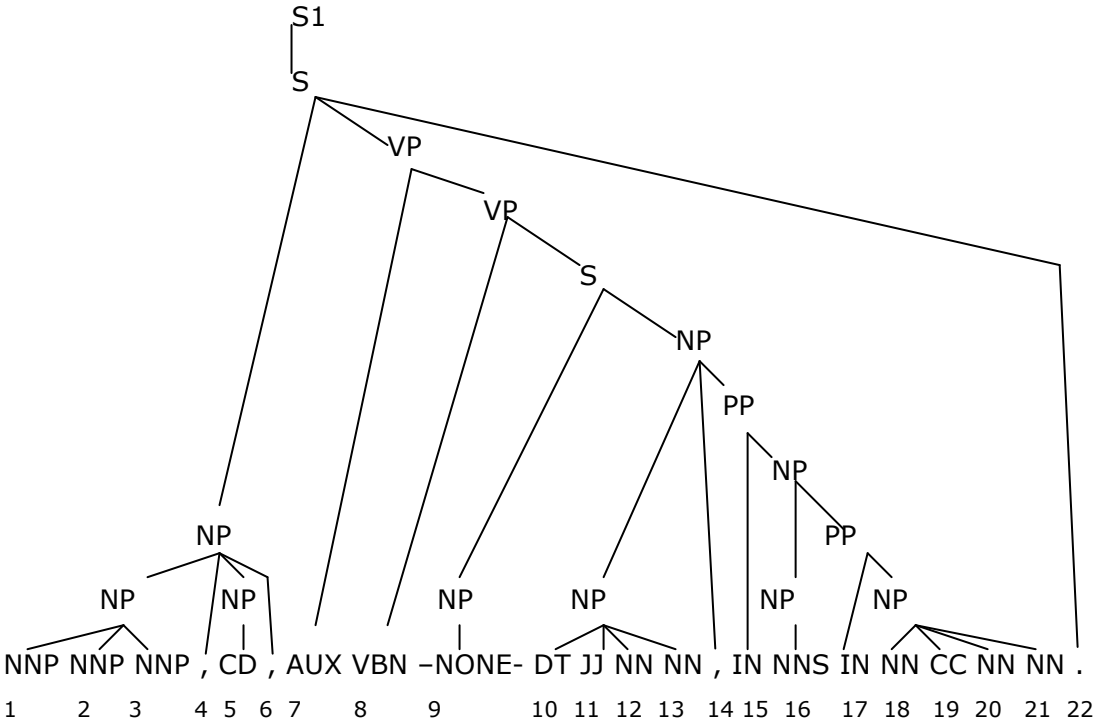

Figure 6.27: Matching Parse tree for the sentence, *Marshall[1] N.[2] Norton[3],[4]44[5],[6] was[7] elected[8] *-1[9] a[10] senior[11] vice[12] president[13],[14] with[15] responsibilities[16] in[17] finance[18] and[19] data[20] processing[21].[22]* (Using combined linguistic information)

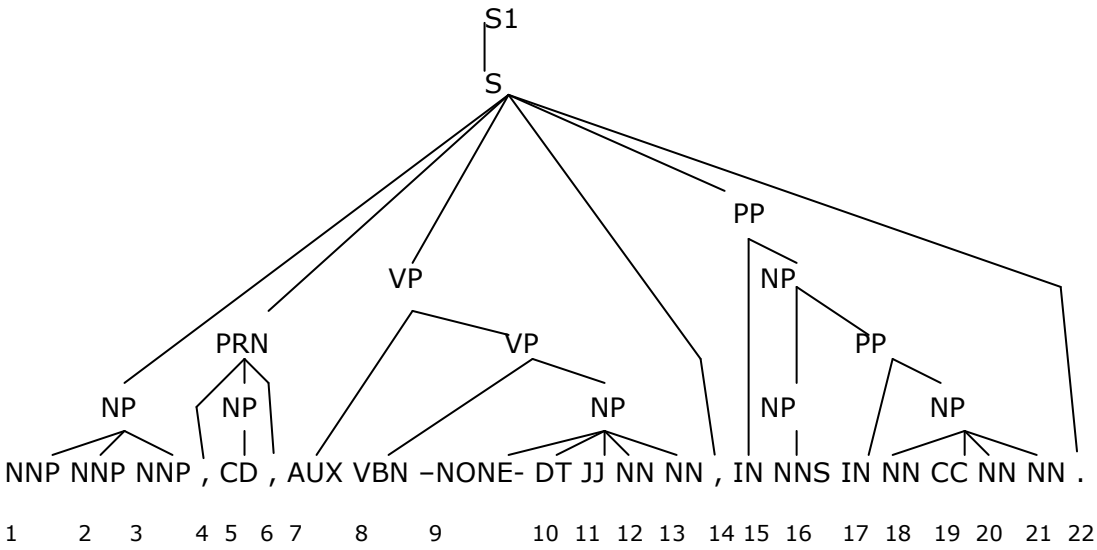

Figure 6.28: Mismatching parse tree for the sentence, *Marshall[1] N.[2] Norton[3],[4]44[5],[6] was[7] elected[8] *-1[9] a[10] senior[11] vice[12] president[13],[14] with[15] responsibilities[16] in[17] finance[18] and[19] data[20] processing[21].[22]* (Using only syntactic information)

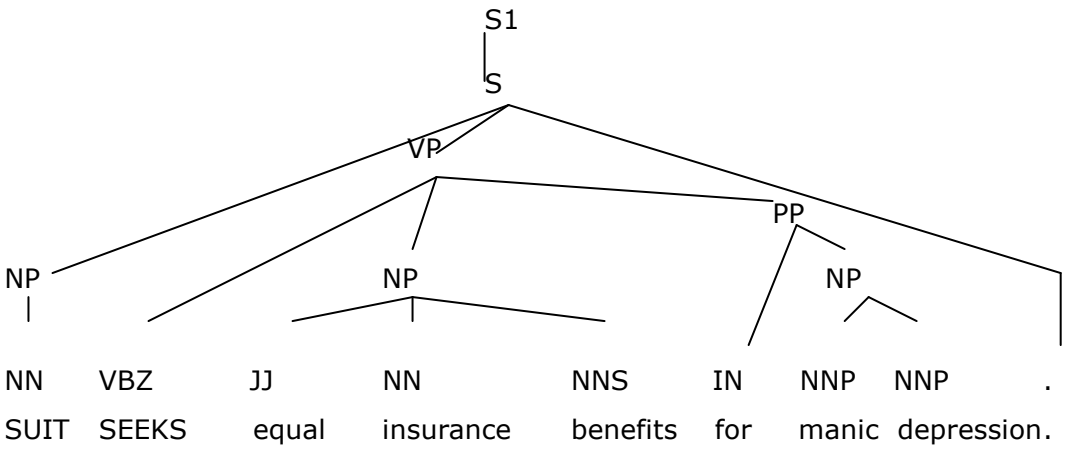

Figure 6.29: Mismatching parse tree for the sentence, *SUIT SEEKS equal insurance benefits for manic depression.*, using combined linguistic information (see fig. 6.5 for matching parse)

Mismatched parsed trees produced by the parser, using a combination of linguistic information in its output is also compared with matching parse trees produced by the parser with only syntactic information. Figure 6.29 shows the mismatched parse

(using a combination of lexical semantic and syntactic information) of the same sentence whose parse tree (using only syntactic information) is depicted in figure 6.5. Although this parse (figure 6.29), where the parser prefers the high attachment (with the VP) for the proposition phrase, does not match the target parse, it is a plausible sentence analysis. It also shows that the parser (using the combined linguistic information) can make decisions on attaching preposition phrases to verb phrases (high attachment), as well as noun phrases (low attachment).

Figure 6.30 shows the mismatched parse (using a combination of lexical semantic and syntactic information) of the same sentence whose parse tree (using only syntactic information) is depicted in figure 6.4. Here the parser (using the combined linguistic information) erroneously identifies *the world was an afterthought* as a reduced relative clause.

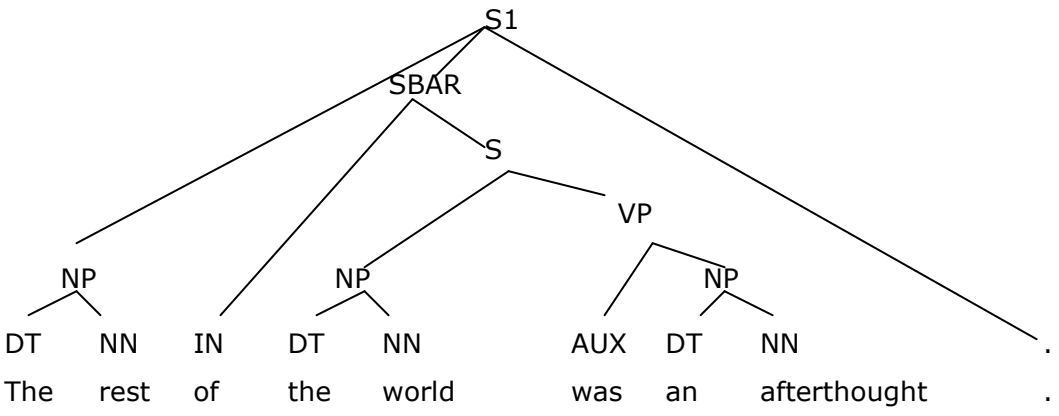

Figure 6.30: Mismatching parse tree for the sentence, *The rest of the world was an afterthought.*, using combined linguistic information (see fig. 6.4 for matching parse)

Of all the matched parses (labelled precision/recall = 100%/100%) achieved by the parser, 16 had preposition phrase attachment issues and were exclusively attained with only one set of input representation. When it used a combination of lexical semantic and syntactic information in its input representation, the parser successfully parsed 62.5% of these cases. On the other hand, when it used only syntactic information in its input representation, the parser was only able to parse

37.5% of these cases. The parser therefore appeared to be able to make better decisions concerning preposition phrase attachment when it used a combination of lexical semantic and syntactic information than when it used only syntactic information in its input representation.

6.4 Summary

The parser has successfully acquired a degree of linguistic knowledge inherent in the BLLIP WSJ Corpus by learning to syntactically analyse sentences from this corpus. Although there is still room for improvement on its learning and generalisation performance, the parser is able to generalise to sentences of the same structural complexity as the training sample. In generalising to test sets that were not used during training, the parser's generalisation performance has been consistent, irrespective of test sample size.

In exploiting the connectionist nature of the parser, and by extension, its ability to make use of multiple constraints during sentence processing, lexical semantic information was combined with syntactic information in the parser's input representation. In terms of the number of successfully parsed sentences, this combination of linguistic information did not yield better performance for the parser. It is envisaged that a further optimisation of the network size used in training with the combination of linguistic information could lead to better performance. However, an in-depth look at the parser's analysis revealed that the parser appeared to be able to make better decisions concerning preposition phrase attachment when it used a combination of lexical semantic and syntactic information than when it used only syntactic information in its input representation; of the 16 matched parses (involving preposition phrase attachment resolution) that were exclusively parsed using either set of input representation, the parser successfully parsed 62.5% of these cases when it used a combination of lexical semantic and syntactic information in its input representation.

7. CONCLUSIONS

7.1 Introduction

The research project presented in this dissertation, has focused on two main characteristics of connectionist models for natural language processing. These characteristics are their adaptability to different tagging conventions, and their ability to use multiple linguistic constraints in parallel during sentence processing. In focusing on these key characteristics, an existing parsing model has been modified. This model is a hybrid connectionist, shift-reduce corpus-based parser.

This parser, which had earlier been trained to acquire some level of linguistic knowledge from the Lancaster Parsed Corpus, has been adapted to learn a degree of linguistic knowledge from the BLLIP Wall Street Journal Corpus. This adaptation is a novel demonstration that this connectionist parser, and possibly, other similar connectionist models, is able to adapt to more than one tagging convention; this implies their ability to adapt to the underlying linguistic theories used to annotate different corpora.

Another characteristic of connectionist systems is their inherent ability to use multiple constraints in decision making. In further exploiting this aspect of the connectionist nature of this parsing model, it has been adapted to integrate shallow lexical semantic information with syntactic information for full syntactic parsing. This novel approach to the integration of lexical semantic and syntactic information was used to investigate the effect of shallow lexical semantic information on full syntactic parsing.

A challenge encountered in the attempt to integrate shallow lexical semantic information with syntactic information for full syntactic parsing is the scarcity of

large-scale, pre-parsed corpora with lexical semantic annotation, as well as part of speech annotation. This challenge was surmounted with the development of a novel algorithm for semantic tagging of nouns in the BLLIP Wall Street Journal Corpus. The lexical semantic information used in this semantic annotation algorithm was extracted from WordNet, an online lexical resource. WordNet provides a lexical inheritance system for nouns.

Using only syntactic information in making parsing decisions, this parsing model was tested on test sets of sentences that were not used during training. The parser generalised to parse these test sentences with an F-measure of 72.5% and 59.5% on sentences from the Lancaster Parsed Corpus and Wall Street Journal Corpus, respectively. On the integration of shallow lexical semantic information with syntactic information in its input representation, the parser generalised to parse test sentences from the Wall Street Journal Corpus with an F-measure of 56.75%. Although the integration of shallow lexical semantic information with syntactic information has not seemed to improve the parser's overall training/generalisation performance yet, given its present configuration, it did appear to improve the parser's decision making in preposition phrase attachment cases.

The demonstrations and findings from investigations conducted in the course of this work contribute to the field of Connectionist Parsing in particular and the field of Artificial Intelligence in general.

Section 7.2 presents details of specific contributions made to the field of Connectionist Parsing by this work. In section 7.3, further lines of investigation and improvements to the parsing model are suggested as future work to the research.

7.2 Contributions

7.2.1 Adaptation of Parsing Model to the BLLIP WSJ Corpus

Most previous connectionist parsers have been trained on hand-made or artificial grammars. Most of the few that have ventured into analysing natural language are yet to be trained and tested on a large scale, using broad-coverage corpora such as the Wall Street Journal Corpus. The generic nature of the connectionist, corpus-based, shift-reduce parsing model used for this project has been demonstrated with the model being successfully trained to acquire linguistic knowledge from two different corpora, the Lancaster Parsed Corpus and the BLLIP Wall Street Journal Corpus.

After the parser had been used to learn the underlying linguistic theory used to pre-parse the Lancaster Parsed Corpus, its adaptation to the Wall Street Journal Corpus was without a change to its architecture or algorithm. However, new binary input representations were designed for the parser, to cater for the different word and constituent tags used in the new corpus.

When used with the Lancaster Parsed Corpus, the parser was trained with a set comprising 654 sentences. The test set used had 687 sentences which were not used during training. The parser analysed sentences from the test set with an F-measure of 72.5%. On being adapted to the BLLIP Wall Street Journal Corpus, the training set used consisted of 202 sentences. There were two test sets of sentences that had not been used during training; one had 74 sentences while the other held 1059 sentences. The parser analysed sentences from the 74-sentence test set with an F-measure of 59.1%. It analysed sentences from the 1059-sentence test set with an F-measure of 59.5%.

7.2.2 Semantic Annotation of Nouns in the BLLIP Corpus

An important part of the process of integrating shallow lexical semantic information with syntactic information for full syntactic parsing is the semantic annotation of words in sentences. However, there is a scarcity of large-scale, pre-parsed corpora with lexical semantic annotation as well as part of speech annotation. Also, the lexical semantic tagging models available are too fine-grained for practical use. The need therefore arose for the development of a semi-automatic algorithm for semantic tagging of nouns in the BLLIP Wall Street Journal Corpus. The lexical semantic information used in this semantic annotation algorithm was extracted from WordNet, an online lexical resource. WordNet provides a lexical inheritance system for nouns. Each noun (and pronoun) in the training and test samples from the BLLIP Wall Street Journal Corpus was semantically annotated using the developed algorithm. The semantic classes used for this annotation are the 25 top-level classes (called unique beginners) in WordNet's lexical inheritance system for nouns.

A lot of the nouns annotated are polysemous and could have senses that belong to more than one of the unique beginners. In such cases, the views of Buitelaar [117] are shared in the design of this annotation scheme; there is no disambiguation between the senses, rather they are left underspecified. However, this algorithm allowed a maximum of four of the most frequently used senses to be extracted from WordNet for each noun in BLLIP Wall Street Journal Corpus samples. The frequency of use for each sense (determined by the number of times a sense is tagged in the various semantic concordance texts built up as part of the WordNet project) is also extracted. This frequency forms part of the semantic tags for the nouns.

The semantic tags for nouns and pronouns are extended to noun phrases and preposition phrases which the nouns are part of.

7.2.3 Integration of Shallow Lexical Semantic Information with Syntactic Information in Full Syntactic Parsing

In introducing shallow lexical semantic information to the syntactic parsing process, the main aim of this work is not to improve parsing accuracy per se. Rather, the main aim is to investigate the role that lexical semantic information (of vary levels of abstraction) plays in the syntactic parsing process. This investigation has also provided some insight into two contrasting parsing theories; i.e. whether syntactic parsing should be modelled as a two-stage "Fodorian" process with a lot of stress on compartmentalism and serial processing, or as an integrated constraint-satisfaction process which stresses the importance of interaction between syntactic information and semantics.

On the integration of shallow lexical semantic information with syntactic information in its input representation, the parser learnt with a training set of 206 sentences. Two test sets consisting of 74 and 1059 sentences each were used to test the parsers generalisation performance. Sentences in the test set were not used during training. When presented with sentences from the training set, the parser analysed them with an F-measure of 73.6%. On being presented with the test sentences, the parser generalised to parse 1059 test sentences from the Wall Street Journal Corpus with an F-measure of 56.75%. It analysed the 74-sentence set with an F-measure of 58.17%.

Although the integration of shallow lexical semantic information with syntactic information did not seem to improve the parser's overall training/generalisation performance, given its present configuration, an examination of the parser's

behaviour showed that it did appear to improve the parser's decision concerning preposition phrase attachment.

On the whole, this connectionist parsing model provides a plausible account of natural language acquisition. This is considering its ability to process sentences sequentially and learn the underlying linguistic theory used to annotate the Lancaster Parsed and Wall Street Journal Copora. It is able to handle recursive structure and the potentially long sentences, including complex, multi-clause sentences that result from it. It is able to process long-distance dependencies, such as centre-embeddings. Apart from being able to learn and make use of multiple linguistic constraints, it includes contextual constraints, in the form of look-backs and look-aheads, in its decision making process. Unlike, other parsing models, especially the statistical models (most use test sets that are 6% the size of their training data sets) which need very large training data compared to their test data, this connectionist model is able to generalise to test sets that are larger than its training data set. Its generalisation performance is not deeply affected by test data size.

7.3 Future Work

Work on the connectionist model used in this project suggests that there is a lot of room for improvement. To begin with, a re-structuring of the modular architecture used by this parser could lead to better sentence level performance. Findings show that strong modular performances by the connectionist modules on the failed/mismatched sentences were matched with weak sentence level performance. This brings to the fore a limitation of the modular model, the knock-on effects that occur throughout the shift-reduce parsing process. During the parsing process, the parser's connectionist modules operate in cascade. The right-to-left delimiter (RLD) module first processes the sequences and passes its results, including any errors, if

available, to the left-to-right delimiter (LRD) module. The LRD passes its own results, including any errors, if available to the phrase structure recogniser (PSR) module. To eradicate or drastically reduce the effects of this limitation, it would be necessary to use one recurrent network for the delimitation process. This would result in two connectionist modules, one for delimitation, and the other for phrase structure recognition. Alternatively, the whole parsing process could be assigned to one network, a recurrent network which is capable of handling the sequential and unbounded nature of natural language processing.

Improvement in parsing performance could also be achieved by increasing the size of the training set. This is considering that the parser had better training/test performance on the Lancaster Parsed Corpus (LPC) than on the Wall Street Journal Corpus (WSJC). The training set used to train the parser on the LPC contained 654 sentences. That used to train the parser on the WSLC had 202 sentences. However, in towing this line, the limitations of training set sizes, networks sizes and training times are bound to re-surface. A way around this would be to make use of faster computer resources; research in Artificial Intelligence has over the years been enhanced by growth in computer power and speed. News of the emergence of Tesla supercomputers from NVIDIA Corporation may provide the solution for optimising these neural networks on a large scale in the future. Another way around this would be to explore the use of less computationally excruciating recurrent neural network architectures, such as Echo State Networks [135]. In addition to increasing the training data size, a further optimisation of the network size would be necessary for improved performance. In ensuring the improvement of generalisation performance during network optimisation, weight regularisation techniques [94, 118] such as weight decay, weight elimination and approximate smoother may come in handy.

A re-structuring of the training programme for the parser's modules could also lead to improvement in parsing performance. This would involve a situation where the

parser's modules are re-trained only on data generated from the failed/non-matching sentences. After this session of re-training on these failed/non-matching data, the parser would again be trained on data from the whole test set. In re-structuring the training programme in this way, it would be necessary to avoid over-fitting. This can be done by using the early stopping method of cross-validation [94, 95, 96].

The above recommendations are aimed at improvement in the parser's performance. They would also lead to a better exposure of the effects of integrating lexical semantic information with syntactic information in full syntactic parsing. Further effects of this integration might also be observed with the use of deeper levels of semantic abstraction for the annotated nouns. To be useful for practical use, the level of semantic abstraction should balance the need for additional information whilst ensuring that the sense distinctions are not too fine-grained.

Appendix A: The Penn Treebank II Word Tags

The word tags can be sub-divided into five separate groups: nouns, verbs, prepositions, conjunctions and punctuation. The following tables list the tag, description and bit representation within the encoding scheme defined in Chapter 4 for each tag in each group:

Tag	Description	Bit Representation
NN	Noun, singular or mass	100001
NNS	Noun, plural	100010
NNP	Proper noun, singular	100011
NNPS	Proper noun, plural	100100
PRP	Personal pronoun	100101
PRP-DEI	Personal pronoun, deictic	100110
PRP-PLE	Personal pronoun, pleonastic	100111
PRP$	Possessive pronoun	101000
PRP$-DEI	Possessive pronoun, deictic	101001
PRP$-PLE	Possessive pronoun, pleonastic	101010
JJ	Adjective	101011
JJR	Adjective, comparative	101100
JJS	Adjective, superlative	101101
CD	Cardinal number	101110
DT	Determiner	101111
EX	Existential *there*	110000
FW	Foreign word	110001
LS	List item marker	110010
PDT	Pre-determiner	110011
POS	Possessive ending	110100
SYM	Symbol	110101
WDT	*wh*-determiner	110110
WP	*wh*-pronoun	110111
WP$	Possessive *wh*-pronoun	111000

Tag	Description	Bit Representation
IN	Preposition/subord. Conjunction	11

Tag	Description	Bit Representation
CC	Coordinating conjunction	11

Tag	Description	Bit Representation
VB	Verb, base form	100001
VBD	Verb, past tense	100010
VBG	Verb, gerund/present participle	100011
VBN	Verb, past participle	100100
VBP	Verb, non-3rd ps. Sing. present	100101
VBZ	Verb, 3rd ps. Sing. Present	100110
AUX	Verb, auxilliary e.g. have, been	100111
AUXG	Verb, auxilliary e.g. having, etc	101000
RB	Adverb	101001
RBR	Adverb, comparative	101010
RBS	Adverb, superlative	101011
RP	Particle	101100
MD	Modal	101101
TO	to	101110
UH	Interjection	101111
WRB	*wh*-adverb	110000

Tag	Description	Bit Representation
£	Pound sign	10001
$	Dollar sign	10010
.	Sentence-final punctuation	10011
,	Comma	10100
:	Colon	10101
;	Semi-colon	10110
-LRB-	Left bracket character	10111
-RRB-	Right bracket character	11000
``	Straight double quote	11001
`	Single open/close quote	11010
"	Double open/close quote	11011
-NONE-	Null element	10000

Appendix B: The Penn Treebank II Constituent Tags

Tag	Description	Bit Representation
ADJP	Adjective phrase	101
WHADJP	Wh-adjective phrase	110
QP	Quantifier phrase	111

Tag	Description	Bit Representation
ADVP	Adverb phrase	101
WHADVP	Wh-adverb phrase	110

Tag	Description	Bit Representation
CONJP	Conjunction phrase	1

Tag	Description	Bit Representation
FRAG	Fragment	1

Tag	Description	Bit Representation
INTJ	Interjection	1

Tag	Description	Bit Representation
NP	Noun phrase	1001
NX	Head of complex NP	1010
NAC	Not a constituent	1011
LST	List marker	1100
WHNP	Wh-noun phrase	1101

Tag	Description	Bit Representation
PP	Prepositional phrase	101
WHPP	Wh-prepositional phrase	110

Tag	Description	Bit Representation
PRN	Parenthesis	1

Tag	Description	Bit Representation
RRC	Reduced relative clause	1

Tag	Description	Bit Representation
S1	Root node	1001
S	Simple declarative clause	1010
SBAR	Clause introd. by surbor. Conj.	1011
SBARQ	Direct quest. introd. by wh-word	1100
SINV	Declar. sent. with subj-aux inversion	1101
SQ	Sub-constituent of SBARQ	1110

Tag	Description	Bit Representation
UCP	Unlike coordinated phrase	1

Tag	Description	Bit Representation
VP	Verb phrase	101
PRT	Particle	110

Tag	Description	Bit Representation
X	Unknown/uncertain category	1

Appendix C: The Word Tags Used In The LPC

The word tags can be sub-divided into five separate groups: nouns, verbs, prepositions, conjunctions and punctuation. The following tables lists the tag, description and bit representation within the encoding scheme used in this work for each tag in each group.

83 Word Tags for Nouns

Tag	Description	Bit Rep.
ABL	pre-qualifier in a noun phrase (QUITE, RATHER, SUCH)	1 0 0 0 0 0 0 1
ABN	pre-quantifier in a noun phrase (ALL, HALF)	1 0 0 0 0 0 1 0
AP	post-determiner (FEW, FEWER, FORMER)	1 0 0 0 0 0 1 1
AP$	OTHER'S	1 0 0 0 0 1 0 0
APS	OTHERS	1 0 0 0 0 1 0 1
APS$	OTHERS'	1 0 0 0 0 1 1 0
AT	singular article (A, AN, EVERY)	1 0 0 0 0 1 1 1
ATI	singular or plural article (THE, NO)	1 0 0 0 1 0 0 0
CD	cardinal number (2, 3, etc; TWO, THREE, THOUSAND)	1 0 0 0 1 0 0 1
CS$	cardinal number + genitive	1 0 0 0 1 0 1 0
CD-CD	hyphenated pair of cardinal numbers (e.g. 1988-90)	1 0 0 0 1 0 1 1
CD1	ONE	1 0 0 0 1 1 0 0
CD1$	ONE'S	1 0 0 0 1 1 0 1
CD1S	ONES	1 0 0 0 1 1 1 0
CDS	cardinal number+plural(TENS, MILLIONS, DOZENS, etc)	1 0 0 0 1 1 1 1
DT	singular determiner (ANOTHER, EACH, THAT, THIS)	1 0 0 1 0 0 0 0
DT$	singular determiner + genitive (ANOTHER'S)	1 0 0 1 0 0 0 1
DTI	determiner neutral for number (ANY, ENOUGH, SOME)	1 0 0 1 0 0 1 0
DTS	plural determiner (THESE, THOSE)	1 0 0 1 0 0 1 1
DTX	determiner / double conjunction (EITHER, NEITHER)	1 0 0 1 0 1 0 0
EX	existential THERE	1 0 0 1 0 1 0 1
JJ	adjective (general)	1 0 0 1 0 1 1 0
JJB	attributive adjective	1 0 0 1 0 1 1 1
JNP	adjective with word-initial cap; e.g. WELSH, KEYNESIAN	1 0 0 1 1 0 0 0
JJR	comparative adjective	1 0 0 1 1 0 0 1
JJT	superlative adjective	1 0 0 1 1 0 1 0
NC	cited word as singular noun (e.g. "LED is a verb")	1 0 0 1 1 0 1 1

NN	singular common noun	1 0 0 1 1 1 0 0
NNP	sing. common noun; word-initial cap; e.g. LONDONER	1 0 0 1 1 1 0 1
NNPS	plural common noun; word-initial cap; e.g. LONDONERS	1 0 0 1 1 1 1 0
NNPS$	plu. common noun; word-init. cap; gen. : LONDONERS'	1 0 0 1 1 1 1 1
NNP$	sing. common noun; word-init.cap; gen.: LONDONER'S	1 0 1 0 0 0 0 0
NNS	plural common noun	1 0 1 0 0 0 0 1
NNS$	plural common noun + genitive	1 0 1 0 0 0 1 0
NNU	singular unit of measurement (e.g. IN. KG.)	1 0 1 0 0 0 1 1
NNUS	plural unit of measurement (e.g. INS. KGS.)	1 0 1 0 0 1 0 0
NNUS$	plural unit of measurement + genitive	1 0 1 0 0 1 0 1
NP	singular proper noun	1 0 1 0 0 1 1 0
NPS	plural proper noun	1 0 1 0 0 1 1 1
NPS$	plural proper noun + genitive	1 0 1 0 1 0 0 0
NP$	singular proper noun + genitive	1 0 1 0 1 0 0 1
NPL	singular locative noun; word-initial cap.; e.g. ISLAND	1 0 1 0 1 0 1 0
NPLS	plural locative noun; word-initial cap.; e.g. ISLANDS	1 0 1 0 1 0 1 1
NPLS$	plu. locative noun; word-init. cap; + gen.; e.g. ISLANDS'	1 0 1 0 1 1 0 0
NPL$	sing. locative noun; word-init. cap; + gen.: ISLAND'S	1 0 1 0 1 1 0 1
NPT	singular titular noun; word-initial cap.; e.g. DR.	1 0 1 0 1 1 1 0
NPTS	plural titular noun; word-initial cap.; e.g. MESSRS.	1 0 1 0 1 1 1 1
NPTS$	plu. titular noun; word-init. cap.; + gen.; e.g. QUEENS'	1 0 1 1 0 0 0 0
NR	singular adverbial noun (JANUARY, MONDAY, EAST)	1 0 1 1 0 0 0 1
NR$	singular adverbial noun + genitive	1 0 1 1 0 0 1 0
NRS	plural adverbial noun	1 0 1 1 0 0 1 1
OD	ordinal number (1ST, 2ND, etc; FIRST, SECOND, etc)	1 0 1 1 0 1 0 0
PN	nominal pronoun (ANYBODY, ANYONE, EVERYONE etc)	1 0 1 1 0 1 0 1
PN$	nominal pronoun + genitive	1 0 1 1 0 1 1 0
PP$	possessive determiner (MY, YOUR, etc)	1 0 1 1 0 1 1 1
PPS$	possessive pronoun (MINE, YOURS, etc)	1 0 1 1 1 0 0 0
PP1A	personal pronoun, 1st pers sing nom (I)	1 0 1 1 1 0 0 1
PP1AS	personal pronoun, 1st pers plur nom (WE)	1 0 1 1 1 0 1 0
PP1O	personal pronoun, 1st pers sing acc (ME)	1 0 1 1 1 0 1 1
PP1OS	personal pronoun, 1st pers plur acc (US, 'S)	1 0 1 1 1 1 0 0
PP2	personal pronoun, 2nd pers (YOU, THOU, THEE, YE)	1 0 1 1 1 1 0 1
PP3	personal pronoun, 3rd pers sing nom+acc (IT)	1 0 1 1 1 1 1 0
PP3A	personal pronoun, 3rd pers sing nom (HE, SHE)	1 0 1 1 1 1 1 1
PP3AS	personal pronoun, 3rd pers plur nom (THEY)	1 1 0 0 0 0 0 0
PP3O	personal pronoun, 3rd pers plur acc (HIM, HER)	1 1 0 0 0 0 0 1

PP3OS	personal pronoun, 3rd pers plur acc (THEM, 'EM)	1 1 0 0 0 0 1 0
PPL	singular reflexive pronoun	1 1 0 0 0 0 1 1
PPLS	plural reflexive pronoun	1 1 0 0 0 1 0 0
QL	qualifier (AS, AWFULLY, LESS, MORE, SO, TOO, VERY, etc)	1 1 0 0 0 1 0 1
QLP	post-qualifier (ENOUGH, INDEED)	1 1 0 0 0 1 1 0
WDT	WH-determiner (WHAT, WHATEVER,WHICH)	1 1 0 0 0 1 1 1
WP	WH-pronoun, nom+acc (WHO, WHOEVER, THAT)	1 1 0 0 1 0 0 0
WP$	WH-pronoun, genitive (WHOSE)	1 1 0 0 1 0 0 1
WPA	WH-pronoun, nom (WHOSOEVER)	1 1 0 0 1 0 1 0
WPO	WH-pronoun, acc (WHOM, WHOMSOEVER)	1 1 0 0 1 0 1 1
PP$$	possessive pronoun (MINE,YOURS etc)	1 1 0 0 1 1 0 0
NN$	singular common noun + genitive	1 1 0 0 1 1 0 1
NPT$	titular noun with w.i.c + genitive	1 1 0 0 1 1 1 0
WDTR	WH-determiner - relative e.g. WHICH	1 1 0 0 1 1 1 1
WP$R	WH-pronoun - relative - gen e.g. WHOSE	1 1 0 1 0 0 0 0
WPOR	WH-pronoun - relative - acc e.g. WHOM	1 1 0 1 0 0 0 1
WPR	WH-pronoun - relative - nom+acc e.g. THAT,relative WHO	1 1 0 1 0 0 1 0
WRB	WH-verb (HOW,WHEN)	1 1 0 1 0 0 1 1

4 Word Tags for Prepositions

Tag	Description	Bit Representation
IN	preposition (general)	1 0 0 1
INF	FOR as a preposition	1 0 1 0
INO	OF as a preposition	1 0 1 1
INW	WITH as a preposition	1 1 0 0

33 Word Tags for Verbs

Tag	Description	Bit Representation
BE	BE	1 0 0 0 0 0 1
BED	WERE	1 0 0 0 0 1 0
BEDZ	WAS	1 0 0 0 0 1 1
BEG	BEING	1 0 0 0 1 0 0
BEM	AM	1 0 0 0 1 0 1
BEN	BEEN	1 0 0 0 1 1 0
BER	ARE, 'RE	1 0 0 0 1 1 1
BEZ	IS, 'S	1 0 0 1 0 0 0
DO	DO	1 0 0 1 0 0 1
DOD	DID	1 0 0 1 0 1 0

DOZ	DOES	1 0 0 1 0 1 1
HV	HAVE	1 0 0 1 1 0 0
HVD	HAD, 'D (past tense)	1 0 0 1 1 0 1
HVG	HAVING	1 0 0 1 1 1 0
HVN	HAD (past participle)	1 0 0 1 1 1 1
HVZ	HAS, 'S	1 0 1 0 0 0 0
MD	modal auxiliary	1 0 1 0 0 0 1
RB	adverb (general)	1 0 1 0 0 1 0
RB$	adverb + genitive (ELSE'S)	1 0 1 0 0 1 1
RBR	comparative adverb	1 0 1 0 1 0 0
RBT	superlative adverb	1 0 1 0 1 0 1
RI	adverb (homograph of preposition: BELOW, NEAR, etc)	1 0 1 0 1 1 0
RN	nominal adverb (HERE, NOW, THERE, THEN, etc)	1 0 1 0 1 1 1
RP	adverbial particle (BACK, DOWN, OFF, etc)	1 0 1 1 0 0 0
TO	infinitival TO	1 0 1 1 0 0 1
UH	interjection	1 0 1 1 0 1 0
VB	base form of lexi. verb (uninflected pres. tense, infinitive)	1 0 1 1 0 1 1
VBD	past tense of lexical verb	1 0 1 1 1 0 0
VBG	present participle or gerund of lexical verb	1 0 1 1 1 0 1
VBN	past participle of lexical verb	1 0 1 1 1 1 0
VBZ	3rd person singular of verb	1 0 1 1 1 1 1
WRB	WH-adverb (HOW, WHEN, WHERE, etc)	1 1 0 0 0 0 0
XNOT	NOT, N'T	1 1 0 0 0 0 1

3 Word Tags for Conjunctions

Tag	**Description**	**Bit Representation**
ABX	pre-quantifier / double conjunction (e.g. BOTH)	1 0 1
CC	coordinating conjunction (e.g. AND, AND/OR, BUT, OR, YET)	1 1 0
CS	subordinating conjunction (e.g. AFTER, ALTHOUGH, etc)	1 1 1

20 Word Tags for Punctuation

Tag	**Description**	**Bit Representation**
^	null	0 0 0 0 0
ZZ	letter of the alphabet (E, X, etc).	1 0 0 0 1
!	exclamation mark (!)	1 0 0 1 0
&FO	formula	1 0 0 1 1
&FW	foreign word	1 0 1 0 0
(	left bracket	1 0 1 0 1

[	left bracket	1 0 1 0 1
)	right bracket	1 0 1 1 0
]	right bracket	1 0 1 1 0
*'	begin quote	1 0 1 1 1
*"	begin quote	1 0 1 1 1
**'	end quote	1 1 0 0 0
**"	end quote	1 1 0 0 0
-	dash	1 1 0 0 1
,	comma	1 1 0 1 0
?	question mark	1 1 0 1 1
...	ellipsis	1 1 1 0 0
:	colon	1 1 1 0 1
;	semicolon	1 1 1 1 0
.	full stop	1 1 1 1 1

Appendix D: The Constituent Tags Used In The LPC

The constituent tags can be sub-divided into five main groups : sentence tags, finite clause tags, non-finite and verbless clause tags, major phrase tags, and minor phrase tags. The following table lists the tag, description and
bit representation within the encoding scheme used in this work.

3 Sentence Tags

Tag	Description	Bit Representation
S	sentence.	1 0 1
Sq	piece of direct quotation.	1 1 0
Si	interpolated sentence.	1 1 1

5 Finite Clause Tags

Tag	Description	Bit Representation
F	finite subor. clause i.e. a clause which contains a finite verb.	1 0 0 1
Fa	finite adverbial clause(e.g. finite subordinate clause of time etc)	1 0 1 0
Fc	comparative clause, normally beginning with `than' or `as'.	1 0 1 1
Fn	finite nominal clause (subord clause func in pos of Noun PH)	1 1 0 0
Fr	relative clause - whether restrictive or non-restrictive.	1 1 0 1

9 Non-finite And Verbless Clause Tags

Tag	Description	Bit Representation
T	nonfinite clause.	1 0 0 0 1
Ti	to-infinitive clause.	1 0 0 1 0
Tg	-ing clause.	1 0 0 1 1
Tn	past participle clause.	1 0 1 0 0
Tb `	bare infinitive clause'.	1 0 1 0 1
Tf	subject of the infinitive which is introduced by `for'.	1 0 1 1 0
W	nonfinite or verbless clause that is introduced by with.	1 0 1 1 1
L	verbless clause that is not intro. by subordinating conjunction.	1 1 0 0 0

17 Major Phrase Tags

Tag	Description	Bit Representation
V	A finite "verb phrase" i.e. one that excludes objects, complements	1 0 0 0 0 1
Vo	Used when a verb phrase is split into two parts by subj-aux inv.	

	o=operator	1 0 0 0 1 0
Vr	Used when a verb phrase is split into two parts by subj-aux inversion.	
	r=remaindr	1 0 0 0 1 1
Vi	Label for nonfinite verb phrases(VP). i.e VP's that are	
	VP's of Ti,Tg and Tn.	1 0 0 1 0 0
Vg	Label for nonfinite verb phrases(VP). i.e VP's that are VP's of	
	Ti,Tg and Tn.	1 0 0 1 0 1
Vn	Label for nonfinite verb phrases(VP). i.e VP's that are VP's of	
	Ti,Tg and Tn.	1 0 0 1 1 0
N	Label for a noun phrase, whether it is a single word or a sequence	
	of words.	1 0 0 1 1 1
Na	A noun phrase marked as subject of the verb.	1 0 1 0 0 0
Nq	A wh- noun phrase, such as `who', `which', `which car',	
	`what time'.	1 0 1 0 0 1
J	An adjective phrase such as `happy', `very tall' etc.	1 0 1 0 1 0
Jq	A phrase beginning with a wh-word e.g. `How old'.	1 0 1 0 1 1
P	A prepositional phrase, e.g. `in London' or `on arriving	
	at the station'.	1 0 1 1 0 0
Pq	A prepositional phrase with a wh-word, e.g. `on whose behalf',	
	`in which case'.	1 0 1 1 0 1
Po	A prepositional phrase beginning with the preposition 'of'.	1 0 1 1 1 0
Poq	A prepositional phrase beginning with the preposition 'of'	
	with wh-word?	1 1 0 0 0 1
R	An adverb phrase, e.g. `there',`quickly' or a sequence such as `quite	
	often' etc	1 0 1 1 1 1
Rq	an adverb phrase beginning with a wh-word, e.g. `How do you feel?',	
	or `how long'	1 1 0 0 0 0

7 Minor Phrase Tags

Tag	**Description**	**Bit Representation**
M	`numeric phrase' when such an expression is part of a	
	noun phrase.	1 0 0 1
D	`determiner phrase'.	1 0 1 0
Dq	determiner phrase beginning with a wh-word.	1 0 1 1
G	genitive phrase - phrase with two or more words acting as the	
	genitive in a noun.	1 1 0 0
X	negative word 'not' when acting as an independent element of	
	a clause.	1 1 0 1

E	label used for existential `there' i.e 'There' is nothing wrong.	1 1 1 0
U	tag used for an exclamatory word such as `Oh','yes', or 'no'.	1 1 1 1

Appendix E: Parse Failures made on the WSJ Corpus Training Set (Using Syntactic Information Only)

Parse failed - could not create a valid parent for NP

Desired parse : (S1 (S (NP (NP (NP NNP NNPS POS) NNP NN NN) (PP IN (NP NNP NNP))) (VP VBD (PP IN (NP CD NN)) (PP IN (ADVP (NP DT NN) RBR)) , (PP VBG (PP TO (NP (NP DT NNP) (PP IN (NP NNP NNP)))))) .))

wsj_Concord-II's attempt ....

State 1 : (NP NNP NNP)
Description : Noun phrase

State 2 : (PP IN (NP NNP NNP))
Description : Prepositional phrase

State 3 : (NP DT NNP)
Description : Noun phrase

State 4 : (NP (NP DT NNP) (PP IN (NP NNP NNP)))
Description : Noun phrase

State 5 : (PP TO (NP (NP DT NNP) (PP IN (NP NNP NNP))))
Description : Prepositional phrase

State 6 : (PP VBG (PP TO (NP (NP DT NNP) (PP IN (NP NNP NNP)))))
Description : Prepositional phrase

State 7 : (NP DT NN)
Description : Noun phrase

Parse failed - could not create a valid parent for NP

Desired parse : (S1 (S (PP IN (PP TO (S (NP -NONE-) (VP VBG (NP DT NN) (PP IN (NP CD)))))) , (NP PRP) (VP VBD (SBAR -NONE- (S (NP DT NNP) `` (VP AUX RB (VP VB (S (NP -NONE-) (VP TO (VP VB (S (NP -NONE-) (VP VBG (NP DT NN) (PP (ADVP RB) IN (NP (NP DT NN) (PP IN (NP NN)))))))))))))) . ''))

wsj_Concord-II's attempt ....

State 1 : (NP NN)
Description : Noun phrase

State 2 : (PP IN (NP NN))
Description : Prepositional phrase

State 3 : (NP DT NN)
Description : Noun phrase

State 4 : (VP IN (NP DT NN) (PP IN (NP NN)))
Description : Verb phrase

State 5 : (ADVP RB)
Description : Adverb phrase

State 6 : (NP DT NN)
Description : Noun phrase

State 7 : (VP VBG (NP DT NN) (ADVP RB) (VP IN (NP DT NN) (PP IN (NP NN))) .)
Description : Verb phrase

State 8 : (NP -NONE-)
Description : Noun phrase

State 9 : (S (NP -NONE-) (VP VBG (NP DT NN) (ADVP RB) (VP IN (NP DT NN) (PP IN (NP NN))) .))
Description : Simple declarative clause

State 10 : (VP VB (S (NP -NONE-) (VP VBG (NP DT NN) (ADVP RB) (VP IN (NP DT NN) (PP IN (NP NN))) .)))
Description : Verb phrase

State 11 : (VP TO (VP VB (S (NP -NONE-) (VP VBG (NP DT NN) (ADVP RB) (VP IN (NP DT NN) (PP IN (NP NN))) .))))
Description : Verb phrase

State 12 : (NP -NONE-)
Description : Noun phrase

State 13 : (S (NP -NONE-) (VP TO (VP VB (S (NP -NONE-) (VP VBG (NP DT NN) (ADVP RB) (VP IN (NP DT NN) (PP IN (NP NN))) .)))))
Description : Simple declarative clause

State 14 : (VP VB (S (NP -NONE-) (VP TO (VP VB (S (NP -NONE-) (VP VBG (NP DT NN) (ADVP RB) (VP IN (NP DT NN) (PP IN (NP NN))) .))))))
Description : Verb phrase

State 15 : (VP RB (VP VB (S (NP -NONE-) (VP TO (VP VB (S (NP -NONE-) (VP VBG (NP DT NN) (ADVP RB) (VP IN (NP DT NN) (PP IN (NP NN))) .)))))))
Description : Verb phrase

State 16 : (VP AUX (VP RB (VP VB (S (NP -NONE-) (VP TO (VP VB (S (NP -NONE-) (VP VBG (NP DT NN) (ADVP RB) (VP IN (NP DT NN) (PP IN (NP NN))) .)))))))) '')
Description : Verb phrase

State 17 : (NP DT NNP)
Description : Noun phrase

State 18 : (NP -NONE- (NP DT NNP) ``)
Description : Noun phrase

Parse failed - could not create a valid parent for NP

Desired parse : (S1 (S (PP IN (NP NNP)) , (NP NN NNS) (VP (VP VBD (NP (NP DT NN) (PP IN (NP NN NNS)))) CC (VP (ADVP RB) VBD (NP (NP DT NN) (PP IN (NP (NP NNS) (PP TO (NP NNP))))) (PP IN (NP (NP DT NN) (PP IN (NP NNP NNS))))))))

wsj_Concord-II's attempt ....

State 1 : (NP NNP NNS)
Description : Noun phrase

State 2 : (PP IN (NP NNP NNS))
Description : Prepositional phrase

State 3 : (NP DT NN)
Description : Noun phrase

Parse failed - could not create a valid parent for NP

Desired parse : (S1 (S (NP DT NNP NN) (VP VBD (NP NNP NNPS) (SBAR IN (S (NP (NP DT NNP NNP) (PP IN (NP NNP NNP))) (VP MD RB (VP VB (NP (NP NNS) (PP IN (NP (NP JJ NNS) (VP VBD (NP NNP NNP)))))))))))

wsj_Concord-II's attempt ....

State 1 : (NP NNP NNP)
Description : Noun phrase

State 2 : (VP VBD (NP NNP NNP))
Description : Verb phrase

State 3 : (NP JJ NNS)
Description : Noun phrase

Parse failed - could not create a valid parent for NP

Desired parse : (S1 (PRN -LRB- (VP VBP (NP (NP VBN NN) : `` (S (NP (NP NNP POS) NNP) (VP VBZ (S (NP NNP NNP POS) (VP VBG '' (S (NP -NONE-) (VP TO (VP VB (NP DT JJ NN NN)))))))) '' : (NP (NP NNP) (NP NNP CD , CD)))) -RRB-))

wsj_Concord-II's attempt ....

State 1 : (NP NNP CD , CD)
Description : Noun phrase

State 2 : (ADVP '')
Description : Adverb phrase

State 3 : (NP NNP)
Description : Noun phrase

State 4 : (NP (NP NNP) (NP NNP CD , CD))
Description : Noun phrase

State 5 : (NP DT JJ NN NN)
Description : Noun phrase

State 6 : (VP VB (NP DT JJ NN NN) (ADVP ''))
Description : Verb phrase

State 7 : (VP TO (VP VB (NP DT JJ NN NN) (ADVP '')) :)
Description : Verb phrase

State 8 : (VP VBG)
Description : Verb phrase

State 9 : (VP (VP VBG) '' -NONE- (VP TO (VP VB (NP DT JJ NN NN) (ADVP '')) :) (NP (NP NNP) (NP NNP CD , CD)))
Description : Verb phrase

State 10 : (NP NNP NNP)
Description : Noun phrase

State 11 : (NP POS)
Description : Noun phrase

State 12 : (NP (NP POS) (NP NNP NNP))
Description : Noun phrase

State 13 : (S (NP (NP POS) (NP NNP NNP)) (VP (VP VBG) '' -NONE- (VP TO (VP VB (NP DT JJ NN NN) (ADVP '')) :) (NP (NP NNP) (NP NNP CD , CD))))
Description : Simple declarative clause

State 14 : (VP VBZ (S (NP (NP POS) (NP NNP NNP)) (VP (VP VBG) '' -NONE- (VP TO (VP VB (NP DT JJ NN NN) (ADVP '')) :) (NP (NP NNP) (NP NNP CD , CD)))) -RRB-)
Description : Verb phrase

State 15 : (NP NNP POS)
Description : Noun phrase

State 16 : (NP `` (NP NNP POS) NNP)
Description : Noun phrase

Parse failed - could not create a valid parent for S1

Desired parse : (S1 (S (PP IN (NP JJ NNS)) , (NP (NP NNS) (PP IN (NP NNP))) (VP AUX (VP VBN (NP (NP NNS) (PP IN (NP NN))) (PP IN (NP (NP DT NN NNS) (PP IN (NP (NP NNS) (VP VBN (S (NP -NONE-) (ADJP JJ (PP IN (NP NN)))))))))) .))

wsj_Concord-II's attempt ....

State 1 : (NP NN)
Description : Noun phrase

State 2 : (PP IN (NP NN))
Description : Prepositional phrase

State 3 : (UCP JJ)
Description : Unlike coordinated phrase

State 4 : (S1 .)
Description : Root node

Parse failed - could not create a valid parent for NP

Desired parse : (S1 (S CC (NP DT NN (S (NP -NONE-) (VP TO (VP VB (NP (NP (NP DT NN POS) JJ NNS NN) (PRN : (SBAR (WHNP WDT) (S (NP -NONE-) (VP MD (VP VB (NP DT NNS) (PP IN (NP NN)))))) :)))))) (VP AUX (VP VBG (NP NN))) .))

wsj_Concord-II's attempt ....

State 1 : (NP NN)
Description : Noun phrase

State 2 : (VP VBG (NP NN))
Description : Verb phrase

State 3 : (VP AUX (VP VBG (NP NN)))
Description : Verb phrase

State 4 : (NP NN)
Description : Noun phrase

State 5 : (PP IN (NP NN))
Description : Prepositional phrase

State 6 : (NP DT NNS)
Description : Noun phrase

State 7 : (VP VB (NP DT NNS) (PP IN (NP NN)))
Description : Verb phrase

State 8 : (VP MD (VP VB (NP DT NNS) (PP IN (NP NN))))
Description : Verb phrase

State 9 : (NP -NONE-)
Description : Noun phrase

State 10 : (S (NP -NONE-) (VP MD (VP VB (NP DT NNS) (PP IN (NP NN)))))
Description : Simple declarative clause

State 11 : (WHNP WDT)
Description : Wh-noun phrase

State 12 : (SBAR (WHNP WDT) (S (NP -NONE-) (VP MD (VP VB (NP DT NNS) (PP IN (NP NN))))))
Description : Clause introduced by sub-ordinating conjunction

State 13 : (PRN : (SBAR (WHNP WDT) (S (NP -NONE-) (VP MD (VP VB (NP DT NNS) (PP IN (NP NN)))))) :)
Description : Parenthetical

State 14 : (NP DT NN POS)
Description : Noun phrase

State 15 : (NP (NP DT NN POS) JJ NNS NN)
Description : Noun phrase

Parse failed - could not create a valid parent for NP

Desired parse : (S1 (S (NP DT NN) (ADVP RB) (VP VBD (NP (NP DT NN NN) (PP IN (NP (QP $ CD CD) -NONE-)) (PP IN (NP JJ JJ NNS)))) .))

wsj_Concord-II's attempt ....

State 1 : (NP JJ JJ NNS)
Description : Noun phrase

State 2 : (PP IN (NP JJ JJ NNS))
Description : Prepositional phrase

State 3 : (QP $ CD CD)
Description : Adjective phrase (Quantitative)

State 4 : (PP IN)
Description : Prepositional phrase

State 5 : (NP DT NN NN)
Description : Noun phrase

State 6 : (NP (NP DT NN NN) (PP IN))
Description : Noun phrase

Parse failed - could not create a valid parent for NP

Desired parse : (S1 (S (PP IN (PP IN (NP (NP (NP DT NN POS) NN NN) (PP TO (NP JJ NNS))))) , (NP DT NN) (VP VBD : `` (S (NP JJS NNS) (VP VBP (PP IN (NP PRP$ NNS)) (PP IN (NP (NP DT NN) (PP IN (NP (NP (ADJP JJ CC JJ) NNS) (ADJP JJ)))))))) .))

wsj_Concord-II's attempt ....

State 1 : (ADJP JJ)
Description : Adjective phrase

State 2 : (ADJP JJ CC JJ)
Description : Adjective phrase

State 3 : (NP (ADJP JJ CC JJ) NNS)
Description : Noun phrase

State 4 : (NP (NP (ADJP JJ CC JJ) NNS) (ADJP JJ))
Description : Noun phrase

State 5 : (PP IN (NP (NP (ADJP JJ CC JJ) NNS) (ADJP JJ)))
Description : Prepositional phrase

State 6 : (NP DT NN)
Description : Noun phrase

State 7 : (NP (NP DT NN) (PP IN (NP (NP (ADJP JJ CC JJ) NNS) (ADJP JJ))))
Description : Noun phrase

State 8 : (PP IN (NP (NP DT NN) (PP IN (NP (NP (ADJP JJ CC JJ) NNS) (ADJP JJ)))))
Description : Prepositional phrase

State 9 : (NP PRP$ NNS)
Description : Noun phrase

State 10 : (NP (NP PRP$ NNS) (PP IN (NP (NP DT NN) (PP IN (NP (NP (ADJP JJ CC JJ) NNS) (ADJP JJ))))))
Description : Noun phrase

NNS , VBP .
Parse failed - the tail of a phrase could not be found!
Desired parse : (S1 (S (NP NNS) , (VP VBP) .))

wsj_Concord-II's attempt ....

State 1 : (VP VBP)

Description : Verb phrase

State 2 : (VP NNS , (VP VBP) .)
Description : Verb phrase

`` DT AUX DT JJ NN , '' VBD -NONE- NNP NNP NNP , CD IN DT NN POS NNS **.Parse failed - the head of a phrase could not be found!**
Desired parse : (S1 (SINV `` (S (NP DT) (VP AUX (NP DT JJ NN))) , '' (VP VBD (S -NONE-)) (NP (NP NNP NNP NNP) , (NP (NP CD) (PP IN (NP (NP DT NN POS) NNS)))) .))

wsj_Concord-II's attempt ....

State 1 : (NP DT NN POS)
Description : Noun phrase

State 2 : (PP IN (NP DT NN POS) NNS)
Description : Prepositional phrase

State 3 : (NP CD)
Description : Noun phrase

State 4 : (NP (NP CD) (PP IN (NP DT NN POS) NNS))
Description : Noun phrase

State 5 : (NP NNP NNP NNP)
Description : Noun phrase

State 6 : (NP (NP NNP NNP NNP) , (NP (NP CD) (PP IN (NP DT NN POS) NNS)))
Description : Noun phrase

State 7 : (S -NONE-)
Description : Simple declarative clause

State 8 : (VP VBD (S -NONE-) (NP (NP NNP NNP NNP) , (NP (NP CD) (PP IN (NP DT NN POS) NNS))))
Description : Verb phrase

State 9 : (NP DT JJ NN)
Description : Noun phrase

State 10 : (VP AUX (NP DT JJ NN))
Description : Verb phrase

State 11 : (NP DT)
Description : Noun phrase

Parse failed - could not create a valid parent for NP

Desired parse : (S1 (S (S (NP -NONE-) (VP VBG (PP IN (NP (NP DT NN) (PP IN (NP PRP$ NNP NNP NN)))) (PP (NP DT JJ NNS) IN (NP NN NN)))) , (NP NNP) (VP AUX (ADJP JJ (PP IN (S (NP -NONE-) (VP VBG (PRT RP) (NP (NP DT NNS) (SBAR (WHNP WDT) (S (NP -NONE-) (PRN , (S (NP PRP) (VP VBZ (SBAR -NONE- (S -NONE-)))) ,) (NP -NONE-) (VP VBP (NP NNP)))))))))) .))

wsj_Concord-II's attempt ....

State 1 : (NP NNP)
Description : Noun phrase

State 2 : (VP VBP (NP NNP))
Description : Verb phrase

State 3 : (NP -NONE-)
Description : Noun phrase

State 4 : (S -NONE-)
Description : Simple declarative clause

State 5 : (S -NONE- (S -NONE-) ,)
Description : Simple declarative clause

State 6 : (VP VBZ (S -NONE- (S -NONE-) ,))
Description : Verb phrase

State 7 : (NP PRP)
Description : Noun phrase

State 8 : (S (NP PRP) (VP VBZ (S -NONE- (S -NONE-) ,)) (NP -NONE-))
Description : Simple declarative clause

State 9 : (NP WDT)
Description : Noun phrase

State 10 : (NP DT NNS)
Description : Noun phrase

State 11 : (NP (NP DT NNS) (NP WDT))
Description : Noun phrase

State 12 : (PRT RP)
Description : Particle; category for words that should be tagged RP

State 13 : (VP VBG (PRT RP) (NP (NP DT NNS) (NP WDT)))
Description : Verb phrase

State 14 : (NP -NONE-)
Description : Noun phrase

State 15 : (S (NP -NONE-) (VP VBG (PRT RP) (NP (NP DT NNS) (NP WDT))))
Description : Simple declarative clause

State 16 : (PP IN (S (NP -NONE-) (VP VBG (PRT RP) (NP (NP DT NNS) (NP WDT)))))

Description : Prepositional phrase

State 17 : (NP JJ)
Description : Noun phrase

State 18 : (VP AUX (NP JJ) (PP IN (S (NP -NONE-) (VP VBG (PRT RP) (NP (NP DT NNS) (NP WDT))))))
Description : Verb phrase

State 19 : (NP NNP)
Description : Noun phrase

State 20 : (S (NP NNP) (VP AUX (NP JJ) (PP IN (S (NP -NONE-) (VP VBG (PRT RP) (NP (NP DT NNS) (NP WDT)))))))
Description : Simple declarative clause

State 21 : (NP NN NN)
Description : Noun phrase

State 22 : (NP (NP NN NN) , (S (NP NNP) (VP AUX (NP JJ) (PP IN (S (NP -NONE-) (VP VBG (PRT RP) (NP (NP DT NNS) (NP WDT)))))))
Description : Noun phrase

Parse failed - could not create a valid parent for VP

Desired parse : (S1 (S (NP DT NN) (VP MD (VP VB (S (NP NNP) (VP TO (VP VB (NP (NP DT (ADJP CD NN) NN NN) (VP VBN (NP -NONE-) (PP IN (NP PRP$ NN (S (NP -NONE-) (VP TO (VP VB)))))))))) , (SBAR RB IN (S (NP NNP CC NNP) (VP AUX RB (VP VB (NP PRP$ NN))))))) .))

wsj_Concord-II's attempt ....

State 1 : (NP PRP$ NN)
Description : Noun phrase

State 2 : (VP VB (NP PRP$ NN))
Description : Verb phrase

State 3 : (VP AUX RB (VP VB (NP PRP$ NN)))
Description : Verb phrase

State 4 : (NP NNP CC NNP)
Description : Noun phrase

State 5 : (S (NP NNP CC NNP) (VP AUX RB (VP VB (NP PRP$ NN))))
Description : Simple declarative clause

State 6 : (PP IN (S (NP NNP CC NNP) (VP AUX RB (VP VB (NP PRP$ NN)))))
Description : Prepositional phrase

State 7 : (UCP RB (PP IN (S (NP NNP CC NNP) (VP AUX RB (VP VB (NP PRP$ NN))))))
Description : Unlike coordinated phrase

State 8 : (VP VB)
Description : Verb phrase

Parse failed - could not create a valid parent for NP

Desired parse : (S1 (S (NP DT NN) (VP MD (VP AUX (VP VBN (NP -NONE-) (PP IN (NP (NP RB CD) CC (NP DT JJ))) (PP IN IN (NP (NP DT NN) (PP IN (NP NN NNS))))))) . ''))

wsj_Concord-II's attempt ....

State 1 : (NP NN NNS)
Description : Noun phrase

State 2 : (PP IN (NP NN NNS))

Description : Prepositional phrase

State 3 : (NP DT NN)
Description : Noun phrase

State 4 : (PP IN (NP DT NN) (PP IN (NP NN NNS)))
Description : Prepositional phrase

State 5 : (PP IN (PP IN (NP DT NN) (PP IN (NP NN NNS))))
Description : Prepositional phrase

State 6 : (NP DT JJ)
Description : Noun phrase

State 7 : (NP RB CD)
Description : Noun phrase

State 8 : (NP (NP RB CD) CC (NP DT JJ))
Description : Noun phrase

State 9 : (PP IN (NP (NP RB CD) CC (NP DT JJ)) (PP IN (PP IN (NP DT NN) (PP IN (NP NN NNS)))))
Description : Prepositional phrase

State 10 : (NP -NONE-)
Description : Noun phrase

State 11 : (VP VBN (NP -NONE-) (PP IN (NP (NP RB CD) CC (NP DT JJ)) (PP IN (PP IN (NP DT NN) (PP IN (NP NN NNS))))) .)
Description : Verb phrase

State 12 : (VP AUX (VP VBN (NP -NONE-) (PP IN (NP (NP RB CD) CC (NP DT JJ)) (PP IN (PP IN (NP DT NN) (PP IN (NP NN NNS))))) .) '')
Description : Verb phrase

State 13 : (ADVP MD)
Description : Adverb phrase

State 14 : (NP (ADVP MD))
Description : Noun phrase

PRP RB VBZ IN NN : IN DT NN IN JJ , JJ NNS JJ IN NNP NNP : AUX VBG DT JJ NN IN IN NNP NNP CC NNP NN .

Parse failed - the tail of a phrase could not be found!

Desired parse : (S1 (S (NP PRP) (ADVP RB) (VP VBZ (SBAR IN (S (NP (NP NN) (PRN : (PP IN (NP (NP DT NN) (PP IN (NP (NP JJ , JJ NNS) (PP JJ IN (NP NNP NNP)))))) :)) (VP AUX (VP VBG (NP DT JJ NN) (PP IN (PP IN (NP NNP NNP CC NNP NN)))))))) .))

wsj_Concord-II's attempt ....

State 1 : (NP NNP NNP CC NNP NN)
Description : Noun phrase

State 2 : (PP IN (NP NNP NNP CC NNP NN))
Description : Prepositional phrase

State 3 : (PP IN (PP IN (NP NNP NNP CC NNP NN)))
Description : Prepositional phrase

State 4 : (NP DT JJ NN)
Description : Noun phrase

State 5 : (VP VBG (NP DT JJ NN) (PP IN (PP IN (NP NNP NNP CC NNP NN))))
Description : Verb phrase

State 6 : (VP AUX (VP VBG (NP DT JJ NN) (PP IN (PP IN (NP NNP NNP CC NNP NN)))))
Description : Verb phrase

State 7 : (NP NNP NNP)

Description : Noun phrase

State 8 : (PP IN (NP NNP NNP))
Description : Prepositional phrase

State 9 : (NP JJ NNS JJ)
Description : Noun phrase

State 10 : (S (NP JJ NNS JJ) (PP IN (NP NNP NNP)) : (VP AUX (VP VBG (NP DT JJ NN) (PP IN (PP IN (NP NNP NNP CC NNP NN))))) .)
Description : Simple declarative clause

State 11 : (SBAR , (S (NP JJ NNS JJ) (PP IN (NP NNP NNP)) : (VP AUX (VP VBG (NP DT JJ NN) (PP IN (PP IN (NP NNP NNP CC NNP NN))))) .))
Description : Clause introduced by sub-ordinating conjunction

State 12 : (ADJP JJ)
Description : Adjective phrase

State 13 : (NP DT NN)
Description : Noun phrase

State 14 : (PP IN (NP DT NN))
Description : Prepositional phrase

State 15 : (PP IN)
Description : Prepositional phrase

State 16 : (NP NN)
Description : Noun phrase

State 17 : (NP (NP NN) : (PP IN) (PP IN (NP DT NN)) (ADJP JJ) (SBAR , (S (NP JJ NNS JJ) (PP IN (NP NNP NNP)) : (VP AUX (VP VBG (NP DT JJ NN) (PP IN (PP IN (NP NNP NNP CC NNP NN))))) .)))
Description : Noun phrase

State 18 : (PP IN)

Description : Prepositional phrase

State 19 : (VP VBZ (PP IN) (NP (NP NN) : (PP IN) (PP IN (NP DT NN)) (ADJP JJ) (SBAR , (S (NP JJ NNS JJ) (PP IN (NP NNP NNP)) : (VP AUX (VP VBG (NP DT JJ NN) (PP IN (PP IN (NP NNP NNP CC NNP NN))))) .))))
Description : Verb phrase

State 20 : (ADVP RB)
Description : Adverb phrase

State 21 : (NP PRP)
Description : Noun phrase

Parse failed - could not create a valid parent for NP

Desired parse : (S1 (S (NP (NP NNP POS) NNP NNP NNS) (VP VBD (NP (NP CD JJ NN) (PRN -LRB- (NP $ CD -NONE-) -RRB-)) (PP TO (NP (NP CD) (PRN -LRB- (NP $ CD -NONE-) -RRB-))) (NP NN) , (SBAR IN (S (NP NNP NNS) (VP VBD (NP CD) (PP TO (NP CD)))))) .))

wsj_Concord-II's attempt ....

State 1 : (NP CD)
Description : Noun phrase

State 2 : (PP TO (NP CD))
Description : Prepositional phrase

State 3 : (NP CD)
Description : Noun phrase

State 4 : (VP VBD (NP CD) (PP TO (NP CD)))
Description : Verb phrase

State 5 : (NP NNP NNS)

Description : Noun phrase

State 6 : (S (NP NNP NNS) (VP VBD (NP CD) (PP TO (NP CD))))
Description : Simple declarative clause

State 7 : (SBAR IN (S (NP NNP NNS) (VP VBD (NP CD) (PP TO (NP CD)))))
Description : Clause introduced by sub-ordinating conjunction

State 8 : (NP $ CD -NONE- -RRB- NN , (SBAR IN (S (NP NNP NNS) (VP VBD (NP CD) (PP TO (NP CD))))))
Description : Noun phrase

State 9 : (NP CD)
Description : Noun phrase

State 10 : (PP TO (NP CD) -LRB-)
Description : Prepositional phrase

State 11 : (NP $ CD -NONE-)
Description : Noun phrase

State 12 : (NP CD JJ NN)
Description : Noun phrase

State 13 : (NP NNP NNP NNS)
Description : Noun phrase

State 14 : (NP POS (NP NNP NNP NNS))
Description : Noun phrase

Parse failed - could not create a valid parent for NP

Desired parse : (S1 (S (NP PRP) (VP VBP (NP (NP DT JJ NN) (PP IN (NP (NP DT JJ JJ NNS) , (VP (ADVP RB) VBN (NP -NONE-) (S (NP -NONE-) (VP TO (VP VB (PP IN (NP

(NP CD NN) (PP IN (NP CD)))) (PP TO (NP (NP CD NN) (PP IN (NP CD CC CD)))))))))))) .))

wsj_Concord-II's attempt ....

State 1 : (NP CD CC CD)
Description : Noun phrase

State 2 : (PP IN (NP CD CC CD))
Description : Prepositional phrase

State 3 : (ADJP TO CD NN)
Description : Adjective phrase

State 4 : (NP CD)
Description : Noun phrase

State 5 : (PP IN (NP CD))
Description : Prepositional phrase

State 6 : (NP CD NN)
Description : Noun phrase

State 7 : (VP VB IN)
Description : Verb phrase

State 8 : (VP TO (VP VB IN))
Description : Verb phrase

State 9 : (NP -NONE-)
Description : Noun phrase

State 10 : (S (NP -NONE-) (VP TO (VP VB IN)))
Description : Simple declarative clause

State 11 : (S -NONE- (S (NP -NONE-) (VP TO (VP VB IN))))
Description : Simple declarative clause

State 12 : (NP (NP CD NN) (PP IN (NP CD)))
Description : Noun phrase

NN NNS VBP PRP AUX VBG IN JJ JJ NN WDT -NONE- MD VB DT NN NN , CC JJS VBP -NONE- DT CD NNP NNS VBN -NONE- IN NN NN MD RB VB PRP$ NNS .

Parse failed - the tail of a phrase could not be found!

Desired parse : (S1 (S (S (NP NN NNS) (VP VBP (S (NP PRP) (VP AUX (VP VBG (PP IN (NP (NP JJ JJ NN) (SBAR (WHNP WDT) (S (NP -NONE-) (VP MD (VP VB (NP DT NN) (NP NN)))))))))) , CC (S (NP JJS) (VP VBP (SBAR -NONE- (S (NP (NP DT CD NNP NNS) (VP VBN (NP -NONE-) (PP IN (NP NN)) (NP NN))) (VP MD RB (VP VB (NP PRP$ NNS))))))) .))

wsj_Concord-II's attempt ....

State 1 : (NP PRP$ NNS)
Description : Noun phrase

State 2 : (VP VB (NP PRP$ NNS))
Description : Verb phrase

State 3 : (ADVP RB)
Description : Adverb phrase

State 4 : (VP MD (ADVP RB) (VP VB (NP PRP$ NNS)) .)
Description : Verb phrase

State 5 : (NP NN NN)
Description : Noun phrase

State 6 : (PP IN (NP NN NN))
Description : Prepositional phrase

State 7 : (NP -NONE-)
Description : Noun phrase

State 8 : (VP VBN (NP -NONE-))
Description : Verb phrase

State 9 : (NP DT CD NNP NNS)
Description : Noun phrase

State 10 : (FRAG (VP VBN (NP -NONE-)))
Description : Fragment

State 11 : (NP (NP DT CD NNP NNS) (FRAG (VP VBN (NP -NONE-))))
Description : Noun phrase

State 12 : (NP -NONE- (NP (NP DT CD NNP NNS) (FRAG (VP VBN (NP -NONE-)))) (PP IN (NP NN NN)))
Description : Noun phrase

State 13 : (VP VBP (NP -NONE- (NP (NP DT CD NNP NNS) (FRAG (VP VBN (NP -NONE-)))) (PP IN (NP NN NN))))
Description : Verb phrase

State 14 : (NP JJS)
Description : Noun phrase

State 15 : (S (NP JJS) (VP VBP (NP -NONE- (NP (NP DT CD NNP NNS) (FRAG (VP VBN (NP -NONE-)))) (PP IN (NP NN NN)))))
Description : Simple declarative clause

State 16 : (NP DT NN NN)
Description : Noun phrase

State 17 : (VP MD VB (NP DT NN NN))
Description : Verb phrase

State 18 : (VP (VP MD VB (NP DT NN NN)) , CC (S (NP JJS) (VP VBP (NP -NONE- (NP (NP DT CD NNP NNS) (FRAG (VP VBN (NP -NONE-)))) (PP IN (NP NN NN))))))

Description : Verb phrase

Parse failed - could not create a valid parent for FRAG

Desired parse : (S1 (S (SBAR (WHADVP WRB RB) (S (NP (NP DT NN POS) NN NN) (VP VBZ (NP DT NN) (ADVP -NONE-)))) (VP VBZ (PP IN (S (NP (NP NNS) (PP IN (NP DT JJ NN))) (VP VBG (ADVP RB))))) .))

wsj_Concord-II's attempt ....

State 1 : (ADVP RB)
Description : Adverb phrase

State 2 : (VP VBG)
Description : Verb phrase

State 3 : (NP DT JJ NN)
Description : Noun phrase

State 4 : (PP IN (NP DT JJ NN))
Description : Prepositional phrase

State 5 : (NP NNS)
Description : Noun phrase

State 6 : (NP (NP NNS) (PP IN (NP DT JJ NN)))
Description : Noun phrase

State 7 : (S (NP (NP NNS) (PP IN (NP DT JJ NN))) (VP VBG))
Description : Simple declarative clause

State 8 : (FRAG (S (NP (NP NNS) (PP IN (NP DT JJ NN))) (VP VBG)))
Description : Fragment

`` PRP AUX RB RB NN -NONE- PRP VBP -NONE- TO VB -NONE- RP CC VB . ''

Parse failed - the tail of a phrase could not be found!

Desired parse : (S1 (S `` (NP PRP) (VP AUX (ADVP RB) RB (NP (NP NN) (SBAR (WHNP -NONE-) (S (NP PRP) (VP VBP (S (NP -NONE-) (VP TO (VP (VP VB (NP -NONE-) (PRT RP)) CC (VP VB)))))))) . ''))

wsj_Concord-II's attempt ....

State 1 : (VP VB .)
Description : Verb phrase

State 2 : (PRT RP CC (VP VB .))
Description : Particle; category for words that should be tagged RP

State 3 : (NP -NONE-)
Description : Noun phrase

State 4 : (VP VB (NP -NONE-) (PRT RP CC (VP VB .)))
Description : Verb phrase

State 5 : (VP TO (VP VB (NP -NONE-) (PRT RP CC (VP VB .))))
Description : Verb phrase

State 6 : (NP -NONE-)
Description : Noun phrase

State 7 : (S (NP -NONE-) (VP TO (VP VB (NP -NONE-) (PRT RP CC (VP VB .)))))
Description : Simple declarative clause

State 8 : (VP VBP (S (NP -NONE-) (VP TO (VP VB (NP -NONE-) (PRT RP CC (VP VB .))))))
Description : Verb phrase

State 9 : (ADJP -NONE-)
Description : Adjective phrase

State 10 : (NP PRP)
Description : Noun phrase

State 11 : (ADVP RB RB)
Description : Adverb phrase

State 12 : (NP NN (ADJP -NONE-) (NP PRP))
Description : Noun phrase

State 13 : (S (NP NN (ADJP -NONE-) (NP PRP)) (VP VBP (S (NP -NONE-) (VP TO (VP VB (NP -NONE-) (PRT RP CC (VP VB .)))))))
Description : Simple declarative clause

State 14 : (VP AUX (ADVP RB RB) (S (NP NN (ADJP -NONE-) (NP PRP)) (VP VBP (S (NP -NONE-) (VP TO (VP VB (NP -NONE-) (PRT RP CC (VP VB .))))))))
Description : Verb phrase

State 15 : (NP PRP)
Description : Noun phrase

DT NNP POS JJ JJ NN AUX VBN -NONE- IN DT NN IN CD CD -NONE- -NONE- TO VB CD NN VBN IN DT NN IN CD CD CC NN IN CD NN NNP .
Parse failed - the tail of a phrase could not be found!
Desired parse : (S1 (S (NP (NP DT NNP POS) JJ JJ NN) (VP AUX (VP VBN (NP -NONE-) (PP IN (NP (NP DT NN) (PP IN (NP (NP (QP CD CD)) (SBAR (WHNP -NONE-) (S (NP -NONE-) (VP TO (VP VB (NP CD NN) (PP VBN (PP IN (NP (NP (NP (NP DT NN) (PP IN (NP (QP CD CD)))) CC (NP NN)) (PP IN (NP (NP CD NN) (NP NNP)))))))))))))))) .))

wsj_Concord-II's attempt ....

State 1 : (NP NNP)
Description : Noun phrase

State 2 : (NP CD NN)
Description : Noun phrase

State 3 : (NP (NP CD NN) (NP NNP))
Description : Noun phrase

State 4 : (PP IN (NP (NP CD NN) (NP NNP)))
Description : Prepositional phrase

State 5 : (QP CD CD CC NN)
Description : Adjective phrase (Quantitative)

State 6 : (NP (QP CD CD CC NN) (PP IN (NP (NP CD NN) (NP NNP))))
Description : Noun phrase

State 7 : (PP IN (NP (QP CD CD CC NN) (PP IN (NP (NP CD NN) (NP NNP)))))
Description : Prepositional phrase

State 8 : (NP DT NN)
Description : Noun phrase

State 9 : (NP (NP DT NN) (PP IN (NP (QP CD CD CC NN) (PP IN (NP (NP CD NN) (NP NNP))))))
Description : Noun phrase

State 10 : (PP IN (NP (NP DT NN) (PP IN (NP (QP CD CD CC NN) (PP IN (NP (NP CD NN) (NP NNP)))))))
Description : Prepositional phrase

State 11 : (PP VBN (PP IN (NP (NP DT NN) (PP IN (NP (QP CD CD CC NN) (PP IN (NP (NP CD NN) (NP NNP))))))))
Description : Prepositional phrase

State 12 : (NP CD NN)
Description : Noun phrase

State 13 : (VP VB (NP CD NN) (PP VBN (PP IN (NP (NP DT NN) (PP IN (NP (QP CD CD CC NN) (PP IN (NP (NP CD NN) (NP NNP)))))))))
Description : Verb phrase

State 14 : (VP TO (VP VB (NP CD NN) (PP VBN (PP IN (NP (NP DT NN) (PP IN (NP (QP CD CD CC NN) (PP IN (NP (NP CD NN) (NP NNP)))))))))
Description : Verb phrase

State 15 : (NP -NONE-)
Description : Noun phrase

State 16 : (S (NP -NONE-) (VP TO (VP VB (NP CD NN) (PP VBN (PP IN (NP (NP DT NN) (PP IN (NP (QP CD CD CC NN) (PP IN (NP (NP CD NN) (NP NNP)))))))))))
Description : Simple declarative clause

State 17 : (QP CD CD)
Description : Adjective phrase (Quantitative)

State 18 : (NP DT NN)
Description : Noun phrase

State 19 : (PP IN (NP DT NN))
Description : Prepositional phrase

State 20 : (NP -NONE-)
Description : Noun phrase

State 21 : (NP (NP -NONE-) (PP IN (NP DT NN)))
Description : Noun phrase

State 22 : (WHNP -NONE-)
Description : Wh-noun phrase

State 23 : (SBAR (WHNP -NONE-) (S (NP -NONE-) (VP TO (VP VB (NP CD NN) (PP VBN (PP IN (NP (NP DT NN) (PP IN (NP (QP CD CD CC NN) (PP IN (NP (NP CD NN) (NP NNP))))))))))))
Description : Clause introduced by sub-ordinating conjunction

State 24 : (NP (NP (NP -NONE-) (PP IN (NP DT NN))) IN)
Description : Noun phrase

State 25 : (VP VBN (NP (NP (NP -NONE-) (PP IN (NP DT NN))) IN))
Description : Verb phrase

Parse failed - could not create a valid parent for NP

Desired parse : (S1 (SBARQ `` (WHNP WP) (SQ (NP PRP) (VP AUX (S (NP -NONE-) (VP TO (VP AUX (S (NP -NONE-) (ADVP RB) (VP AUX (S (NP -NONE-) (VP VB (ADJP JJ (SBAR (SBAR IN (S (NP (NP DT JJ NN) (PP IN (NP NN NN CC NNS))) (VP AUX (VP VBD)))) CC (SBAR IN (S (NP PRP) (VP MD (VP VB (NP DT JJR NN)))))))))))))))) . ''))

wsj_Concord-II's attempt ....

State 1 : (NP DT JJR NN)
Description : Noun phrase

State 2 : (VP VB (NP DT JJR NN))
Description : Verb phrase

State 3 : (VP MD (VP VB (NP DT JJR NN)) .)
Description : Verb phrase

State 4 : (NP PRP)
Description : Noun phrase

State 5 : (PP IN (NP PRP))
Description : Prepositional phrase

State 6 : (VP VBD CC (PP IN (NP PRP)))
Description : Verb phrase

State 7 : (VP (VP VBD CC (PP IN (NP PRP))) (VP MD (VP VB (NP DT JJR NN)) .) '')
Description : Verb phrase

State 8 : (VP AUX)
Description : Verb phrase

State 9 : (NP NN CC NNS)
Description : Noun phrase

State 10 : (S (NP NN CC NNS) (VP AUX))
Description : Simple declarative clause

State 11 : (NP NN)
Description : Noun phrase

State 12 : (NP (NP NN) (S (NP NN CC NNS) (VP AUX)))
Description : Noun phrase

State 13 : (PP IN (NP (NP NN) (S (NP NN CC NNS) (VP AUX))))
Description : Prepositional phrase

State 14 : (NP DT JJ NN)
Description : Noun phrase

State 15 : (NP (NP DT JJ NN) (PP IN (NP (NP NN) (S (NP NN CC NNS) (VP AUX)))))
Description : Noun phrase

Parse failed - could not create a valid parent for S1

Desired parse : (S1 (S `` (NP (NP PRP) (PP IN (NP DT NNS))) (VP MD (VP VB (SBAR -NONE- (S -LRB- (NP WDT) -RRB- (NP (NP DT NNS) (PP IN (NP NN CC NN))) (VP MD (VP AUX (VP VBN (NP -NONE-) (PP IN (NP NN CC NN)) , (PP RB IN (NP (NP DT JJ NN) (PP IN (NP NNP))))))))))) . ''))

wsj_Concord-II's attempt ....

State 1 : (NP NNP)
Description : Noun phrase

State 2 : (PP IN (NP NNP))
Description : Prepositional phrase

State 3 : (NP DT JJ NN)
Description : Noun phrase

State 4 : (NP (NP DT JJ NN) (PP IN (NP NNP)))
Description : Noun phrase

State 5 : (PP RB IN (NP (NP DT JJ NN) (PP IN (NP NNP))))
Description : Prepositional phrase

State 6 : (NP NN CC NN)
Description : Noun phrase

State 7 : (PP IN (NP NN CC NN))
Description : Prepositional phrase

State 8 : (NP -NONE-)
Description : Noun phrase

State 9 : (VP VBN (NP -NONE-) (PP IN (NP NN CC NN)) , (PP RB IN (NP (NP DT JJ NN) (PP IN (NP NNP)))))
Description : Verb phrase

State 10 : (VP AUX (VP VBN (NP -NONE-) (PP IN (NP NN CC NN)) , (PP RB IN (NP (NP DT JJ NN) (PP IN (NP NNP))))))

Description : Verb phrase

State 11 : (VP MD (VP AUX (VP VBN (NP -NONE-) (PP IN (NP NN CC NN)) , (PP RB IN (NP (NP DT JJ NN) (PP IN (NP NNP)))))))
Description : Verb phrase

State 12 : (NP NN CC NN)
Description : Noun phrase

State 13 : (PP IN (NP NN CC NN))
Description : Prepositional phrase

State 14 : (NP DT NNS)
Description : Noun phrase

State 15 : (NP (NP DT NNS) (PP IN (NP NN CC NN)))
Description : Noun phrase

State 16 : (NP WDT -RRB-)
Description : Noun phrase

State 17 : (S (NP WDT -RRB-) (NP (NP DT NNS) (PP IN (NP NN CC NN))) (VP MD (VP AUX (VP VBN (NP -NONE-) (PP IN (NP NN CC NN)) , (PP RB IN (NP (NP DT JJ NN) (PP IN (NP NNP)))))))
Description : Simple declarative clause

State 18 : (VP MD VB)
Description : Verb phrase

State 19 : (NP DT NNS)
Description : Noun phrase

State 20 : (PP IN (NP DT NNS))
Description : Prepositional phrase

State 21 : (S1 (S (NP WDT -RRB-) (NP (NP DT NNS) (PP IN (NP NN CC NN))) (VP MD (VP AUX (VP VBN (NP -NONE-) (PP IN (NP NN CC NN)) , (PP RB IN (NP (NP DT JJ NN) (PP IN (NP NNP)))))))) .)
Description : Root node

Parse failed - could not create a valid parent for NP

Desired parse : (S1 (S (NP DT NNP NN) (VP VBD (SBAR -NONE- (S (NP NNS) (VP VBD (PP IN (NP DT (ADJP (QP $ CD CD) -NONE-) JJ NN)) , (ADVP RB (PP IN (NP DT (ADJP (QP $ CD CD) -NONE-) NN) (PP IN (NP DT JJ NN)))))))) .))

wsj_Concord-II's attempt ....

State 1 : (NP DT JJ NN)
Description : Noun phrase

State 2 : (PP IN (NP DT JJ NN))
Description : Prepositional phrase

State 3 : (QP $ CD CD)
Description : Adjective phrase (Quantitative)

State 4 : (NP DT (QP $ CD CD) -NONE- NN)
Description : Noun phrase

State 5 : (PP IN (NP DT (QP $ CD CD) -NONE- NN) (PP IN (NP DT JJ NN)))
Description : Prepositional phrase

State 6 : (ADVP RB)
Description : Adverb phrase

State 7 : (FRAG (ADVP RB) (PP IN (NP DT (QP $ CD CD) -NONE- NN) (PP IN (NP DT JJ NN))))
Description : Fragment

State 8 : (NP CD CD -NONE- JJ NN)
Description : Noun phrase

State 9 : (S (NP CD CD -NONE- JJ NN) , (FRAG (ADVP RB) (PP IN (NP DT (QP $ CD CD) -NONE- NN) (PP IN (NP DT JJ NN)))) .)
Description : Simple declarative clause

State 10 : (NP DT)
Description : Noun phrase

State 11 : (PP IN (NP DT) $ (S (NP CD CD -NONE- JJ NN) , (FRAG (ADVP RB) (PP IN (NP DT (QP $ CD CD) -NONE- NN) (PP IN (NP DT JJ NN)))) .))
Description : Prepositional phrase

State 12 : (VP VBD (PP IN (NP DT) $ (S (NP CD CD -NONE- JJ NN) , (FRAG (ADVP RB) (PP IN (NP DT (QP $ CD CD) -NONE- NN) (PP IN (NP DT JJ NN)))) .)))
Description : Verb phrase

State 13 : (NP -NONE- NNS)
Description : Noun phrase

Appendix F: A Sample of Parse Failures made on the WSJ Corpus Training Set (Using Lexical Semantic and Syntactic Information)

Parse failed - could not create a valid parent for NP_0000000000000000000A000A~NN

Desired parse : (S1 (S (NP (NP (NP NNP NNPS POS) NNP NN NN) (PP IN (NP NNP NNP))) (VP VBD (PP IN (NP CD NN)) (PP IN (ADVP (NP DT NN) RBR)) , (PP VBG (PP TO (NP (NP DT NNP) (PP IN (NP NNP NNP)))))) .))

wsj_Concord-II's attempt ....

State 1 : (NP_0000000000A00000000000000~NNP NNP_0000000000A00000000000000 NNP_0000000000A00000000000000)
Description : Noun phrase

State 2 : (PP_0000000000A00000000000000 IN (NP_0000000000A00000000000000~NNP NNP_0000000000A00000000000000 NNP_0000000000A00000000000000))
Description : Prepositional phrase

State 3 : (NP_0000000000A00000000000000~NNP DT NNP_0000000000A00000000000000)
Description : Noun phrase

State 4 : (PP_0000000000A00000000000000 TO (NP_0000000000A00000000000000~NNP DT NNP_0000000000A00000000000000))
Description : Prepositional phrase

State 5 : (NP_0000000000B00000000000000~-NONE- (PP_0000000000A00000000000000 TO (NP_0000000000A00000000000000~NNP DT NNP_0000000000A00000000000000)))
Description : Noun phrase

State 6 : (VP VBG (NP_0000000000B00000000000000~-NONE- (PP_0000000000A00000000000000 TO (NP_0000000000A00000000000000~NNP DT NNP_0000000000A00000000000000))) (PP_0000000000A00000000000000 IN (NP_0000000000A00000000000000~NNP NNP_0000000000A00000000000000 NNP_0000000000A00000000000000)))
Description : Verb phrase

State 7 : (ADVP RBR)
Description : Adverb phrase

State 8 : (NP_000000000000000000000000A~NN DT NN_0000000000D0000000000000A)
Description : Noun phrase

State 9 : (PP_000000000000000000000000A IN (NP_000000000000000000000000A~NN DT NN_0000000000D0000000000000A))
Description : Prepositional phrase

State 10 : (NP_000000000000000000000A000A~NN CD NN_000000000000000000000A0000 (PP_000000000000000000000000A IN (NP_000000000000000000000000A~NN DT NN_0000000000D0000000000000A)))
Description : Noun phrase

State 11 : (NP_000000000000000000000A000A~NN (NP_000000000000000000000A000A~NN CD NN_000000000000000000000A0000 (PP_000000000000000000000000A IN (NP_000000000000000000000000A~NN DT NN_0000000000D0000000000000A))) (ADVP RBR))
Description : Noun phrase

State 12 : (NP_000000000000000000000B000A~NN (NP_000000000000000000000A000A~NN (NP_000000000000000000000A000A~NN CD NN_000000000000000000000A0000 (PP_000000000000000000000000A IN (NP_000000000000000000000000A~NN DT NN_0000000000D0000000000000A))) (ADVP RBR)) , (VP VBG (NP_0000000000B00000000000000~-NONE- (PP_0000000000A00000000000000 TO (NP_0000000000A00000000000000~NNP DT NNP_0000000000A00000000000000))) (PP_0000000000A00000000000000 IN

(NP_0000000000A00000000000000~NNP NNP_0000000000A00000000000000 NNP_0000000000A00000000000000))))
Description : Noun phrase

State 13 : (NP_0000000000000000000A000A~NN (NP_0000000000000000000B000A~NN (NP_0000000000000000000A000A~NN (NP_0000000000000000000A000A~NN CD NN_0000000000000000000A0000 (PP_0000000000000000000000000A IN (NP_0000000000000000000000000A~NN DT NN_0000000000D0000000000000A))) (ADVP RBR)) , (VP VBG (NP_0000000000B00000000000000~-NONE- (PP_0000000000A00000000000000 TO (NP_0000000000A00000000000000~NNP DT NNP_0000000000A00000000000000))) (PP_0000000000A00000000000000 IN (NP_0000000000A00000000000000~NNP NNP_0000000000A00000000000000 NNP_0000000000A00000000000000)))))
Description : Noun phrase

Parse failed - could not create a valid parent for NP_0000000000000000000A000A~NN

Desired parse : (S1 (S (NP DT JJ NN) (VP AUX (ADJP JJ) (PP IN (NP (NP CD NN) (PP IN (NP DT JJ CD NNS)) (PP IN (NP NNP))))) .))

wsj_Concord-II's attempt ....

State 1 : (NP_000000000000000000000000A~NNP NNP_000000000000000000000000A)
Description : Noun phrase

State 2 : (PP_000000000000000000000000A IN (NP_000000000000000000000000A~NNP NNP_000000000000000000000000A))
Description : Prepositional phrase

State 3 : (NP_000000000000000000000000A~NNS DT JJ CD NNS_000000000000000000000000A)
Description : Noun phrase

State 4 : (PP_0000000000000000000000A IN (NP_0000000000000000000000A~NNS DT JJ CD NNS_0000000000000000000000A))
Description : Prepositional phrase

State 5 : (NP_00000000000000000000A0000~NN CD NN_00000000000000000000A0000)
Description : Noun phrase

State 6 : (NP_00000000000000000000A000A~NN (NP_00000000000000000000A0000~NN CD NN_00000000000000000000A0000) (PP_0000000000000000000000A IN (NP_0000000000000000000000A~NNS DT JJ CD NNS_0000000000000000000000A)) (PP_0000000000000000000000A IN (NP_0000000000000000000000A~NNP NNP_0000000000000000000000A)))
Description : Noun phrase

IN NNP_00000000000A0000000000000 NNP_00000000000A0000000000000 , NNP_00000000000A0000000000000 , DT NN_0000000000000A0000000000 IN PRP_0000000000A00000000000000 NNP_0000000000A00000000000000 NNPS_0000000000A00000000000000 NNP_0000000000A00000000000000 , DT NN_00000D0000C00000000AB0000 IN NNP_0000000000A00000000000000 NNPS_0000000000A00000000000000 NNP_0000000000A00000000000000 VBD -NONE- DT JJ NN_000000A000000000000000000 NN_00A00000000000A00000000D0 MD RB VB DT NN_A000000000000000000000000 .
Parse failed - the tail of a phrase could not be found!
Desired parse : (S1 (S (PP IN (NP (NP NNP NNP) , (NP NNP))) , (NP (NP DT NN) (PP IN (NP PRP))) (NP (NP NNP NNPS NNP) , (NP (NP DT NN) (PP IN (NP NNP NNPS NNP)))) (VP VBD (SBAR -NONE- (S (NP DT JJ NN NN) (VP MD (ADVP RB) (VP VB (NP DT NN)))))) .))

wsj_Concord-II's attempt ....

State 1 : (NP_A000000000000000000000000~NN DT NN_A000000000000000000000000)
Description : Noun phrase

State 2 : (VP VB (NP_A000000000000000000000000~NN DT NN_A000000000000000000000000))
Description : Verb phrase

State 3 : (ADVP RB)
Description : Adverb phrase

State 4 : (VP (ADVP RB) (VP VB (NP_A000000000000000000000000~NN DT NN_A000000000000000000000000)))
Description : Verb phrase

State 5 : (VP MD (VP (ADVP RB) (VP VB (NP_A000000000000000000000000~NN DT NN_A000000000000000000000000))))
Description : Verb phrase

State 6 : (NP_00A000A0000000A00000000D0~NN DT JJ NN_000000A000000000000000000 NN_00A00000000000A00000000D0)
Description : Noun phrase

State 7 : (S (NP_00A000A0000000A00000000D0~NN DT JJ NN_000000A000000000000000000 NN_00A00000000000A00000000D0) (VP MD (VP (ADVP RB) (VP VB (NP_A000000000000000000000000~NN DT NN_A000000000000000000000000)))))
Description : Simple declarative clause

State 8 : (NP~-NONE- -NONE-)
Description : Noun phrase

State 9 : (SBAR (NP~-NONE- -NONE-) (S (NP_00A000A0000000A00000000D0~NN DT JJ NN_000000A000000000000000000 NN_00A00000000000A00000000D0) (VP MD (VP (ADVP RB) (VP VB

(NP_A0000000000000000000000~NN DT NN_A0000000000000000000000))))))
Description : Clause introduced by sub-ordinating conjunction

State 10 : (VP VBD (SBAR (NP~-NONE- -NONE-) (S (NP_00A000A0000000A00000000D0~NN DT JJ NN_000000A000000000000000000 NN_00A00000000000A00000000D0) (VP MD (VP (ADVP RB) (VP VB (NP_A0000000000000000000000~NN DT NN_A0000000000000000000000)))))))
Description : Verb phrase

State 11 : (NP_0000000000A00000000000000~NNP NNP_0000000000A00000000000000)
Description : Noun phrase

State 12 : (NP_0000000000A00000000000000~NN NNP_0000000000A00000000000000)
Description : Noun phrase

State 13 : (PP_0000000000A00000000000000 IN (NP_0000000000A00000000000000~NN NNP_0000000000A00000000000000))
Description : Prepositional phrase

State 14 : (NP_00000000000000000000C00000~NN NN_00000D0000C00000000AB0000)
Description : Noun phrase

State 15 : (ADJP (NP_00000000000000000000C00000~NN NN_00000D0000C00000000AB0000) (PP_0000000000A00000000000000 IN (NP_0000000000A00000000000000~NN NNP_0000000000A00000000000000)))
Description : Adjective phrase

State 16 : (NP_0000000000A00000000000000~PRP-PLE NNPS_0000000000A00000000000000 (NP_0000000000A00000000000000~NNP NNP_0000000000A00000000000000))
Description : Noun phrase

State 17 : (NP~NN DT)
Description : Noun phrase

State 18 : (NP_0000000000A00000000000000~NN (NP~NN DT) (ADJP (NP_000000000000000000C00000~NN NN_00000D0000C00000000AB0000) (PP_0000000000A00000000000000 IN (NP_0000000000A00000000000000~NN NNP_0000000000A00000000000000))) (NP_0000000000A00000000000000~PRP-PLE NNPS_0000000000A00000000000000 (NP_0000000000A00000000000000~NNP NNP_0000000000A00000000000000)))
Description : Noun phrase

State 19 : (NP_0000000000A00000000000000~NNP NNP_0000000000A00000000000000 NNPS_0000000000A00000000000000 NNP_0000000000A00000000000000)
Description : Noun phrase

State 20 : (NP_0000000000A00000000000000~PRP PRP_0000000000A00000000000000 (NP_0000000000A00000000000000~NNP NNP_0000000000A00000000000000 NNPS_0000000000A00000000000000 NNP_0000000000A00000000000000))
Description : Noun phrase

State 21 : (PP_0000000000A00000000000000 IN (NP_0000000000A00000000000000~PRP PRP_0000000000A00000000000000 (NP_0000000000A00000000000000~NNP NNP_0000000000A00000000000000 NNPS_0000000000A00000000000000 NNP_0000000000A00000000000000)))
Description : Prepositional phrase

State 22 : (NP_00000000000000A0000000000~NN DT NN_00000000000000A0000000000)
Description : Noun phrase

State 23 : (NP_0000000000A000A0000000000~NN (NP_00000000000000A0000000000~NN DT NN_00000000000000A0000000000) (PP_0000000000A00000000000000 IN (NP_0000000000A00000000000000~PRP PRP_0000000000A00000000000000 (NP_0000000000A00000000000000~NNP

NNP_0000000000A00000000000000 NNPS_0000000000A00000000000000 NNP_0000000000A00000000000000))))
Description : Noun phrase

State 24 : (NP_00000000000A0000000000000~NNP NNP_00000000000A0000000000000)
Description : Noun phrase

State 25 : (NP_00000000000A0000000000000~NNP NNP_00000000000A0000000000000)
Description : Noun phrase

State 26 : (NP_00000000000A0000000000000~NNP (NP_00000000000A0000000000000~NNP NNP_00000000000A0000000000000) , (NP_00000000000A0000000000000~NNP NNP_00000000000A0000000000000))
Description : Noun phrase

State 27 : (NP_00000000000A0000000000000~NN (NP_00000000000A0000000000000~NNP (NP_00000000000A0000000000000~NNP NNP_00000000000A0000000000000) , (NP_00000000000A0000000000000~NNP NNP_00000000000A0000000000000)) ,)
Description : Noun phrase

State 28 : (NP_00000000000A0000000000000~NNP NNP_00000000000A0000000000000)
Description : Noun phrase

State 29 : (UCP (NP_00000000000A0000000000000~NNP NNP_00000000000A0000000000000) (NP_00000000000A0000000000000~NN (NP_00000000000A0000000000000~NNP (NP_00000000000A0000000000000~NNP NNP_00000000000A0000000000000) , (NP_00000000000A0000000000000~NNP NNP_00000000000A0000000000000)) ,))
Description : Unlike coordinated phrase

DT JJ NN_000A0B0000000000000000000 AUX WRB RB PRP_0000000A00000000000000000 MD AUX DT NN_000B000000000000A000000D00 -NONE- . ''

Parse failed - the tail of a phrase could not be found!

Desired parse : (S1 (S (NP DT JJ NN) (VP AUX (SBAR (WHADVP WRB RB) (S (NP PRP) (VP MD (VP AUX (NP DT NN) (ADVP -NONE-)))))) . ''))

wsj_Concord-II's attempt ....

State 1 : (ADVP -NONE-)
Description : Adverb phrase

State 2 : (NP_000B000000000000A000000D00~NN DT NN_000B000000000000A000000D00)
Description : Noun phrase

State 3 : (VP AUX (NP_000B000000000000A000000D00~NN DT NN_000B000000000000A000000D00) (ADVP -NONE-))
Description : Verb phrase

State 4 : (VP MD (VP AUX (NP_000B000000000000A000000D00~NN DT NN_000B000000000000A000000D00) (ADVP -NONE-)))
Description : Verb phrase

State 5 : (NP_0000000A00000000000000000~PRP PRP_0000000A00000000000000000)
Description : Noun phrase

State 6 : (S (NP_0000000A00000000000000000~PRP PRP_0000000A00000000000000000) (VP MD (VP AUX (NP_000B000000000000A000000D00~NN DT NN_000B000000000000A000000D00) (ADVP -NONE-))))
Description : Simple declarative clause

State 7 : (PRN RB (S (NP_0000000A00000000000000000~PRP PRP_0000000A00000000000000000) (VP MD (VP AUX

(NP_000B00000000000A000000D00~NN DT NN_000B00000000000A000000D00) (ADVP -NONE-)))))
Description : Parenthetical

State 8 : (VP AUX WRB (PRN RB (S (NP_0000000A00000000000000000~PRP PRP_0000000A00000000000000000) (VP MD (VP AUX (NP_000B00000000000A000000D00~NN DT NN_000B00000000000A000000D00) (ADVP -NONE-))))))
Description : Verb phrase

Parse failed - could not create a valid parent for NP_00000000000A0000000000000~NN

Desired parse : (S1 (S (NP (NP DT JJ NN) (PP IN (NP NNP))) (VP AUX (NP (NP CD NN) (PP IN (NP DT NNP NNP NNP NN)) , (SBAR (WHNP WDT) (S (NP -NONE-) (VP VBZ (NP (NP (NP NNP NNP) , (NP NNP NNP) CC (NP (NP NNP) , (NP NNP) , CC (NP NNP))) CC (NP (NP NNP NNP) , (NP NNP)))))))) .))

wsj_Concord-II's attempt ....

State 1 : (NP_00000000000A0000000000000~NNP NNP_00000000000A0000000000000)
Description : Noun phrase

State 2 : (NP_00000000000A0000000000000~NNP NNP_00000000000A0000000000000 NNP_00000000000A0000000000000)
Description : Noun phrase

State 3 : (NP_00000000000A0000000000000~NNP NNP_00000000000A0000000000000)
Description : Noun phrase

State 4 : (NP_00000000000A0000000000000~NNP NNP_00000000000A0000000000000)
Description : Noun phrase

State 5 : (NP_00000000000A0000000000000~NN (NP_00000000000A0000000000000~NNP NNP_00000000000A0000000000000) , CC (NP_00000000000A0000000000000~NNP NNP_00000000000A0000000000000) CC (NP_00000000000A0000000000000~NNP NNP_00000000000A0000000000000 NNP_00000000000A0000000000000))
Description : Noun phrase

State 6 : (NP_00000000000A0000000000000~NN CC NNP_00000000000A0000000000000)
Description : Noun phrase

State 7 : (NP_00000000000A00D0000000000~NNP NNP_00000000000A0000000000000 NNP_00000000000A0000000000000 (NP_00000000000A0000000000000~NN CC NNP_00000000000A0000000000000) , (NP_00000000000A0000000000000~NN (NP_00000000000A0000000000000~NNP NNP_00000000000A0000000000000) , CC (NP_00000000000A0000000000000~NNP NNP_00000000000A0000000000000) CC (NP_00000000000A0000000000000~NNP NNP_00000000000A0000000000000 NNP_00000000000A0000000000000)))
Description : Noun phrase

State 8 : (NP_00000000000A0000000000000~NNP NNP_00000000000A0000000000000 NNP_00000000000A0000000000000)
Description : Noun phrase

State 9 : (VP VBZ (NP_00000000000A0000000000000~NNP NNP_00000000000A0000000000000 NNP_00000000000A0000000000000) ,)
Description : Verb phrase

State 10 : (WHNP WDT)
Description : Wh-noun phrase

State 11 : (NP~-NONE- (WHNP WDT) -NONE-)
Description : Noun phrase

State 12 : (NP_00000000000A0000000000000~IN NNP_00000000000A0000000000000)
Description : Noun phrase

State 13 : (NP_00000000000A0000000000000~NN NN_0000B000000A0000000D00000)
Description : Noun phrase

Parse failed - could not create a valid parent for UCP

Desired parse : (S1 (S (NP NNP NNP NNP NNP NNP) (VP VBD (SBAR -NONE- (S (NP NN NN) (VP VBD (NP CD NN) (PP TO (NP (NP (QP CD CD) NNS) (PRN -LRB- (NP (QP $ CD CD) -NONE-) -RRB-))) (PP IN (NP DT JJ NN)) (PP IN (NP (QP CD CD) NNS) (ADVP (NP DT NN) RBR)))))) .))

wsj_Concord-II's attempt ....

State 1 : (ADVP RBR)
Description : Adverb phrase

State 2 : (NP_000000000000000000000000A~NN DT NN_0000000000D0000000000000A)
Description : Noun phrase

State 3 : (QP CD CD)
Description : Adjective phrase (Quantitative)

State 4 : (PP IN (QP CD CD))
Description : Prepositional phrase

State 5 : (NP_000000000000000000B0000D~NN JJ NN_00000000000000000000A0000B (PP IN (QP CD CD)) NNS_00000000000000000000A00000)
Description : Noun phrase

State 6 : (NP_0000000000000000000B00000~NN DT (NP_0000000000000000000B0000D~NN JJ NN_0000000000000000000A0000B (PP IN (QP CD CD)) NNS_0000000000000000000A00000))
Description : Noun phrase

State 7 : (PP_0000000000000000000B00000 IN (NP_0000000000000000000B00000~NN DT (NP_0000000000000000000B0000D~NN JJ NN_0000000000000000000A0000B (PP IN (QP CD CD)) NNS_0000000000000000000A00000)))
Description : Prepositional phrase

State 8 : (UCP (PP_0000000000000000000B00000 IN (NP_0000000000000000000B00000~NN DT (NP_0000000000000000000B0000D~NN JJ NN_0000000000000000000A0000B (PP IN (QP CD CD)) NNS_0000000000000000000A00000))) (NP_0000000000000000000000000A~NN DT NN_0000000000D0000000000000A))
Description : Unlike coordinated phrase

State 9 : (NP~CD CD CD)
Description : Noun phrase

State 10 : (PP $ (NP~CD CD CD))
Description : Prepositional phrase

State 11 : (UCP (PP $ (NP~CD CD CD)))
Description : Unlike coordinated phrase

State 12 : (NP~-NONE- (UCP (PP $ (NP~CD CD CD))) -NONE-)
Description : Noun phrase

State 13 : (UCP -LRB- (NP~-NONE- (UCP (PP $ (NP~CD CD CD))) -NONE-) -RRB-)
Description : Unlike coordinated phrase

State 14 : (PRN (UCP -LRB- (NP~-NONE- (UCP (PP $ (NP~CD CD CD))) -NONE-) -RRB-) (ADVP RBR))

Description : Parenthetical

DT NNP_00000000000000A0000000000 NN_A000000000B00000000000000D VBD NNP_00000000000000A0000000000 NNPS_00000000000000A0000000000 IN DT NNP_00A0000000B00000000000000 NNP_00A0000000B00000000000000 IN NNP_00000000000A0000000000000 NNP_00000000000A0000000000000 MD RB VB NNS_B000000A00000000000000000 IN JJ NNS_00000000000000A0000000000 VBD NNP_0000000000000000000000A00 NNP_0000000000000000000000A00

Parse failed - the tail of a phrase could not be found!

Desired parse : (S1 (S (NP DT NNP NN) (VP VBD (NP NNP NNPS) (SBAR IN (S (NP (NP DT NNP NNP) (PP IN (NP NNP NNP))) (VP MD RB (VP VB (NP (NP NNS) (PP IN (NP (NP JJ NNS) (VP VBD (NP NNP NNP))))))))))))

wsj_Concord-II's attempt ....

State 1 : (ADJP NNP_0000000000000000000000A00)
Description : Adjective phrase

State 2 : (NP~NNP NNP_0000000000000000000000A00)
Description : Noun phrase

State 3 : (INTJ (ADJP NNP_0000000000000000000000A00) (NP~NNP NNP_0000000000000000000000A00))
Description : Interjection - corresponds approximately to the word tag 'UH'

State 4 : (VP VBD)
Description : Verb phrase

State 5 : (NP_00000000000000A0000000000~NNS NNS_00000000000000A0000000000)
Description : Noun phrase

State 6 : (RRC (NP_00000000000000A0000000000~NNS NNS_00000000000000A0000000000) (VP VBD))
Description : Reduced relative clause

Parse failed - could not create a valid parent for NP_000000A000000000000000000~NNS

Desired parse : (S1 (S (NP DT NNS) (VP VBP (SBAR IN (S (NP NNP CC NNP) (ADVP RB) (VP VBZ (NP DT JJ NN) (PP TO (NP DT NN)) (SBAR RB IN (S (NP (NP DT NN) (NP PRP)) (VP VBZ (NP PRP) (ADVP RB RB) (PP IN (NP PRP$ NNS)))))))) .))

wsj_Concord-II's attempt ....

State 1 : (NP_D00000A0000000000C0000000~NNS PRP$ NNS_D00000A0000000000C0000000)
Description : Noun phrase

State 2 : (NP_000000A000000000000000000~NNS (NP_D00000A0000000000C0000000~NNS PRP$ NNS_D00000A0000000000C0000000))
Description : Noun phrase

Parse failed - could not create a valid parent for NP_000000000000000000000000A~NN

Desired parse : (S1 (S (NP (NP NNP NNP NNP) , (NP (NP NN) (PP IN (NP NNP NNP))) ,) (ADVP RB) (VP VBD (SBAR -NONE- (S (NP NNS) (VP MD (VP VB (PP TO (NP (QP RB CD CD) NNS)) (NP JJ NN)))))) .))

wsj_Concord-II's attempt ....

State 1 : (QP CD CD NNS_00A000B000000C000000000C0 JJ)
Description : Adjective phrase (Quantitative)

State 2 : (NP_000000000000000000000000A~NN (QP CD CD NNS_00A000B000000C000000000C0 JJ) NN_0000000000D0000000000000A)

Description : Noun phrase

DT NN_00D0B0A000000000000C00000 AUX VBN IN $ CD CD -NONE- IN DT CD NNS_0000000000000000000000000A IN NNP_0000000000000000000000000A CD , RB IN NNS_A00000000000000C0B0000000 IN NNS_00000000A0000000000B00000 CC NNS_00000BC0000000A0000000000 .
Parse failed - the tail of a phrase could not be found!
Desired parse : (S1 (S (NP DT NN) (VP AUX (VP VBN (PP IN (NP (QP $ CD CD) -NONE-)) (PP IN (NP DT CD NNS)) (PP IN (NP NNP CD)) , (PP (ADVP RB) IN (NP (NP NNS) (PP IN (NP (NP NNS) CC (NP NNS))))))) .))

wsj_Concord-II's attempt ....

State 1 : (NP_00000BC0000000A0000000000~NNS NNS_00000BC0000000A0000000000)
Description : Noun phrase

State 2 : (NP_00000000A0000000000B00000~NNS NNS_00000000A0000000000B00000)
Description : Noun phrase

State 3 : (NP_00000BC0A00000A0000B00000~NNS (NP_00000000A0000000000B00000~NNS NNS_00000000A0000000000B00000) CC (NP_00000BC0000000A0000000000~NNS NNS_00000BC0000000A0000000000))
Description : Noun phrase

State 4 : (PP_00000BC0A00000A0000B00000 IN (NP_00000BC0A00000A0000B00000~NNS (NP_00000000A0000000000B00000~NNS NNS_00000000A0000000000B00000) CC (NP_00000BC0000000A0000000000~NNS NNS_00000BC0000000A0000000000)))
Description : Prepositional phrase

State 5 : (NP_A00000000000000C0B0000000~NNS NNS_A00000000000000C0B0000000)
Description : Noun phrase

State 6 : (PP_A0000000000000000D000000D IN (NP_A00000000000000C0B0000000~NNS NNS_A00000000000000C0B0000000))
Description : Prepositional phrase

State 7 : (ADVP RB)
Description : Adverb phrase

State 8 : (NP_0000000000000000000000000A~CD NNP_0000000000000000000000000A CD)
Description : Noun phrase

State 9 : (NP_B0000000000000000000000000A~DT (NP_0000000000000000000000000A~CD NNP_0000000000000000000000000A CD) , (ADVP RB) (PP_A0000000000000000D000000D IN (NP_A00000000000000C0B0000000~NNS NNS_A00000000000000C0B0000000)))
Description : Noun phrase

State 10 : (NP_AC000000A00000D0000A0000A~JJS (NP_B0000000000000000000000000A~DT (NP_0000000000000000000000000A~CD NNP_0000000000000000000000000A CD) , (ADVP RB) (PP_A0000000000000000D000000D IN (NP_A00000000000000C0B0000000~NNS NNS_A00000000000000C0B0000000))) (PP_00000BC0A00000A0000B00000 IN (NP_00000BC0A00000A0000B00000~NNS (NP_00000000A0000000000B00000~NNS NNS_00000000A0000000000B00000) CC (NP_00000BC0000000A0000000000~NNS NNS_00000BC0000000A0000000000))))
Description : Noun phrase

State 11 : (PP_B0000000A0000000000A0000A IN (NP_AC000000A00000D0000A0000A~JJS (NP_B0000000000000000000000000A~DT (NP_0000000000000000000000000A~CD NNP_0000000000000000000000000A CD) , (ADVP RB) (PP_A0000000000000000D000000D IN (NP_A00000000000000C0B0000000~NNS NNS_A00000000000000C0B0000000)))

(PP_00000BC0A00000A0000B00000 IN (NP_00000BC0A00000A0000B00000~NNS (NP_00000000A0000000000B00000~NNS NNS_00000000A0000000000B00000) CC (NP_00000BC0000000A0000000000~NNS NNS_00000BC0000000A0000000000)))))
Description : Prepositional phrase

State 12 : (NP_0000000000000000000000000A~NNS DT CD NNS_0000000000000000000000000A)
Description : Noun phrase

Parse failed - could not create a valid parent for NP_A0000000000000000000000000~NN

Desired parse : (S1 (S (NP NNP) (VP VBD (SBAR IN (S (NP (NP NN) , (SBAR (WHNP WDT) (S (NP -NONE-) (VP AUX (PP IN (NP (NP DT JJ NNS) (PP IN (NP NN))))))) ,) (VP AUX (ADVP RB) (VP AUXG (VP VBN (PP IN (NP DT JJ NN)))))))) .))

wsj_Concord-II's attempt ....

State 1 : (NP_00000B000000000000000A0000~NN DT JJ NN_00000B000000000000000A0000)
Description : Noun phrase

State 2 : (NP_00000D000000000000000A0000~NN (NP_00000B000000000000000A0000~NN DT JJ NN_00000B000000000000000A0000))
Description : Noun phrase

State 3 : (PP_0000000000000000000A0000 IN (NP_00000D000000000000000A0000~NN (NP_00000B000000000000000A0000~NN DT JJ NN_00000B000000000000000A0000)))
Description : Prepositional phrase

State 4 : (VP VBN (PP_0000000000000000000A0000 IN (NP_00000D00000000000000A0000~NN (NP_00000B00000000000000A0000~NN DT JJ NN_00000B00000000000000A0000))))
Description : Verb phrase

State 5 : (ADVP AUXG)
Description : Adverb phrase

State 6 : (VP (ADVP AUXG) (VP VBN (PP_0000000000000000000A0000 IN (NP_00000D00000000000000A0000~NN (NP_00000B00000000000000A0000~NN DT JJ NN_00000B00000000000000A0000)))))
Description : Verb phrase

State 7 : (ADVP RB)
Description : Adverb phrase

State 8 : (VP AUX (ADVP RB) (VP (ADVP AUXG) (VP VBN (PP_0000000000000000000A0000 IN (NP_00000D00000000000000A0000~NN (NP_00000B00000000000000A0000~NN DT JJ NN_00000B00000000000000A0000))))))
Description : Verb phrase

State 9 : (NP_A000000C0000000000B000000~NN NN_A000000C0000000000B000000)
Description : Noun phrase

State 10 : (NP_B00000000000000000B000000~NN (NP_A000000C0000000000B000000~NN NN_A000000C0000000000B000000) , (VP AUX (ADVP RB) (VP (ADVP AUXG) (VP VBN (PP_0000000000000000000A0000 IN (NP_00000D00000000000000A0000~NN (NP_00000B00000000000000A0000~NN DT JJ NN_00000B00000000000000A0000)))))))
Description : Noun phrase

State 11 : (NP_A00000000000000000D000000~NN (NP_B00000000000000000B000000~NN (NP_A000000C0000000000B000000~NN

NN_A000000C0000000000B000000) , (VP AUX (ADVP RB) (VP (ADVP AUXG) (VP VBN (PP_000000000000000000000A0000 IN (NP_00000D000000000000000A0000~NN (NP_00000B000000000000000A0000~NN DT JJ NN_00000B000000000000000A0000))))))))
Description : Noun phrase

State 12 : (NP_A000000000000000000000000~NN (NP_A000000000000000000D000000~NN (NP_B000000000000000000B000000~NN (NP_A000000C0000000000B000000~NN NN_A000000C0000000000B000000) , (VP AUX (ADVP RB) (VP (ADVP AUXG) (VP VBN (PP_000000000000000000000A0000 IN (NP_00000D000000000000000A0000~NN (NP_00000B000000000000000A0000~NN DT JJ NN_00000B000000000000000A0000)))))))))
Description : Noun phrase

CC DT NN_A0D0000000B000000000000000 RB VBD , CC NNP_000000000000000000000000A NNS_A00000B000000000000000000 VBD NNP_000000000000000000000000A CD , CD .
Parse failed - the tail of a phrase could not be found!
Desired parse : (S1 (S (S CC (NP DT NN) (ADVP RB) (VP VBD)) , CC (S (NP NNP NNS) (VP VBD (NP NNP CD , CD))) .))

wsj_Concord-II's attempt ....

State 1 : (UCP CD , CD)
Description : Unlike coordinated phrase

State 2 : (NP_000000000000000000000000A~NNP NNP_000000000000000000000000A)
Description : Noun phrase

State 3 : (NP_000000000000000000000000A~NNP (NP_000000000000000000000000A~NNP NNP_000000000000000000000000A) (UCP CD , CD))
Description : Noun phrase

State 4 : (VP VBD (NP_000000000000000000000000A~NNP (NP_000000000000000000000000A~NNP NNP_000000000000000000000000A) (UCP CD , CD)))
Description : Verb phrase

State 5 : (NP_A00000B000000000000000000A~NNS NNP_000000000000000000000000A NNS_A00000B000000000000000000)
Description : Noun phrase

State 6 : (S (NP_A00000B000000000000000000A~NNS NNP_000000000000000000000000A NNS_A00000B000000000000000000) (VP VBD (NP_000000000000000000000000A~NNP (NP_000000000000000000000000A~NNP NNP_000000000000000000000000A) (UCP CD , CD))))
Description : Simple declarative clause

State 7 : (S ,)
Description : Simple declarative clause

State 8 : (LST (S ,) CC)
Description : List marker phrase

Parse failed - could not create a valid parent for S1

Desired parse : (S1 (S (S (NP -NONE-) (VP NNS (NP NNP NNP))) , `` (NP (NP DT NNS) (PP IN (NP DT NNS))) (VP AUX (ADJP DT JJ CC JJ)) . ''))

wsj_Concord-II's attempt ....

State 1 : (ADJP DT JJ CC JJ)
Description : Adjective phrase

State 2 : (VP AUX (ADJP DT JJ CC JJ))
Description : Verb phrase

State 3 : (NP_00000AC000000000000000000~NNS DT NNS_00000AC000000000000000000)
Description : Noun phrase

State 4 : (S (NP_00000AC000000000000000000~NNS DT NNS_00000AC000000000000000000) (VP AUX (ADJP DT JJ CC JJ)))
Description : Simple declarative clause

State 5 : (S1 IN (S (NP_00000AC000000000000000000~NNS DT NNS_00000AC000000000000000000) (VP AUX (ADJP DT JJ CC JJ))))
Description : Root node

State 6 : (NP_D00000A000000000000000000~NNS DT NNS_C00000A000000000000000000)
Description : Noun phrase

State 7 : (NP_000000A0000000A0000000000~NNP NNP_000000A000000000000000000 NNP_00000000000000A0000000000)
Description : Noun phrase

State 8 : (VP NNS_00A000000000000000000000 (NP_000000A0000000A0000000000~NNP NNP_000000A000000000000000000 NNP_00000000000000A0000000000))
Description : Verb phrase

State 9 : (NP~-NONE- -NONE-)
Description : Noun phrase

State 10 : (S (NP~-NONE- -NONE-) (VP NNS_00A000000000000000000000 (NP_000000A0000000A0000000000~NNP NNP_000000A000000000000000000 NNP_00000000000000A0000000000)))
Description : Simple declarative clause

State 11 : (S1 (S (NP~-NONE- -NONE-) (VP NNS_00A000000000000000000000 (NP_000000A0000000A0000000000~NNP NNP_000000A000000000000000000 NNP_00000000000000A0000000000))) ,)

Description : Root node

Parse failed - could not create a valid parent for VP

Desired parse : (S1 (S (PP IN (NP CD NN)) (NP NNP NNP) (ADVP RB) (VP VBD (S (NP DT NN NN) (VP TO (VP VB (PP IN (NP (NP JJ NNS) , (NP JJ NNS) CC (NP (NP DT NN) (PP IN (NP (NP NNS) (PP IN (NP (NP DT NN) (SBAR (WHNP -NONE-) (S (NP PRP) (VP VBD (S (NP -NONE-) (VP TO (VP VB (NP -NONE-)))))))))))))))))) .))

wsj_Concord-II's attempt ....

State 1 : (NP~-NONE- -NONE-)
Description : Noun phrase

State 2 : (S (NP~-NONE- -NONE-))
Description : Simple declarative clause

State 3 : (VP VB (S (NP~-NONE- -NONE-)))
Description : Verb phrase

State 4 : (VP (VP VB (S (NP~-NONE- -NONE-))) .)
Description : Verb phrase

Parse failed - could not create a valid parent for NP_000000A000000000000000000~NN

Desired parse : (S1 (S (PP VBG (NP DT (UCP NN CC JJ) NN NNS)) , (NP (NP DT NN) (PP IN (NP (NP NNS) (VP VBG (NP NN NNS) (PP IN (NP (NP DT NN) (VP VBN (NP NNP CD)))))))) (VP VBD (PP TO (NP CD)) (PP IN (NP CD) (ADVP (NP DT NN) RBR))) .))

wsj_Concord-II's attempt ....

State 1 : (ADVP RBR)
Description : Adverb phrase

State 2 : (NP_000000000000000000000000A~PRP CD DT NN_000000000000000000000000A)
Description : Noun phrase

State 3 : (NP_000000000000000000000000A~JJS (NP_000000000000000000000000A~PRP CD DT NN_000000000000000000000000A) (ADVP RBR))
Description : Noun phrase

State 4 : (PP_000000000000000000000000A IN (NP_000000000000000000000000A~JJS (NP_000000000000000000000000A~PRP CD DT NN_000000000000000000000000A) (ADVP RBR)))
Description : Prepositional phrase

State 5 : (NP~CD CD)
Description : Noun phrase

State 6 : (PP_000000000000000000000000D TO (NP~CD CD) (PP_000000000000000000000000A IN (NP_000000000000000000000000A~JJS (NP_000000000000000000000000A~PRP CD DT NN_000000000000000000000000A) (ADVP RBR))))
Description : Prepositional phrase

State 7 : (VP VBD (PP_000000000000000000000000D TO (NP~CD CD) (PP_000000000000000000000000A IN (NP_000000000000000000000000A~JJS (NP_000000000000000000000000A~PRP CD DT NN_000000000000000000000000A) (ADVP RBR)))))
Description : Verb phrase

State 8 : (NP~NNS CD)
Description : Noun phrase

State 9 : (NP_000000000000000000000000A~NNS NNP_000000000000000000000000A (NP~NNS CD))

Description : Noun phrase

State 10 : (VP VBN (NP_0000000000000000000000A~NNS NNP_0000000000000000000000A (NP~NNS CD)))
Description : Verb phrase

State 11 : (NP_0000000000000000000000A~NN DT NN_0000000000000000000000A)
Description : Noun phrase

State 12 : (NP_0000000000000000000000A~NN (NP_0000000000000000000000A~NN DT NN_0000000000000000000000A) (VP VBN (NP_0000000000000000000000A~NNS NNP_0000000000000000000000A (NP~NNS CD))))
Description : Noun phrase

State 13 : (NP_0000000000000000000000A~NN (NP_0000000000000000000000A~NN (NP_0000000000000000000000A~NN DT NN_0000000000000000000000A) (VP VBN (NP_0000000000000000000000A~NNS NNP_0000000000000000000000A (NP~NNS CD)))) (VP VBD (PP_0000000000000000000000D TO (NP~CD CD) (PP_0000000000000000000000A IN (NP_0000000000000000000000A~JJS (NP_0000000000000000000000A~PRP CD DT NN_0000000000000000000000A) (ADVP RBR))))))
Description : Noun phrase

State 14 : (PP_0000000000000000000000A IN (NP_0000000000000000000000A~NN (NP_0000000000000000000000A~NN (NP_0000000000000000000000A~NN DT NN_0000000000000000000000A) (VP VBN (NP_0000000000000000000000A~NNS NNP_0000000000000000000000A (NP~NNS CD)))) (VP VBD (PP_0000000000000000000000D TO (NP~CD CD) (PP_0000000000000000000000A IN (NP_0000000000000000000000A~JJS (NP_0000000000000000000000A~PRP CD DT NN_0000000000000000000000A) (ADVP RBR)))))))
Description : Prepositional phrase

State 15 : (NP_000A00C0000000000A0000A00~NNS NN_000000000000000000000A00 NNS_000A00C0000000000A0000000)
Description : Noun phrase

State 16 : (NP_000A000000000000A0000A00~NNS (NP_000A00C0000000000A0000A00~NNS NN_000000000000000000000A00 NNS_000A00C0000000000A0000000) (PP_000000000000000000000000A IN (NP_000000000000000000000000A~NN (NP_000000000000000000000000A~NN (NP_000000000000000000000000A~NN DT NN_000000000000000000000000A) (VP VBN (NP_000000000000000000000000A~NNS NNP_000000000000000000000000A (NP~NNS CD)))) (VP VBD (PP_000000000000000000000000D TO (NP~CD CD) (PP_000000000000000000000000A IN (NP_000000000000000000000000A~JJS (NP_000000000000000000000000A~PRP CD DT NN_000000000000000000000000A) (ADVP RBR)))))))
Description : Noun phrase

State 17 : (VP VBG (NP_000A000000000000A0000A00~NNS (NP_000A00C0000000000A0000A00~NNS NN_000000000000000000000A00 NNS_000A00C0000000000A0000000) (PP_000000000000000000000000A IN (NP_000000000000000000000000A~NN (NP_000000000000000000000000A~NN (NP_000000000000000000000000A~NN DT NN_000000000000000000000000A) (VP VBN (NP_000000000000000000000000A~NNS NNP_000000000000000000000000A (NP~NNS CD)))) (VP VBD (PP_000000000000000000000000D TO (NP~CD CD) (PP_000000000000000000000000A IN (NP_000000000000000000000000A~JJS (NP_000000000000000000000000A~PRP CD DT NN_000000000000000000000000A) (ADVP RBR)))))))))
Description : Verb phrase

State 18 : (NP_0000000000A00000000000000~NNS NNS_0000000000A00000000000000)
Description : Noun phrase

State 19 : (S (NP_0000000000A00000000000000~NNS NNS_0000000000A00000000000000) (VP VBG (NP_000A000000000000A0000A00~NNS

(NP_000A00C0000000000A0000A00~NNS NN_0000000000000000000000A00 NNS_000A00C0000000000A0000000) (PP_0000000000000000000000000A IN (NP_0000000000000000000000000A~NN (NP_0000000000000000000000000A~NN (NP_0000000000000000000000000A~NN DT NN_0000000000000000000000000A) (VP VBN (NP_0000000000000000000000000A~NNS NNP_0000000000000000000000000A (NP~NNS CD)))) (VP VBD (PP_0000000000000000000000000D TO (NP~CD CD) (PP_0000000000000000000000000A IN (NP_0000000000000000000000000A~JJS (NP_0000000000000000000000000A~PRP CD DT NN_0000000000000000000000000A) (ADVP RBR)))))))))

Description : Simple declarative clause

State 20 : (SBAR IN (S (NP_0000000000A00000000000000~NNS NNS_0000000000A00000000000000) (VP VBG (NP_000A0000000000000A0000A00~NNS (NP_000A00C0000000000A0000A00~NNS NN_0000000000000000000000A00 NNS_000A00C0000000000A0000000) (PP_0000000000000000000000000A IN (NP_0000000000000000000000000A~NN (NP_0000000000000000000000000A~NN (NP_0000000000000000000000000A~NN DT NN_0000000000000000000000000A) (VP VBN (NP_0000000000000000000000000A~NNS NNP_0000000000000000000000000A (NP~NNS CD)))) (VP VBD (PP_0000000000000000000000000D TO (NP~CD CD) (PP_0000000000000000000000000A IN (NP_0000000000000000000000000A~JJS (NP_0000000000000000000000000A~PRP CD DT NN_0000000000000000000000000A) (ADVP RBR)))))))))))

Description : Clause introduced by sub-ordinating conjunction

State 21 : (NP_000A00C000000000000D00000~NN DT NN_000A00C000000000000B00000)

Description : Noun phrase

State 22 : (ADJP JJ NN_000A00C0000000000A0000000)

Description : Adjective phrase

State 23 : (NP_000B00A000000000000000000~NN (NP_000A00C000000000000D00000~NN DT NN_000A00C000000000000B00000) (SBAR IN (S (NP_0000000000A00000000000000~NNS

NNS_0000000000A00000000000000) (VP VBG (NP_000A0000000000000A0000A00~NNS (NP_000A00C0000000000A0000A00~NNS NN_0000000000000000000000A00 NNS_000A00C0000000000A0000000) (PP_0000000000000000000000000A IN (NP_0000000000000000000000000A~NN (NP_0000000000000000000000000A~NN (NP_0000000000000000000000000A~NN DT NN_0000000000000000000000000A) (VP VBN (NP_0000000000000000000000000A~NNS NNP_0000000000000000000000000A (NP~NNS CD)))) (VP VBD (PP_0000000000000000000000000D TO (NP~CD CD) (PP_0000000000000000000000000A IN (NP_0000000000000000000000000A~JJS (NP_0000000000000000000000000A~PRP CD DT NN_0000000000000000000000000A) (ADVP RBR))))))))))))

Description : Noun phrase

State 24 : (NP_000000A000000000000000000~NN (NP_000B00A000000000000000000~NN (NP_000A00C000000000000D00000~NN DT NN_000A00C000000000000B00000) (SBAR IN (S (NP_0000000000A00000000000000~NNS NNS_0000000000A00000000000000) (VP VBG (NP_000A0000000000000A0000A00~NNS (NP_000A00C0000000000A0000A00~NNS NN_0000000000000000000000A00 NNS_000A00C0000000000A0000000) (PP_0000000000000000000000000A IN (NP_0000000000000000000000000A~NN (NP_0000000000000000000000000A~NN (NP_0000000000000000000000000A~NN DT NN_0000000000000000000000000A) (VP VBN (NP_0000000000000000000000000A~NNS NNP_0000000000000000000000000A (NP~NNS CD)))) (VP VBD (PP_0000000000000000000000000D TO (NP~CD CD) (PP_0000000000000000000000000A IN (NP_0000000000000000000000000A~JJS (NP_0000000000000000000000000A~PRP CD DT NN_0000000000000000000000000A) (ADVP RBR))))))))))))) .)

Description : Noun phrase

Parse failed - could not create a valid parent for VP

Desired parse : (S1 (PRN -LRB- (VP VBP (NP (NP VBN NN) : `` (S (NP (NP NNP POS) NNP) (VP VBZ (S (NP NNP NNP POS) (VP VBG '' (S (NP -NONE-) (VP TO (VP VB (NP DT JJ NN NN)))))))) '' : (NP (NP NNP) (NP NNP CD , CD)))) -RRB-))

wsj_Concord-II's attempt ....

State 1 : (NP_000000000000000000000000A~CD NNP_000000000000000000000000A CD , CD)
Description : Noun phrase

State 2 : (NP_0000000000A00000000000000~NNP NNP_0000000000A00000000000000)
Description : Noun phrase

State 3 : (NP_A0A000C000A00000000000000~NN DT JJ NN_A000000000000000000000000 NN_00A000C000A00000000000000)
Description : Noun phrase

State 4 : (VP TO)
Description : Verb phrase

Parse failed - could not create a valid parent for NP_000A000000000000000000B00~NNS

Desired parse : (S1 (S (PP IN (NP JJ NNS)) , (NP (NP NNS) (PP IN (NP NNP))) (VP AUX (VP VBN (NP (NP NNS) (PP IN (NP NN))) (PP IN (NP (NP DT NN NNS) (PP IN (NP (NP NNS) (VP VBN (S (NP -NONE-) (ADJP JJ (PP IN (NP NN)))))))))) .))

wsj_Concord-II's attempt ....

State 1 : (NP_A00000000000000000B000B00~NN NN_A00000000000000000B000B00)
Description : Noun phrase

State 2 : (PP_A00000000000000000B000B00 IN (NP_A00000000000000000B000B00~NN NN_A00000000000000000B000B00))
Description : Prepositional phrase

State 3 : (ADJP JJ)
Description : Adjective phrase

State 4 : (NP~-NONE- -NONE-)
Description : Noun phrase

State 5 : (S (NP~-NONE- -NONE-) (ADJP JJ) (PP_A00000000000000000B000B00 IN (NP_A00000000000000000B000B00~NN NN_A00000000000000000B000B00)))
Description : Simple declarative clause

State 6 : (VP VBN (S (NP~-NONE- -NONE-) (ADJP JJ) (PP_A00000000000000000B000B00 IN (NP_A00000000000000000B000B00~NN NN_A00000000000000000B000B00))))
Description : Verb phrase

State 7 : (NP_000A00000000000000000B00~NNS NNS_000A00000000000000000B00)
Description : Noun phrase

NN_A00000000000000000000000A IN DT NN_A00000000000000000000000000 NNS_000C00000000000A0000000 AUX RB JJ IN JJ NN_00000000000000000A0000000 NN_00000CA000A00000000000000 .
Parse failed - the tail of a phrase could not be found!
Desired parse : (S1 (S (NP (NP NN) (PP IN (NP DT NN NNS))) (VP AUX (ADJP RB JJ) (PP IN (NP JJ NN NN))) .))

wsj_Concord-II's attempt ....

State 1 : (NP_00000CA000A000000A0000000~NN JJ NN_00000000000000000A0000000 NN_00000CA000A00000000000000)

Description : Noun phrase

State 2 : (PP_00000CA000A000000A0000000 IN (NP_00000CA000A000000A0000000~NN JJ NN_00000000000000000A0000000 NN_00000CA000A00000000000000))
Description : Prepositional phrase

State 3 : (ADJP JJ (PP_00000CA000A000000A0000000 IN (NP_00000CA000A000000A0000000~NN JJ NN_00000000000000000A0000000 NN_00000CA000A00000000000000)))
Description : Adjective phrase

State 4 : (ADVP RB)
Description : Adverb phrase

State 5 : (PRN (ADVP RB) (ADJP JJ (PP_00000CA000A000000A0000000 IN (NP_00000CA000A000000A0000000~NN JJ NN_00000000000000000A0000000 NN_00000CA000A00000000000000))))
Description : Parenthetical

State 6 : (INTJ (PRN (ADVP RB) (ADJP JJ (PP_00000CA000A000000A0000000 IN (NP_00000CA000A000000A0000000~NN JJ NN_00000000000000000A0000000 NN_00000CA000A00000000000000)))))
Description : Interjection - corresponds approximately to the word tag 'UH'

State 7 : (PRN (INTJ (PRN (ADVP RB) (ADJP JJ (PP_00000CA000A000000A0000000 IN (NP_00000CA000A000000A0000000~NN JJ NN_00000000000000000A0000000 NN_00000CA000A00000000000000))))) .)
Description : Parenthetical

State 8 : (VP AUX)
Description : Verb phrase

State 9 : (NP_000D0000000000000A0000000~NNS NNS_000C0000000000000A0000000)
Description : Noun phrase

State 10 : (S (NP_000D0000000000000A0000000~NNS NNS_000C0000000000000A0000000) (VP AUX))
Description : Simple declarative clause

State 11 : (RRC DT NN_A000000000000000000000000 (S (NP_000D0000000000000A0000000~NNS NNS_000C0000000000000A0000000) (VP AUX)))
Description : Reduced relative clause

State 12 : (S1 IN (RRC DT NN_A000000000000000000000000 (S (NP_000D0000000000000A0000000~NNS NNS_000C0000000000000A0000000) (VP AUX))) (PRN (INTJ (PRN (ADVP RB) (ADJP JJ (PP_00000CA000A000000A0000000 IN (NP_00000CA000A000000A0000000~NN JJ NN_00000000000000000A0000000 NN_00000CA000A00000000000000))))) .))
Description : Root node

State 13 : (NP_A0000000000000000000000A~PRP NN_A0000000000000000000000A)
Description : Noun phrase

Parse failed - could not create a valid parent for NP_0000000000A00000000000000~NNS

Desired parse : (S1 (S (NP DT NN NNS) (VP VBD (PP IN (NP (NP NNP NNPS NNP) , (NP (NP DT JJ NN NN) (SBAR (WHNP WDT) (S (NP NNP) (ADVP RB) (VP VBD (NP -NONE-)))))))) .))

wsj_Concord-II's attempt ....

State 1 : (NP~-NONE- -NONE-)
Description : Noun phrase

State 2 : (S (NP~-NONE- -NONE-) .)
Description : Simple declarative clause

State 3 : (VP VBD)
Description : Verb phrase

State 4 : (ADVP RB)
Description : Adverb phrase

State 5 : (VP (ADVP RB) (VP VBD))
Description : Verb phrase

State 6 : (NP_00000000000000A0000000000~NNP NNP_00000000000000A0000000000)
Description : Noun phrase

State 7 : (NP~NNPS WDT (NP_00000000000000A0000000000~NNP NNP_00000000000000A0000000000))
Description : Noun phrase

State 8 : (S (NP~NNPS WDT (NP_00000000000000A0000000000~NNP NNP_00000000000000A0000000000)) (VP (ADVP RB) (VP VBD)))
Description : Simple declarative clause

State 9 : (WHADVP (S (NP~NNPS WDT (NP_00000000000000A0000000000~NNP NNP_00000000000000A0000000000)) (VP (ADVP RB) (VP VBD))))
Description : Wh-adverb phrase

State 10 : (ADJP DT JJ)
Description : Adjective phrase

State 11 : (NP_000000D000A000A0000000D00~NN NN_000000B00A0000B0000000000 NN_0000000000A00000000000C00)
Description : Noun phrase

State 12 : (NP_0000000000A00000000000000~NNS NNPS_0000000000A00000000000000 NNP_0000000000A00000000000000)
Description : Noun phrase

State 13 : (NP_0000000000A00000000000000~NNS (NP_0000000000A00000000000000~NNS NNPS_0000000000A00000000000000 NNP_0000000000A00000000000000) ,)
Description : Noun phrase

State 14 : (NP_0000000000A00000000000000~NNS NNP_0000000000A00000000000000 (NP_0000000000A00000000000000~NNS (NP_0000000000A00000000000000~NNS NNPS_0000000000A00000000000000 NNP_0000000000A00000000000000) ,))
Description : Noun phrase

Parse failed - could not create a valid parent for NP_00A00000000A0000000000000~EX

Desired parse : (S1 (S (NP (NP NNS) (PP IN (NP JJ NNS))) (VP VBD (NP (NP NNP POS) JJ NN) (SBAR IN (S (NP PRP) (VP VBD (PP TO (NP (NP NNP POS) NN)) (PP IN (S (NP -NONE-) (VP VBG (NP NNP NNP)))))))) .))

wsj_Concord-II's attempt ....

State 1 : (ADJP NNP_00000000000A0000000000000)
Description : Adjective phrase

State 2 : (NP_00000000000A0000000000000~NNP NNP_00000000000A0000000000000)
Description : Noun phrase

State 3 : (VP VBG (ADJP NNP_00000000000A0000000000000) (NP_00000000000A0000000000000~NNP NNP_00000000000A0000000000000))
Description : Verb phrase

State 4 : (NP~-NONE- -NONE-)
Description : Noun phrase

State 5 : (S (NP~-NONE- -NONE-) (VP VBG (ADJP NNP_00000000000A0000000000000) (NP_00000000000A0000000000000~NNP NNP_00000000000A0000000000000)))
Description : Simple declarative clause

State 6 : (PP IN (S (NP~-NONE- -NONE-) (VP VBG (ADJP NNP_00000000000A0000000000000) (NP_00000000000A0000000000000~NNP NNP_00000000000A0000000000000))))
Description : Prepositional phrase

State 7 : (NP_00000000000A0000000000000~POS NNP_00000000000A0000000000000 POS)
Description : Noun phrase

State 8 : (NP_00D00000000A000000000B000~POS (NP_00000000000A0000000000000~POS NNP_00000000000A0000000000000 POS) NN_00D00000000A000000000B000)
Description : Noun phrase

State 9 : (PP_00D00000000A000000000B000 TO (NP_00D00000000A000000000B000~POS (NP_00000000000A0000000000000~POS NNP_00000000000A0000000000000 POS) NN_00D00000000A000000000B000))
Description : Prepositional phrase

State 10 : (VP VBD (PP_00D00000000A000000000B000 TO (NP_00D00000000A000000000B000~POS (NP_00000000000A0000000000000~POS NNP_00000000000A0000000000000 POS) NN_00D00000000A000000000B000)) (PP IN (S (NP~-NONE- -NONE-) (VP VBG (ADJP NNP_00000000000A0000000000000) (NP_00000000000A0000000000000~NNP NNP_00000000000A0000000000000)))))
Description : Verb phrase

State 11 : (NP_0000000000A00000000000000~PRP PRP_0000000000A00000000000000)
Description : Noun phrase

State 12 : (S (NP_0000000000A00000000000000~PRP PRP_0000000000A00000000000000) (VP VBD (PP_00D00000000A000000000B000 TO (NP_00D00000000A000000000B000~POS (NP_00000000000A0000000000000~POS NNP_00000000000A0000000000000 POS) NN_00D00000000A000000000B000)) (PP IN (S (NP~-NONE- -NONE-) (VP VBG (ADJP NNP_00000000000A0000000000000) (NP_00000000000A0000000000000~NNP NNP_00000000000A0000000000000))))))
Description : Simple declarative clause

State 13 : (SBAR IN (S (NP_0000000000A00000000000000~PRP PRP_0000000000A00000000000000) (VP VBD (PP_00D00000000A000000000B000 TO (NP_00D00000000A000000000B000~POS (NP_00000000000A0000000000000~POS NNP_00000000000A0000000000000 POS) NN_00D00000000A000000000B000)) (PP IN (S (NP~-NONE- -NONE-) (VP VBG (ADJP NNP_00000000000A0000000000000) (NP_00000000000A0000000000000~NNP NNP_00000000000A0000000000000)))))))
Description : Clause introduced by sub-ordinating conjunction

State 14 : (NP_00000000000A0000000000000~POS NNP_00000000000A0000000000000 POS)
Description : Noun phrase

State 15 : (NP_00A00000000A0000000000000~POS (NP_00000000000A0000000000000~POS NNP_00000000000A0000000000000 POS) JJ NN_00A0000000000000000000000)
Description : Noun phrase

State 16 : (NP_00A00000000A0000000000000~EX (NP_00A00000000A0000000000000~POS (NP_00000000000A0000000000000~POS NNP_00000000000A0000000000000 POS) JJ NN_00A0000000000000000000000) (SBAR IN (S (NP_0000000000A00000000000000~PRP PRP_0000000000A00000000000000) (VP VBD (PP_00D00000000A000000000B000 TO (NP_00D00000000A000000000B000~POS

(NP_00000000000A0000000000000~POS NNP_00000000000A0000000000000 POS) NN_00D00000000A000000000B000)) (PP IN (S (NP~-NONE- -NONE-) (VP VBG (ADJP NNP_00000000000A0000000000000) (NP_00000000000A0000000000000~NNP NNP_00000000000A0000000000000))))))))
Description : Noun phrase

Parse failed - could not create a valid parent for ADVP

Desired parse : (S1 (S (NP PRP) (VP VBD (SBAR -NONE- (S (NP DT NNS) (VP AUX (VP VBN (NP (NP NN POS) NN)))))) .))

wsj_Concord-II's attempt ....

State 1 : (UCP POS)
Description : Unlike coordinated phrase

State 2 : (NP_000A00000000000000000000~NN (UCP POS) NN_000A00000000000000000000)
Description : Noun phrase

State 3 : (NP_A00D00000000000000000000~NN NN_A000000000B00000000000000 (NP_000A00000000000000000000~NN (UCP POS) NN_000A00000000000000000000))
Description : Noun phrase

State 4 : (VP VBN (NP_A00D00000000000000000000~NN NN_A000000000B00000000000000 (NP_000A00000000000000000000~NN (UCP POS) NN_000A00000000000000000000)))
Description : Verb phrase

State 5 : (ADVP (VP VBN (NP_A00D00000000000000000000~NN NN_A000000000B00000000000000 (NP_000A00000000000000000000~NN (UCP POS) NN_000A00000000000000000000))))
Description : Adverb phrase

DT NN_0000000000A00000000000C00 RB VBD DT NN_A00B00B000B00000000000000 NN_000000A000000000000000000 IN $ CD CD -NONE- IN JJ JJ NNS_000C00A0000000000D0000000 .

Parse failed - the tail of a phrase could not be found!

Desired parse : (S1 (S (NP DT NN) (ADVP RB) (VP VBD (NP (NP DT NN NN) (PP IN (NP (QP $ CD CD) -NONE-)) (PP IN (NP JJ JJ NNS)))) .))

wsj_Concord-II's attempt ....

State 1 : (NP_000C00A0000000000D0000000~NNS JJ JJ NNS_000C00A0000000000D0000000)
Description : Noun phrase

State 2 : (PP_000C00A0000000000D0000000 IN (NP_000C00A0000000000D0000000~NNS JJ JJ NNS_000C00A0000000000D0000000))
Description : Prepositional phrase

State 3 : (WHADJP CD CD)
Description : Wh-adjective phrase

State 4 : (S -NONE-)
Description : Simple declarative clause

State 5 : (UCP (S -NONE-) (PP_000C00A0000000000D0000000 IN (NP_000C00A0000000000D0000000~NNS JJ JJ NNS_000C00A0000000000D0000000)))
Description : Unlike coordinated phrase

State 6 : (INTJ (WHADJP CD CD) (UCP (S -NONE-) (PP_000C00A0000000000D0000000 IN (NP_000C00A0000000000D0000000~NNS JJ JJ NNS_000C00A0000000000D0000000))))
Description : Interjection - corresponds approximately to the word tag 'UH'

`` PRP_0000000000A00000000000000 AUX IN DT NN_000A00000000000000000B0B00 IN -NONE- VBG NN_000000000000000A0000000000 NNS_000A00C0000000000A0000000 , '' VBZ -NONE- NNP_000000000000000A0000000000 NNP_000000000000000A0000000000 , JJ NN_B00A00000000000000000000000 NN_000000000000000A0000000000 IN NNP_0000000000A00000000000000 NNP_0000000000A00000000000000 , NNP_000000000000A0000000000000 , NNP_000000000000A0000000000000 .

Parse failed - the tail of a phrase could not be found!

Desired parse : (S1 (SINV `` (S (NP PRP) (VP AUX (PP IN (NP (NP DT NN) (PP IN (S (NP -NONE-) (VP VBG (NP NN NNS)))))))) , '' (VP VBZ (S -NONE-)) (NP (NP NNP NNP) , (NP (NP JJ NN NN) (PP IN (NP (NP NNP NNP) , (NP (NP NNP) , (NP NNP)))))) .))

wsj_Concord-II's attempt ....

State 1 : (NP_0000000000A0000000000000~NNP NNP_0000000000A0000000000000)
Description : Noun phrase

State 2 : (NP_0000000000A0000000000000~NNP NNP_0000000000A0000000000000)
Description : Noun phrase

State 3 : (NP_0000000000A00000000000000~NNP NNP_0000000000A00000000000000 NNP_0000000000A00000000000000)
Description : Noun phrase

State 4 : (NP_0000000000AA000000000000~NNP (NP_0000000000A00000000000000~NNP NNP_0000000000A00000000000000 NNP_0000000000A00000000000000) , (NP_0000000000A0000000000000~NNP NNP_0000000000A0000000000000))
Description : Noun phrase

State 5 : (NP_0000000000AA000000000000~NNP (NP_0000000000AA000000000000~NNP

(NP_0000000000A00000000000000~NNP NNP_0000000000A00000000000000 NNP_0000000000A00000000000000) , (NP_00000000000A0000000000000~NNP NNP_00000000000A0000000000000)) , (NP_00000000000A0000000000000~NNP NNP_00000000000A0000000000000))
Description : Noun phrase

State 6 : (PP_0000000000AA0000000000000 IN (NP_0000000000AA0000000000000~NNP (NP_0000000000AA0000000000000~NNP (NP_0000000000A00000000000000~NNP NNP_0000000000A00000000000000 NNP_0000000000A00000000000000) , (NP_00000000000A0000000000000~NNP NNP_00000000000A0000000000000)) , (NP_00000000000A0000000000000~NNP NNP_00000000000A0000000000000)))
Description : Prepositional phrase

State 7 : (NP_B00A0000000000A0000000000~NN JJ NN_B00A000000000000000000000 NN_00000000000000A0000000000)
Description : Noun phrase

State 8 : (NP_00000000000000A0000000000~NNP NNP_00000000000000A0000000000 NNP_00000000000000A0000000000)
Description : Noun phrase

State 9 : (VP VBZ -NONE- (NP_00000000000000A0000000000~NNP NNP_00000000000000A0000000000 NNP_00000000000000A0000000000))
Description : Verb phrase

State 10 : (INTJ ,)
Description : Interjection - corresponds approximately to the word tag 'UH'

State 11 : (S (INTJ ,) '' (VP VBZ -NONE- (NP_00000000000000A0000000000~NNP NNP_00000000000000A0000000000 NNP_00000000000000A0000000000)))
Description : Simple declarative clause

State 12 : (FRAG , (NP_B00A0000000000A0000000000~NN JJ NN_B00A000000000000000000000 NN_00000000000000A0000000000)

(PP_0000000000AA0000000000000 IN (NP_0000000000AA0000000000000~NNP (NP_0000000000AA0000000000000~NNP (NP_0000000000A00000000000000~NNP NNP_0000000000A00000000000000 NNP_0000000000A00000000000000) , (NP_0000000000A0000000000000~NNP NNP_0000000000A0000000000000)) , (NP_0000000000A0000000000000~NNP NNP_0000000000A0000000000000))))

Description : Fragment

State 13 : (NP_000A00D0000000A00A0000000~NNS NN_0000000000000A0000000000 NNS_000A00C0000000000A0000000)

Description : Noun phrase

State 14 : (NP_000C0000000000000A0000000~NNS (NP_000A00D0000000A00A0000000~NNS NN_0000000000000A0000000000 NNS_000A00C0000000000A0000000) (S (INTJ ,) '' (VP VBZ -NONE- (NP_0000000000000A0000000000~NNP NNP_0000000000000A0000000000 NNP_0000000000000A0000000000))))

Description : Noun phrase

State 15 : (S1 (NP_000C0000000000000A0000000~NNS (NP_000A00D0000000A00A0000000~NNS NN_0000000000000A0000000000 NNS_000A00C0000000000A0000000) (S (INTJ ,) '' (VP VBZ -NONE- (NP_0000000000000A0000000000~NNP NNP_0000000000000A0000000000 NNP_0000000000000A0000000000)))) (FRAG , (NP_B00A0000000000A0000000000~NN JJ NN_B00A000000000000000000000 NN_0000000000000A0000000000) (PP_0000000000AA0000000000000 IN (NP_0000000000AA0000000000000~NNP (NP_0000000000AA0000000000000~NNP (NP_0000000000A00000000000000~NNP NNP_0000000000A00000000000000 NNP_0000000000A00000000000000) , (NP_0000000000A0000000000000~NNP NNP_0000000000A0000000000000)) , (NP_0000000000A0000000000000~NNP NNP_0000000000A0000000000000)))))

Description : Root node

Parse failed - could not create a valid parent for NP_00000000000A00000A0000B00~POS

Desired parse : (S1 (S (S (PP IN (NP JJ NNP)) , (NP DT JJS) (VP NNP (NP -NONE-) (PP IN (NP NN)))) , (NP DT NN) (VP AUX (NP (NP DT NN POS) (ADJP (QP $ CD CD) -NONE-) JJ NN)) .))

wsj_Concord-II's attempt ....

State 1 : (QP $ CD CD)
Description : Adjective phrase (Quantitative)

State 2 : (ADJP (QP $ CD CD) -NONE- JJ)
Description : Adjective phrase

State 3 : (NP_0000000000000000000D00000~POS POS)
Description : Noun phrase

State 4 : (NP_0000B000000A0000000000000~POS DT NN_0000B000000A0000000D00000 (NP_0000000000000000000D00000~POS POS) (ADJP (QP $ CD CD) -NONE- JJ))
Description : Noun phrase

State 5 : (NP_0000CDC0000A00000A0000A00~POS (NP_0000B000000A0000000000000~POS DT NN_0000B000000A0000000D00000 (NP_0000000000000000000D00000~POS POS) (ADJP (QP $ CD CD) -NONE- JJ)) NN_000000C0000000000A0000A00)
Description : Noun phrase

State 6 : (NP_000000D0000A00000A0000B00~POS (NP_0000CDC0000A00000A0000A00~POS (NP_0000B000000A0000000000000~POS DT NN_0000B000000A0000000D00000 (NP_0000000000000000000D00000~POS POS) (ADJP (QP $ CD CD) -NONE- JJ)) NN_000000C0000000000A0000A00))
Description : Noun phrase

State 7 : (NP_00000000000A00000A0000B00~POS (NP_000000D0000A00000A0000B00~POS (NP_0000CDC0000A00000A0000A00~POS (NP_0000B000000A0000000000000~POS DT NN_0000B000000A0000000D00000 (NP_00000000000000000000D00000~POS POS) (ADJP (QP $ CD CD) -NONE- JJ)) NN_000000C0000000000A0000A00)))
Description : Noun phrase

Parse failed - could not create a valid parent for NP_00000000000A0000000000000~POS

Desired parse : (S1 (S (NP NN) , (NP NNP NNP NNP) (VP VBD (S (NP -NONE-) (VP TO (VP VB (NP (NP DT NNP NN NN NNS) (PP IN (NP (NP NNP POS) NNP NNP NNP))) (PP IN (NP (QP RB $ CD CD) -NONE-)))))) .))

wsj_Concord-II's attempt ....

State 1 : (WHADJP $ CD CD)
Description : Wh-adjective phrase

State 2 : (S -NONE-)
Description : Simple declarative clause

State 3 : (S (WHADJP $ CD CD) (S -NONE-))
Description : Simple declarative clause

State 4 : (ADVP RB)
Description : Adverb phrase

State 5 : (SBAR (ADVP RB) (S (WHADJP $ CD CD) (S -NONE-)))
Description : Clause introduced by sub-ordinating conjunction

State 6 : (UCP IN (SBAR (ADVP RB) (S (WHADJP $ CD CD) (S -NONE-))))
Description : Unlike coordinated phrase

State 7 : (NP_0000000000AA00D0000000000~POS NNP_00000000000A0000000000000 POS NNP_0000000000A00000000000000 NNP_0000000000A00000000000000 NNP_0000000000A00000000000000)
Description : Noun phrase

State 8 : (NP_0000000000DA0000000000000~POS (NP_0000000000AA00D0000000000~POS NNP_00000000000A0000000000000 POS NNP_0000000000A00000000000000 NNP_0000000000A00000000000000 NNP_0000000000A00000000000000) (UCP IN (SBAR (ADVP RB) (S (WHADJP $ CD CD) (S -NONE-)))))
Description : Noun phrase

State 9 : (NP_00000000000A0000000000000~POS (NP_0000000000DA0000000000000~POS (NP_0000000000AA00D0000000000~POS NNP_00000000000A0000000000000 POS NNP_0000000000A00000000000000 NNP_0000000000A00000000000000 NNP_0000000000A00000000000000) (UCP IN (SBAR (ADVP RB) (S (WHADJP $ CD CD) (S -NONE-))))))
Description : Noun phrase

Parse failed - could not create a valid parent for NP_00000AA000000000000000000~NN

Desired parse : (S1 (S CC (NP JJ NNS) (VP VBP (SBAR IN (S (NP (NP NNS POS) NNS) (VP AUX (VP JJ (NP -NONE-) (PP IN (NP (NP NN) (PP IN (NP (NP NNS) (PP IN (NP DT JJ NN)))))))))) .))

wsj_Concord-II's attempt ....

State 1 : (NP_00000BA0000000000D0C00000~NN DT JJ NN_00000BA0000000000D0C00000)
Description : Noun phrase

State 2 : (NP_00000BA000000000000000000~NN (NP_00000BA0000000000D0C00000~NN DT JJ NN_00000BA0000000000D0C00000))
Description : Noun phrase

State 3 : (NP_00000AA000000000000000000~NN (NP_00000BA000000000000000000~NN (NP_00000BA0000000000D0C00000~NN DT JJ NN_00000BA0000000000D0C00000)))
Description : Noun phrase

Parse failed - could not create a valid parent for S

Desired parse : (S1 (S (PP IN (PP IN (NP (NP (NP DT NN POS) NN NN) (PP TO (NP JJ NNS))))) , (NP DT NN) (VP VBD : `` (S (NP JJS NNS) (VP VBP (PP IN (NP PRP$ NNS)) (PP IN (NP (NP DT NN) (PP IN (NP (NP (ADJP JJ CC JJ) NNS) (ADJP JJ)))))))) .))

wsj_Concord-II's attempt ....

State 1 : (ADJP JJ)
Description : Adjective phrase

State 2 : (NP_0000000000000000B0000000~NNS JJ CC JJ NNS_000000B0000000000A0000000)
Description : Noun phrase

State 3 : (PP_0000000000000000A0000000 IN (NP_0000000000000000B0000000~NNS JJ CC JJ NNS_000000B0000000000A0000000))
Description : Prepositional phrase

State 4 : (NP_00000000000000000000A0000~NN DT NN_00000000000000000000A0000)
Description : Noun phrase

State 5 : (NP_0000000000000000A00A0000~NN (NP_0000000000000000000A0000~NN DT NN_0000000000000000000A0000) (PP_000000000000000A0000000 IN (NP_00000000000000000B0000000~NNS JJ CC JJ NNS_000000B0000000000A0000000)))
Description : Noun phrase

State 6 : (NP_0000000000000000A00A0000~NN (NP_0000000000000000A00A0000~NN (NP_0000000000000000000A0000~NN DT NN_0000000000000000000A0000) (PP_000000000000000000A0000000 IN (NP_00000000000000000B0000000~NNS JJ CC JJ NNS_000000B0000000000A0000000))) (ADJP JJ))
Description : Noun phrase

State 7 : (PP_000000000000000000A00A0000 IN (NP_0000000000000000A00A0000~NN (NP_0000000000000000A00A0000~NN (NP_0000000000000000000A0000~NN DT NN_0000000000000000000A0000) (PP_000000000000000000A0000000 IN (NP_00000000000000000B0000000~NNS JJ CC JJ NNS_000000B0000000000A0000000))) (ADJP JJ)))
Description : Prepositional phrase

State 8 : (NP_000B000000A000C00C0000000~NNS PRP$ NNS_000B000000A000C00C0000000)
Description : Noun phrase

State 9 : (PP_000B000000A000C00C0000000 IN (NP_000B000000A000C00C0000000~NNS PRP$ NNS_000B000000A000C00C0000000))
Description : Prepositional phrase

State 10 : (VP VBP (PP_000B000000A000C00C0000000 IN (NP_000B000000A000C00C0000000~NNS PRP$ NNS_000B000000A000C00C0000000)))
Description : Verb phrase

State 11 : (NP_00000000000000A0000000000~NNS JJS NNS_00000000000000A0000000000)
Description : Noun phrase

State 12 : (S ``)

Description : Simple declarative clause

Appendix G: A Sample of Matching Parses from the WSJ Corpus Test Set (Using Syntactic Information Only)

JJ NNS IN DT NN IN NN :
Desired parse : (S1 (NP (NP (NP JJ NNS) (PP IN (NP (NP DT NN) (PP IN (NP NN))))) :))
Actual parse : (S1 (NP (NP (NP JJ NNS) (PP IN (NP (NP DT NN) (PP IN (NP NN))))) :))

DT NN IN DT NNPS IN DT NNP AUX IN JJ NN .
Desired parse : (S1 (S (NP (NP DT NN) (PP IN (NP (NP DT NNPS) (PP IN (NP DT NNP))))) (VP AUX (PP IN (NP JJ NN))) .))
Actual parse : (S1 (S (NP (NP DT NN) (PP IN (NP (NP DT NNPS) (PP IN (NP DT NNP))))) (VP AUX (PP IN (NP JJ NN))) .))

NNP : JJ NNS .
Desired parse : (S1 (NP (NP NNP) : (NP JJ NNS) .))
Actual parse : (S1 (NP (NP NNP) : (NP JJ NNS) .))

DT NN AUX VBN IN -NONE- VBG DT NN POS NN RB .
Desired parse : (S1 (S (NP DT NN) (VP AUX (VP VBN (PP IN (S (NP -NONE-) (VP VBG (NP (NP DT NN POS) NN) (ADVP RB)))))) .))
Actual parse : (S1 (S (NP DT NN) (VP AUX (VP VBN (PP IN (S (NP -NONE-) (VP VBG (NP (NP DT NN POS) NN) (ADVP RB)))))) .))

NN
Desired parse : (S1 (NP NN))
Actual parse : (S1 (NP NN))

NNP NNP , DT JJ NN NN , VBZ NNS CC NNS .
Desired parse : (S1 (S (NP (NP NNP NNP) , (NP DT JJ NN NN) ,) (VP VBZ (NP NNS CC NNS)) .))
Actual parse : (S1 (S (NP (NP NNP NNP) , (NP DT JJ NN NN) ,) (VP VBZ (NP NNS CC NNS)) .))

IN NN , DT NN VBZ -NONE- TO VB JJ NNS .
Desired parse : (S1 (S (PP IN (NP NN)) , (NP DT NN) (VP VBZ (S (NP -NONE-) (VP TO (VP VB (NP JJ NNS))))) .))
Actual parse : (S1 (S (PP IN (NP NN)) , (NP DT NN) (VP VBZ (S (NP -NONE-) (VP TO (VP VB (NP JJ NNS))))) .))

RB , DT NN AUX VBN -NONE- TO VB PRP$ JJ NNS .
Desired parse : (S1 (S (ADVP RB) , (NP DT NN) (VP AUX (VP VBN (S (NP -NONE-) (VP TO (VP VB (NP PRP$ JJ NNS)))))) .))
Actual parse : (S1 (S (ADVP RB) , (NP DT NN) (VP AUX (VP VBN (S (NP -NONE-) (VP TO (VP VB (NP PRP$ JJ NNS)))))) .))

NN NN : CD NN .
Desired parse : (S1 (NP (NP NN NN) : (NP CD NN) .))
Actual parse : (S1 (NP (NP NN NN) : (NP CD NN) .))

DT JJ NNS AUX VBN -NONE- TO VB DT NN .
Desired parse : (S1 (S (NP DT JJ NNS) (VP AUX (VP VBN (S (NP -NONE-) (VP TO (VP VB (NP DT NN)))))) .))
Actual parse : (S1 (S (NP DT JJ NNS) (VP AUX (VP VBN (S (NP -NONE-) (VP TO (VP VB (NP DT NN)))))) .))

NN VBZ JJ NN NNS IN NNP NNP .
Desired parse : (S1 (S (NP NN) (VP VBZ (NP (NP JJ NN NNS) (PP IN (NP NNP NNP)))) .))
Actual parse : (S1 (S (NP NN) (VP VBZ (NP (NP JJ NN NNS) (PP IN (NP NNP NNP)))) .))

DT NN IN DT NN AUX DT NN .
Desired parse : (S1 (S (NP (NP DT NN) (PP IN (NP DT NN))) (VP AUX (NP DT NN)) .))
Actual parse : (S1 (S (NP (NP DT NN) (PP IN (NP DT NN))) (VP AUX (NP DT NN)) .))

`` PRP AUX VBG -NONE- TO AUX DT JJ NN . ''
Desired parse : (S1 (S `` (NP PRP) (VP AUX (VP VBG (S (NP -NONE-) (VP TO (VP AUX (NP DT JJ NN)))))) . ''))

Actual parse : (S1 (S `` (NP PRP) (VP AUX (VP VBG (S (NP -NONE-) (VP TO (VP AUX (NP DT JJ NN)))))) . ''))

NN AUX DT NN NN .
Desired parse : (S1 (S (NP NN) (VP AUX (NP DT NN NN)) .))
Actual parse : (S1 (S (NP NN) (VP AUX (NP DT NN NN)) .))

CC DT NN NN CC NN NN IN DT NN VBD DT NN .
Desired parse : (S1 (S CC (NP (NP DT NN NN CC NN NN) (PP IN (NP DT NN))) (VP VBD (NP DT NN)) .))
Actual parse : (S1 (S CC (NP (NP DT NN NN CC NN NN) (PP IN (NP DT NN))) (VP VBD (NP DT NN)) .))

DT JJ NN VBZ CD NNS .
Desired parse : (S1 (S (NP DT JJ NN) (VP VBZ (NP CD NNS)) .))
Actual parse : (S1 (S (NP DT JJ NN) (VP VBZ (NP CD NNS)) .))

DT NNP NNP NNP NNP VBD CD NNS TO CD .
Desired parse : (S1 (S (NP DT NNP NNP NNP NNP) (VP VBD (NP CD NNS) (PP TO (NP CD))) .))
Actual parse : (S1 (S (NP DT NNP NNP NNP NNP) (VP VBD (NP CD NNS) (PP TO (NP CD))) .))

IN NNP , JJ JJ NN NNP NNP VBD IN NNP , -NONE- VBG CD NNS IN NN .
Desired parse : (S1 (S (PP IN (NP NNP)) , (NP JJ JJ NN NNP NNP) (VP VBD (PP IN (NP NNP)) , (S (NP -NONE-) (VP VBG (NP CD NNS) (PP IN (NP NN))))) .))
Actual parse : (S1 (S (PP IN (NP NNP)) , (NP JJ JJ NN NNP NNP) (VP VBD (PP IN (NP NNP)) , (S (NP -NONE-) (VP VBG (NP CD NNS) (PP IN (NP NN))))) .))

DT JJ NN AUX JJ , '' VBD -NONE- NNP NNP IN NNP .
Desired parse : (S1 (SINV (S (NP DT JJ NN) (VP AUX (ADJP JJ))) , '' (VP VBD (S -NONE-)) (NP (NP NNP NNP) (PP IN (NP NNP))) .))
Actual parse : (S1 (SINV (S (NP DT JJ NN) (VP AUX (ADJP JJ))) , '' (VP VBD (S -NONE-)) (NP (NP NNP NNP) (PP IN (NP NNP))) .))

NNP NNP VBZ NNP NNP TO VB DT NN IN NNS .

Desired parse : (S1 (S (NP NNP NNP) (VP VBZ (S (NP NNP NNP) (VP TO (VP VB (NP DT NN) (PP IN (NP NNS)))))) .))
Actual parse : (S1 (S (NP NNP NNP) (VP VBZ (S (NP NNP NNP) (VP TO (VP VB (NP DT NN) (PP IN (NP NNS)))))) .))

DT NN POS NN AUX AUX VBG JJ NNS VB NN NNS .
Desired parse : (S1 (S (NP (NP DT NN POS) NN) (VP AUX (VP AUX (VP VBG (S (NP JJ NNS) (VP VB (NP NN NNS)))))) .))
Actual parse : (S1 (S (NP (NP DT NN POS) NN) (VP AUX (VP AUX (VP VBG (S (NP JJ NNS) (VP VB (NP NN NNS)))))) .))

IN NNP CD , DT NN VBD DT NN .
Desired parse : (S1 (S (PP IN (NP NNP CD)) , (NP DT NN) (VP VBD (NP DT NN)) .))
Actual parse : (S1 (S (PP IN (NP NNP CD)) , (NP DT NN) (VP VBD (NP DT NN)) .))

IN DT NN , NNP NNP VBZ DT NNPS .
Desired parse : (S1 (S (PP IN (NP DT NN)) , (NP NNP NNP) (VP VBZ (NP DT NNPS)) .))
Actual parse : (S1 (S (PP IN (NP DT NN)) , (NP NNP NNP) (VP VBZ (NP DT NNPS)) .))

NNP NNP VBZ DT JJ NN .
Desired parse : (S1 (S (NP NNP NNP) (VP VBZ (NP DT JJ NN)) .))
Actual parse : (S1 (S (NP NNP NNP) (VP VBZ (NP DT JJ NN)) .))

IN DT JJ NN , DT NN IN NN VBZ IN NN NN .
Desired parse : (S1 (S (PP IN (NP DT JJ NN)) , (NP (NP DT NN) (PP IN (NP NN))) (VP VBZ (PP IN (NP NN NN))) .))
Actual parse : (S1 (S (PP IN (NP DT JJ NN)) , (NP (NP DT NN) (PP IN (NP NN))) (VP VBZ (PP IN (NP NN NN))) .))

Appendix H: A Sample of Matching Parses from the WSJ Corpus Test Set (Using Lexical Semantic and Syntactic Information)

JJ NNS_00000AD000000000000000000 IN DT NN_00B00C0000A000000000000000 IN NN_A0B00D0000000000000000000 :
Desired parse : (S1 (NP (NP (NP JJ NNS) (PP IN (NP (NP DT NN) (PP IN (NP NN))))) :))
Actual parse : (S1 (NP (NP (NP JJ NNS) (PP IN (NP (NP DT NN) (PP IN (NP NN))))) :))

NNP_0000000000A00000000000000 NNS_000000000000000A0000000000 VBD DT NNS_000000AD00000000000000000 RB .
Desired parse : (S1 (S (NP NNP NNS) (VP VBD (NP DT NNS) (ADVP RB)) .))
Actual parse : (S1 (S (NP NNP NNS) (VP VBD (NP DT NNS) (ADVP RB)) .))

NN
Desired parse : (S1 (NP NN))
Actual parse : (S1 (NP NN))

NNP_000000000000000A0000000000 NNP_000000000000000A0000000000 , DT JJ NN_00C0000000A000000000000000 NN_000000000000000A0000000000 , VBZ NNS_000000000000000A0000000000 CC NNS_000000000000000A0000000000 .
Desired parse : (S1 (S (NP (NP NNP NNP) , (NP DT JJ NN NN) ,) (VP VBZ (NP NNS CC NNS)) .))
Actual parse : (S1 (S (NP (NP NNP NNP) , (NP DT JJ NN NN) ,) (VP VBZ (NP NNS CC NNS)) .))

IN NN_D0C0000000000B00000000000A , DT NN_00A000000000000A00000000D0 VBZ -NONE- TO VB JJ NNS_00B00000000A0000000000000 .
Desired parse : (S1 (S (PP IN (NP NN)) , (NP DT NN) (VP VBZ (S (NP -NONE-) (VP TO (VP VB (NP JJ NNS))))) .))
Actual parse : (S1 (S (PP IN (NP NN)) , (NP DT NN) (VP VBZ (S (NP -NONE-) (VP TO (VP VB (NP JJ NNS))))) .))

DT NN_000000A000000000000000000 RB MD VB NN_00000000000000000A0000000 NNS_000A00C0000000000A0000000 IN JJ NN_B000000000000000000000A00 NNS_000000B000000000000000A00 .
Desired parse : (S1 (S (NP DT NN) (ADVP RB) (VP MD (VP VB (NP (NP NN NNS) (PP IN (NP JJ NN NNS))))) .))
Actual parse : (S1 (S (NP DT NN) (ADVP RB) (VP MD (VP VB (NP (NP NN NNS) (PP IN (NP JJ NN NNS))))) .))

PRP_A0000000000000000000000000 AUX JJ .
Desired parse : (S1 (S (NP PRP) (VP AUX (ADJP JJ)) .))
Actual parse : (S1 (S (NP PRP) (VP AUX (ADJP JJ)) .))

NNS_00A00000000000000D0000000 : NN_00CB00D000000000000A00000 CD NNS_C000000000000000A0000000 .
Desired parse : (S1 (NP (NP NNS) : (NP (NP NN) (NP CD NNS)) .))
Actual parse : (S1 (NP (NP NNS) : (NP (NP NN) (NP CD NNS)) .))

DT JJ NNS_000000A000000000000000000 AUX VBN -NONE- TO VB DT NN_B000000C00000D0000000000A .
Desired parse : (S1 (S (NP DT JJ NNS) (VP AUX (VP VBN (S (NP -NONE-) (VP TO (VP VB (NP DT NN)))))) .))
Actual parse : (S1 (S (NP DT JJ NNS) (VP AUX (VP VBN (S (NP -NONE-) (VP TO (VP VB (NP DT NN)))))) .))

DT JJ NNS_B000000C0000000000000000A AUX AUX JJ -NONE- IN NNP_000000A000000000000000000 NNP_00000000000000A0000000000 POS NN_A0000000000000000000000000 .
Desired parse : (S1 (S (NP DT JJ NNS) (VP AUX (VP AUX (VP JJ (NP -NONE-) (PP IN (NP (NP NNP NNP POS) NN))))) .))
Actual parse : (S1 (S (NP DT JJ NNS) (VP AUX (VP AUX (VP JJ (NP -NONE-) (PP IN (NP (NP NNP NNP POS) NN))))) .))

NN_0000000000A00000000000000 AUX DT NN_B0A000D0000000C0000000000 NN_0000000000B000A0000000000 .
Desired parse : (S1 (S (NP NN) (VP AUX (NP DT NN NN)) .))
Actual parse : (S1 (S (NP NN) (VP AUX (NP DT NN NN)) .))

JJ NNS_00A000000C000A00000000000 RB VBP CD NN_B00000A000000000000000000 IN DT NN_0000000A00000000000000000B .
Desired parse : (S1 (S (NP JJ NNS) (ADVP RB) (VP VBP (NP (NP CD NN) (PP IN (NP DT NN)))) .))
Actual parse : (S1 (S (NP JJ NNS) (ADVP RB) (VP VBP (NP (NP CD NN) (PP IN (NP DT NN)))) .))

NNP_000000A000000000000000000 NNP_00000000000000A0000000000 AUX JJ NN_A0000B000000000000000000 NN_00000000000000A0000000000 IN DT NNP_0000000000A00000000000000 NNP_0000000000A00000000000000 .
Desired parse : (S1 (S (NP NNP NNP) (VP AUX (NP (NP JJ NN NN) (PP IN (NP DT NNP NNP)))) .))
Actual parse : (S1 (S (NP NNP NNP) (VP AUX (NP (NP JJ NN NN) (PP IN (NP DT NNP NNP)))) .))

NNP_000000A000000000000000000 NNP_00000000000000A0000000000 AUX DT JJ NN_00000000000000A0000000000 IN DT NNP_0000000000A00000000000000 NNP_0000000000A00000000000000 NNP_0000000000A00000000000000 .
Desired parse : (S1 (S (NP NNP NNP) (VP AUX (NP (NP DT JJ NN) (PP IN (NP DT NNP NNP NNP)))) .))
Actual parse : (S1 (S (NP NNP NNP) (VP AUX (NP (NP DT JJ NN) (PP IN (NP DT NNP NNP NNP)))) .))

PRP_00000000000000A0000000000 VBP DT NN_00B00AC000000000000000000 .
Desired parse : (S1 (S (NP PRP) (VP VBP (NP DT NN)) .))
Actual parse : (S1 (S (NP PRP) (VP VBP (NP DT NN)) .))

PRP_00000000000000A0000000000 VBZ -NONE- DT JJ NN_00000000000000A0000000000 .
Desired parse : (S1 (S (NP PRP) (VP VBZ (S (NP -NONE-) (NP DT JJ NN))) .))
Actual parse : (S1 (S (NP PRP) (VP VBZ (S (NP -NONE-) (NP DT JJ NN))) .))

`` PRP_00000000000000A0000000000 AUX JJ JJ NNS_00000000000000A0000000000 .
Desired parse : (S1 (S `` (NP PRP) (VP AUX (NP JJ JJ NNS)) .))
Actual parse : (S1 (S `` (NP PRP) (VP AUX (NP JJ JJ NNS)) .))

NNS_00000000000C0000000A00D00 IN NNS_00B00DA0000000000000B0000 AUX IN NN_A000000000000000000000C00 .
Desired parse : (S1 (S (NP (NP NNS) (PP IN (NP NNS))) (VP AUX (PP IN (NP NN))) .))
Actual parse : (S1 (S (NP (NP NNS) (PP IN (NP NNS))) (VP AUX (PP IN (NP NN))) .))

JJ JJ NNS_00000000000000A0000000000 VBP JJ .
Desired parse : (S1 (S (NP JJ JJ NNS) (VP VBP (ADJP JJ)) .))
Actual parse : (S1 (S (NP JJ JJ NNS) (VP VBP (ADJP JJ)) .))

NNP_000000A000000000000000000 NNP_00000000000000A0000000000 VBZ DT JJ NN_000B0A0000000000000000000 .
Desired parse : (S1 (S (NP NNP NNP) (VP VBZ (NP DT JJ NN)) .))
Actual parse : (S1 (S (NP NNP NNP) (VP VBZ (NP DT JJ NN)) .))

DT JJ NN_B0000000000000A0000000000 AUX JJ .
Desired parse : (S1 (S (NP DT JJ NN) (VP AUX (ADJP JJ)) .))
Actual parse : (S1 (S (NP DT JJ NN) (VP AUX (ADJP JJ)) .))

DT JJ VBG IN NN_D00B0A0000000000000000000 .
Desired parse : (S1 (S (NP DT JJ) (VP VBG (PP IN (NP NN))) .))
Actual parse : (S1 (S (NP DT JJ) (VP VBG (PP IN (NP NN))) .))

JJ JJ JJ NN_A0D0000000B00000000000000 NNS_00A0000000000000000000000 VBD IN JJ NN_00000B00000A0000000000000 .
Desired parse : (S1 (S (NP JJ JJ JJ NN NNS) (VP VBD (PP IN (NP JJ NN))) .))
Actual parse : (S1 (S (NP JJ JJ JJ NN NNS) (VP VBD (PP IN (NP JJ NN))) .))

NNP_00000000000000A0000000000 NNP_00000000000000A0000000000 NNP_00000000000000A0000000000 , CD , AUX VBN -NONE- DT JJ NN_B00A00000000000000000000 NN_00000000000000A0000000000 , IN NNS_A00B000000000000000000B00 IN NN_A0000B0000000000000000000 CC NN_00000B0000A00000000000000 NN_000000000000000000A000000 .
Desired parse : (S1 (S (NP (NP NNP NNP NNP) , (NP CD) ,) (VP AUX (VP VBN (S (NP -NONE-) (NP (NP DT JJ NN NN) , (PP IN (NP (NP NNS) (PP IN (NP NN CC NN NN)))))))) .))

Actual parse : (S1 (S (NP (NP NNP NNP NNP) , (NP CD) ,) (VP AUX (VP VBN (S (NP -NONE-) (NP (NP DT JJ NN NN) , (PP IN (NP (NP NNS) (PP IN (NP NN CC NN NN)))))))) .))

DT NN_A00C000000D00000000000000 AUX JJ .
Desired parse : (S1 (S (NP DT NN) (VP AUX (ADJP JJ)) .))
Actual parse : (S1 (S (NP DT NN) (VP AUX (ADJP JJ)) .))

DT NN_B00B00A000000000000000000 VBZ NNP_0000000000000000000000000A CD , CD .
Desired parse : (S1 (S (NP DT NN) (VP VBZ (NP NNP CD , CD)) .))
Actual parse : (S1 (S (NP DT NN) (VP VBZ (NP NNP CD , CD)) .))

Appendix I: A Sample of Mismatching Parses from the WSJ Corpus Test Set (using Syntactic Information Only)

DT NN VBD JJ NNS IN NN IN NNP CC NNP , -NONE- VBG IN DT NN IN NN NN IN DT JJ NN IN NNP MD AUX JJ TO CD NN , DT JJ NN VBD -NONE- -NONE- .
Desired parse : (S1 (S (S (NP DT NN) (VP VBD (NP (NP JJ NNS) (PP IN (NP (NP NN) (PP IN (NP NNP CC NNP))))) , (S (NP -NONE-) (VP VBG (SBAR IN (S (NP (NP DT NN) (PP IN (NP (NP NN NN) (PP IN (NP DT JJ NN)))) (PP IN (NP NNP))) (VP MD (VP AUX (ADJP JJ (PP TO (NP CD NN))))))))))) , (NP DT JJ NN) (VP VBD (SBAR -NONE- (S -NONE-))) .))
Actual parse : (S1 (S (S (NP DT NN) (VP VBD (NP (NP JJ NNS) (PP IN (NP NN))) (PP IN (NP NNP CC NNP)))) , (NP -NONE-) (VP VBG (PP IN (NP (NP DT NN) (PP IN (NP (NP NN NN) (PP IN (NP (NP DT JJ NN) (PP IN (NP NNP)))))))) (VP MD (VP AUX (NP (NP JJ) (PP TO (NP CD NN) ,) (S (NP DT JJ NN) (VP VBD (SBAR -NONE- (S -NONE-)))))))) .))

-NONE- VBN -NONE- IN DT NN IN NNP NNP , DT NN IN DT NN WP -NONE- VBD IN CD .
Desired parse : (S1 (S (NP -NONE-) (VP VBN (NP -NONE-) (PP IN (NP (NP DT NN) (PP IN (NP (NP NNP NNP) , (NP (NP DT NN) (PP IN (NP DT NN)) (SBAR (WHNP WP) (S (NP -NONE-) (VP VBD (PP IN (NP CD))))))))))) .))
Actual parse : (S1 (S (NP -NONE-) (VP VBN (NP -NONE-) (PP IN (NP (NP DT NN) (PP IN (NP NNP NNP)))) , (NP (NP DT NN) (PP IN (NP (NP DT NN) (SBAR (WHNP WP) (S (NP -NONE-) (VP VBD (PP IN (NP CD))))))) .)))

: NNP NNP .
Desired parse : (S1 (NP : NNP NNP .))
Actual parse : (S1 (SBAR (SBAR (S1 (PRN : (NP NNP NNP)))) .))

NNP NNP , DT NNP , NNP , NN , VBZ -NONE- NNS AUX JJ IN IN DT CD NN NN VBZ TO DT JJ NN NN CC TO JJ VBG NNS .
Desired parse : (S1 (S (NP (NP NNP NNP) , (NP DT (NAC NNP , NNP ,) NN) ,) (VP VBZ (SBAR -NONE- (S (NP NNS) (VP AUX (ADJP JJ) (PP IN (SBAR IN (S (NP DT (ADJP CD NN) NN) (VP VBZ (PP (PP TO (NP DT JJ NN NN)) CC (PP TO (NP JJ VBG NNS)))))))))))) .))

Actual parse : (S1 (S (NP NNP NNP) , (NP (NP DT NNP) , (NP (NP NNP) , NN) ,) (VP VBZ (S (NP -NONE- NNS) (VP AUX (S (NP (NP JJ) (PP IN (PP IN (NP DT CD NN NN)))) (VP VBZ (VP TO (S (NP (NP DT JJ NN NN) CC (PP TO (NP JJ))) (VP VBG NNS)))))))) .))

DT NNS VBP IN PRP MD VB CD NNS IN NNS VBP DT NN JJR .
Desired parse : (S1 (S (NP DT NNS) (VP VBP (SBAR IN (S (NP PRP) (VP MD (VP VB (NP CD NNS) (SBAR IN (S (NP NNS) (VP VBP (NP DT NN JJR)))))))) .))
Actual parse : (S1 (S (NP DT NNS) (VP VBP (SBAR IN (S (NP PRP) (VP MD (VP VB (S (NP (NP CD NNS) (PP IN (NP NNS))) (VP VBP (NP DT NN JJR)))))))) .))

RB , NNP NNP RB AUX AUXG VBN , IN DT NN DT NN CC CD DT NN .
Desired parse : (S1 (S (ADVP RB) , (NP NNP NNP) (ADVP RB) (VP AUX (VP AUXG (VP VBN , (PP IN (NP (NP (NP DT NN) (NP DT NN)) CC (NP (NP CD) (NP DT NN))))))) .))
Actual parse : (S1 (S (ADVP RB) , (NP NNP NNP) (VP RB (VP AUX (ADVP AUXG) (VP VBN , (PP IN (NP DT NN DT) NN) CC (NP CD DT NN)))) .))

NNP NNP , NN IN NN IN DT JJ NN NN NN , AUX AUX VBG IN NN NNS VBG NN IN DT NNP NN .
Desired parse : (S1 (S (NP (NP NNP NNP) , (NP (NP NN) (PP IN (NP NN)) (PP IN (NP DT JJ NN NN NN))) ,) (VP AUX (VP AUX (VP VBG (PP IN (NP (NP NN NNS) (VP VBG (NP NN)))) (PP IN (NP DT NNP NN))))) .))
Actual parse : (S1 (NP (NP NNP NNP) (SBAR , (S (NP NN) (PP IN (NP NN)) (PP IN (NP DT JJ NN NN NN)) , (VP AUX (VP AUX (VP VBG (PP IN (S (NP NN NNS) (VP VBG (NP (NP NN) (PP IN (NP DT NNP NN)))))))) .))))

NNS IN DT CD NNS VBN -NONE- IN NNP POS VBD RB $ CD CD -NONE- TO DT NN IN $ CD CD -NONE- IN -NONE- VBG DT JJ NN .
Desired parse : (S1 (S (NP (NP NNS) (PP IN (NP (NP DT CD NNS) (VP VBN (NP -NONE-) (PP IN (NP NNP POS)))))) (VP VBD (NP (QP RB $ CD CD) -NONE-) (PP TO (NP (NP DT NN) (PP IN (NP (QP $ CD CD) -NONE-)))) (PP IN (S (NP -NONE-) (VP VBG (NP DT JJ NN))))) .))
Actual parse : (S1 (S (NP (NP NNS) (PP IN (NP DT CD NNS))) (VP VBN (NP (NP -NONE-) (FRAG IN NNP)) (S POS (VP VBD)) (ADVP RB) (SBAR $ (S (NP (NP (NP -NONE-) (PP IN (NP -NONE-))) (NP (NP (NP DT NN) (PP TO (NP -NONE-))) (SBAR (NP (NP CD) (NP CD) (PP IN)) (S (QP $ CD CD))))) (VP VBG (NP DT JJ NN))))) .))

IN CD NNS , JJ NNS AUX VBN IN RB -NONE- TO VB DT VBG NNS IN NNS IN NN IN NNP JJ NNS .
Desired parse : (S1 (S (PP IN (NP CD NNS)) , (NP JJ NNS) (VP AUX (VP VBN (PP IN (NP RB)) (S (NP -NONE-) (VP TO (VP VB (NP (NP DT VBG NNS) (PP IN (NP (NP NNS) (PP IN (NP NN)) (PP IN (NP NNP JJ NNS))))))))))) .))
Actual parse : (S1 (SBAR (SBAR (S (PP IN (NP CD NNS)) , (S (S (NP JJ NNS) (VP AUX (VP VBN (PP IN RB (NP -NONE-))))) (VP TO (VP VB (NP DT (UCP VBG NNS) (PP IN (NP (NP NNS) (PP IN (NP (NP NN) (PP IN (NP NNP JJ NNS))))))))))) .))

NN NNS NNS RB VBD IN NNP POS NNS IN NNS VBD RB JJ .
Desired parse : (S1 (S (NP NN NNS NNS) (ADVP RB) (VP VBD (PP IN (NP (NP NNP POS) NNS)) (SBAR IN (S (NP NNS) (VP VBD (ADJP RB JJ))))) .))
Actual parse : (S1 (S (NP NN NNS NNS) (ADVP RB) (VP VBD (SBAR IN (S (NP (NP (NP NNP POS) NNS) (PP IN (NP NNS))) (VP VBD RB (ADJP JJ))))) .))

RB , NNP MD RB AUX -NONE- TO VB TO DT NNP NNP NNP IN PRP MD VB CC VB JJ NNS .
Desired parse : (S1 (S (ADVP RB) , (NP NNP) (VP MD (ADVP RB) (VP AUX (S (NP -NONE-) (VP TO (VP VB (PP TO (NP DT NNP NNP NNP)) (SBAR IN (S (NP PRP) (VP MD (VP VB CC VB (NP JJ NNS))))))))))) .))
Actual parse : (S1 (S (ADVP RB) , (NP NNP) (VP MD (ADVP RB) (VP AUX (S (NP -NONE-) (VP TO (VP VB (VP TO (S (NP (NP DT NNP NNP NNP) (PP IN (NP PRP))) (VP MD VB CC (VP VB (NP JJ NNS))))))))))) .))

NNS VBP -NONE- NNP NNP MD AUX JJR IN CD NNS RB IN DT NN .
Desired parse : (S1 (S (NP NNS) (VP VBP (SBAR -NONE- (S (NP NNP NNP) (VP MD (VP AUX (ADJP (ADJP JJR) (PP IN (NP (NP CD NNS) (ADVP RB (PP IN (NP DT NN))))))))))) .))
Actual parse : (S1 (S (NP NNS) (VP VBP (SBAR -NONE- (S (NP NNP NNP) (VP MD (VP AUX (NP (NP JJR) (PP IN (NP (NP CD NNS) RB))) (PP IN (NP DT NN))))))) .))

IN JJ NNS IN NNP NNP , NN NNP NNP NNP IN NNP , NNP CC NNP VBD , `` PRP AUX RB VB NNS VBG '' IN DT NN IN NN NN .
Desired parse : (S1 (S (PP IN (NP (NP JJ NNS) (PP IN (NP NNP NNP)))) , (NP (NP NN NNP NNP NNP) (PP IN (NP NNP , NNP CC NNP))) (VP VBD , `` (S (NP PRP) (VP

AUX RB (VP VB (S (NP NNS) (VP VBG '' (PP IN (NP (NP DT NN) (PP IN (NP NN NN))))))))))) .))
Actual parse : (S1 (S (PP IN (NP (NP JJ NNS) (PP IN (NP NNP))) (NP (NP NNP) , (NP NN NNP NNP NNP) IN (NP NNP))) , (NP NNP CC NNP) (VP VBD (S (NP , `` PRP) (VP AUX (ADVP RB) (VP VB (ADJP NNS VBG '' (PP IN (NP (NP DT NN) (PP IN (NP NN NN)))))))))) .))

NNP NNP VBD PRP IN DT JJ NN -NONE- DT NN AUX VBN -NONE- .
Desired parse : (S1 (S (NP NNP NNP) (VP VBD (NP PRP) (PP IN (NP (NP DT JJ NN) (SBAR (WHADVP -NONE-) (S (NP DT NN) (VP AUX (VP VBN (NP -NONE-)))))))) .))
Actual parse : (S1 (S (NP NNP NNP) (VP VBD (NP PRP) (PP IN (NP DT JJ NN) (S (NP (NP -NONE-) (NP DT NN)) (VP AUX (VP VBN (NP -NONE-)))))) .))

NNP NNP RB AUX VBN NN IN NNP POS NN NN , WDT -NONE- VBZ RB $ CD CD -NONE- IN NN CC NN NNS .
Desired parse : (S1 (S (NP NNP NNP) (ADVP RB) (VP AUX (VP VBN (NP (NP NN) (PP IN (NP (NP (NP NNP POS) NN NN) , (SBAR (WHNP WDT) (S (NP -NONE-) (VP VBZ (NP (NP (QP RB $ CD CD) -NONE-) (PP IN (NP NN CC NN NNS))))))))))) .))
Actual parse : (S1 (S (NP NNP NNP) (ADVP RB) (VP AUX (VP VBN (NP (NP NN) (PP IN (NP NNP POS NN NN))) , (SBAR (WHNP WDT) (S (NP -NONE-) (VP VBZ (ADVP RB) (QP $ CD CD) -NONE-)) (PP IN (NP NN CC NN NNS))))) .))

DT JJ NN POS NN -NONE- TO VB NNP NNP IN $ CD CD -NONE- AUX VBN -NONE- IN DT NNP NN NN .
Desired parse : (S1 (S (NP (NP DT JJ NN POS) NN (S (NP -NONE-) (VP TO (VP VB (NP NNP NNP) (PP IN (NP (QP $ CD CD) -NONE-)))))) (VP AUX (VP VBN (NP -NONE-) (PP IN (NP DT NNP NN NN)))) .))
Actual parse : (S1 (SBAR (SBAR (S1 (NP (NP -NONE-) NN (S (NP DT JJ NN POS) (VP TO (VP VB (S (NP (NP NNP NNP IN) (QP $ CD CD) (NP -NONE-)) (VP AUX (VP VBN (NP -NONE-) (PP IN (NP DT NNP NN NN))))))))))) .))

NNP NNS AUX -NONE- TO VB RB NN , IN NNP NNP NNP NNP NNP IN NNP VBD : `` PRP MD AUX RB JJ -NONE- -NONE- TO VB NN . ''
Desired parse : (S1 (S (NP NNP NNS) (VP AUX (S (NP -NONE-) (VP TO (VP VB (NP RB NN)))) , (SBAR IN (S (NP (NP NNP NNP NNP NNP NNP) (PP IN (NP NNP))) (VP VBD : `` (S (NP PRP) (VP MD (VP AUX (ADJP RB JJ (SBAR (WHNP -NONE-) (S (NP -NONE-) (VP TO (VP VB (NP NN)))))))))))) . ''))

Actual parse : (S1 (NP NNP NNS (VP AUX (S (NP -NONE-) (VP TO (VP VB (NP RB NN) , (SBAR IN (S (NP (NP (NP NNP) (PP IN (NP (NP NNP) (VP VBD : `` PRP)))) (NP NNP NNP NNP NNP)) (VP MD (VP AUX RB (NP JJ) (S (NP (NP -NONE-) (NP -NONE-)) (VP TO (VP VB (NP NN)))) .)))))))) ''))

NNP NNP RB VBD IN DT NN IN DT NNP JJ NN NN , -NONE- VBG CD TO CD .
Desired parse : (S1 (S (NP NNP NNP) (ADVP RB) (VP VBD (PP IN (NP (NP DT NN) (PP IN (NP DT NNP JJ NN NN)))) , (S (NP -NONE-) (VP VBG (NP CD) (PP TO (NP CD))))) .))
Actual parse : (S1 (S (NP NNP NNP) (ADVP RB) (VP VBD (PP IN (NP DT NN)) (PP IN (NP DT NNP JJ NN NN)) , (S (NP -NONE-) (VP VBG (NP CD) (PP TO (NP CD))))) .))

IN PRP AUX RB RB VBN -NONE- TO VB NNS IN DT NN DT NN , NN PRP VBD CD NNS IN JJ NNS .
Desired parse : (S1 (S (SBAR IN (S (NP PRP) (VP AUX RB (ADVP RB) (VP VBN (S (NP -NONE-) (VP TO (VP VB (NP (NP NNS) (PP IN (NP DT NN))) (NP DT NN)))))))) , (NP NN) (NP PRP) (VP VBD (NP (NP CD NNS) (PP IN (NP JJ NNS)))) .))
Actual parse : (S1 (S (SBAR IN (S (NP PRP) (VP AUX (VP RB (ADVP RB) (VP VBN (S (NP -NONE-) (VP TO (VP VB (NP NNS) (PP IN (NP DT NN DT NN)))))))))) , (NP NN PRP) (VP VBD (NP CD NNS) (PP IN (NP JJ NNS))) .))

Appendix J: A Sample of Mismatching Parses from the WSJ Corpus Test Set (Using Lexical Semantic and Syntactic Information)

RB , NNP_000000A000000000000000000 NNP_000000000000000A0000000000 RB AUX AUXG VBN , IN DT NN_C00000A000D0000000000000B DT NN_000000BC0000000000000000A CC CD DT NN_000000B000000000000000000A .

Desired parse : (S1 (S (ADVP RB) , (NP NNP NNP) (ADVP RB) (VP AUX (VP AUXG (VP VBN , (PP IN (NP (NP (NP DT NN) (NP DT NN)) CC (NP (NP CD) (NP DT NN))))))) .))

Actual parse : (S1 (S (ADVP RB) , (NP NNP NNP) (ADVP RB) (VP AUX (VP (FRAG AUXG) (VP VBN , (PP IN (NP (NP DT NN DT NN CC) (NP CD DT NN)))))) .))

NNP_000000000000000A0000000000 NNP_000000000000000A0000000000 , NN_000000000000000A0000000000 IN NN_0000000000A000000000000000 IN DT JJ NN_000000000C000B000C00000A0 NN_00B000000BA000000000000000 NN_0000000000B000A0000000000 , AUX AUX VBG IN NN_000000000C000B000C00000A0 NNS_00B000000BA000000000000000 VBG NN_000000000000000000000000A0 IN DT NNP_0000000000A000000000000000 NN_00A00C00000000D0B00000000 .

Desired parse : (S1 (S (NP (NP NNP NNP) , (NP (NP NN) (PP IN (NP NN)) (PP IN (NP DT JJ NN NN NN))) ,) (VP AUX (VP AUX (VP VBG (PP IN (NP (NP NN NNS) (VP VBG (NP NN)))) (PP IN (NP DT NNP NN))))) .))

Actual parse : (S1 (S (NP (NP NNP NNP) , (NP (NP NN) (PP IN (NP NN (PP IN (NP DT JJ NN NN)) NN))) ,) (VP AUX (VP AUX (VP VBG (SBAR IN (S (NP NN NNS) (VP VBG (NP (NP NN) (PP IN (NP DT NNP NN)))))))) .))

IN CD NNS_000000000000000000000000A , JJ NNS_0000000000B00CA00D0000000 AUX VBN IN RB -NONE- TO VB DT VBG NNS_000000000000000A0000000000 IN NNS_00000AC000000000000000000 IN NN_D00000A0000000000B0000000 IN NNP_000000000000A0000000000000 JJ NNS_000000000000000A0B00C0000 .

Desired parse : (S1 (S (PP IN (NP CD NNS)) , (NP JJ NNS) (VP AUX (VP VBN (PP IN (NP RB)) (S (NP -NONE-) (VP TO (VP VB (NP (NP DT VBG NNS) (PP IN (NP (NP NNS) (PP IN (NP NN)) (PP IN (NP NNP JJ NNS)))))))))) .))

Actual parse : (S1 (S (NP (PP IN (NP (NP CD NNS) , (NP JJ NNS) (VP AUX)))) (VP VBN (PP IN (PRN RB (S (NP -NONE-) (VP TO (VP VB (S (NP DT) (VP VBG (NP (NP NNS) (PP IN (NP (NP NNS) (PP IN (NP (NP (NP NN) (PP IN (NP NNP JJ NNS)))))))))))))))) .))

RB , NNP_0000000000A00000000000000 MD RB AUX -NONE- TO VB TO DT NNP_0000000000A00000000000000 NNP_0000000000A00000000000000 NNP_0000000000A00000000000000 IN PRP_0000000000A00000000000000 MD VB CC VB JJ NNS_00A000B000000C000000000C0 .
Desired parse : (S1 (S (ADVP RB) , (NP NNP) (VP MD (ADVP RB) (VP AUX (S (NP -NONE-) (VP TO (VP VB (PP TO (NP DT NNP NNP NNP)) (SBAR IN (S (NP PRP) (VP MD (VP VB CC VB (NP JJ NNS)))))))))) .))
Actual parse : (S1 (S (ADVP RB) , (NP NNP) (VP MD (VP (ADVP RB) (VP AUX (S (NP -NONE-) (VP TO (VP VB (PP TO (NP (S (NP DT NNP NNP) (NP NNP)) (UCP IN (S (NP PRP) (VP MD (VP VB CC (VP VB (NP JJ NNS)))))))))))))) .))

NNS_00000000000000A0000000000 VBP -NONE- NNP_0000000000A00000000000000 NNP_0000000000A00000000000000 MD AUX JJR IN CD NNS_000000000000000000000000A RB IN DT NN_B000000B000000D00000A0000 .
Desired parse : (S1 (S (NP NNS) (VP VBP (SBAR -NONE- (S (NP NNP NNP) (VP MD (VP AUX (ADJP (ADJP JJR) (PP IN (NP (NP CD NNS) (ADVP RB (PP IN (NP DT NN)))))))))) .))
Actual parse : (S1 (S (NP NNS) (VP VBP (SBAR (WHNP -NONE-) (S (NP NNP NNP) (ADVP MD) (VP AUX (PRN JJR (PP IN (NP CD NNS))) (INTJ (UCP RB) (PP IN (NP DT NN))))))) .))

IN JJ NNS_00000AB000000000000000000 IN NNP_000000000000A0000000000000 NNP_000000000000A0000000000000 , NN_000000000000000A0000000000 NNP_000000000000000A0000000000 NNP_000000000000000A0000000000 NNP_000000000000000A0000000000 IN NNP_0000000000A00000000000000 , NNP_0000000000A00000000000000 CC NNP_0000000000A00000000000000 VBD , `` PRP_000000000000000A0000000000 AUX RB VB NNS_00000000000000000A0000000 VBG '' IN DT NN_000000BC0000000A000000000 IN NN_000000000000A0000B0000000 NN_00000BA00000000000C000C00 .

Desired parse : (S1 (S (PP IN (NP (NP JJ NNS) (PP IN (NP NNP NNP)))) , (NP (NP NN NNP NNP NNP) (PP IN (NP NNP , NNP CC NNP))) (VP VBD , `` (S (NP PRP) (VP AUX RB (VP VB (S (NP NNS) (VP VBG '' (PP IN (NP (NP DT NN) (PP IN (NP NN NN))))))))))) .))
Actual parse : (S1 (SBAR (NP IN (PRN JJ (NP NNS) IN (NP (NP NNP NNP) , (NP NN (NP NNP NNP (NP NNP IN (NP NNP)))))) , (NP NNP) CC (S (NP NNP) (VP VBD , `` (S (NP PRP) (VP AUX (VP (ADVP RB) (VP VB (S (NP NNS) (VP VBG '' (PP IN (NP (NP DT NN) (PP IN (NP NN NN)))))))))))) .)))

NNP_00000000000000A0000000000 NNP_00000000000000A0000000000 VBD PRP_000000000000000000A0000000 IN DT JJ NN_0000000000D0000000000000A -NONE- DT NN_00C000000000000000A0000000 AUX VBN -NONE- .
Desired parse : (S1 (S (NP NNP NNP) (VP VBD (NP PRP) (PP IN (NP (NP DT JJ NN) (SBAR (WHADVP -NONE-) (S (NP DT NN) (VP AUX (VP VBN (NP -NONE-)))))))) .))
Actual parse : (S1 (S (NP NNP NNP) (VP VBD (SBAR (WHNP (NP PRP) (PP IN (NP (ADJP DT JJ) NN) (WHADVP -NONE-))) (S (NP DT NN) (VP AUX (VP VBN (NP -NONE-)))))) .))

NNP_0000000000A00000000000000 NNP_0000000000A00000000000000 RB AUX VBN NN_000B0A00000000000D0000000 IN NNP_0000000000A00000000000000 POS NN_0000000000A00000000000000 NN_00000D0000C00000000AB0000 , WDT -NONE- VBZ RB $ CD CD -NONE- IN NN_000B0B0000D000000A0000000 CC NN_B000000000000000000000A00 NNS_000B000000000000A0000000 .
Desired parse : (S1 (S (NP NNP NNP) (ADVP RB) (VP AUX (VP VBN (NP (NP NN) (PP IN (NP (NP (NP NNP POS) NN NN) , (SBAR (WHNP WDT) (S (NP -NONE-) (VP VBZ (NP (NP (QP RB $ CD CD) -NONE-) (PP IN (NP NN CC NN NNS)))))))))) .))
Actual parse : (S1 (S (NP NNP NNP) (ADVP RB) (VP AUX (VP VBN NN (PP IN (NP (NP (NP NNP POS NN) NN) , (SBAR (WHNP WDT) (S (NP -NONE-) (VP VBZ (PRN RB (S (QP $ CD CD) -NONE- (PP IN (NP (NP NN) CC (NP NN NNS)))))))))) .))

DT JJ NN_00000A00B0C00000000000000 POS NN_000000A000000000000000000 -NONE- TO VB NNP_0000000000A00000000000000 NNP_0000000000A00000000000000 IN $ CD CD -NONE- AUX VBN -NONE- IN DT NNP_00000000000A0000000000000 NN_A000000000000000000000000 NN_0000000000A00000000000C00 .

Desired parse : (S1 (S (NP (NP DT JJ NN POS) NN (S (NP -NONE-) (VP TO (VP VB (NP NNP NNP) (PP IN (NP (QP $ CD CD) -NONE-)))))) (VP AUX (VP VBN (NP -NONE-) (PP IN (NP DT NNP NN NN)))) .))
Actual parse : (S1 (SBAR (NP (NP DT JJ NN POS) NN) (S -NONE- (VP TO (VP VB (NP NNP NNP)))) (PRN IN (SBAR (NP -NONE-) (S (NP (NP $) (NP CD CD)) (VP AUX (VP VBN (S (NP -NONE-) (PP IN (NP DT NNP NN NN)))))))) .))

NNP_0000000000A00000000000000 NNP_000000A000000000000000000 RB VBD IN DT NN_000B0A00000000000D0000000 IN DT NNP_00000000000A0000000000000 JJ NN_00A000B000000000000000000 NN_0000000000A00000000000C00 , -NONE- VBG CD TO CD .
Desired parse : (S1 (S (NP NNP NNP) (ADVP RB) (VP VBD (PP IN (NP (NP DT NN) (PP IN (NP DT NNP JJ NN NN)))) , (S (NP -NONE-) (VP VBG (NP CD) (PP TO (NP CD))))) .))
Actual parse : (S1 (S (NP NNP NNP) (ADVP RB) (VP VBD (PP IN (NP (NP DT NN) (PP IN (ADJP DT NNP JJ NN) NN))) , (S (NP -NONE-) (VP (VP VBG (NP CD)) (PP TO CD)))) .))

NNP_000000A000000000000000000 : JJ NNS_B00000A000000000000000000C .
Desired parse : (S1 (NP (NP NNP) : (NP JJ NNS) .))
Actual parse : (S1 (NP NNP : (S1 (NP JJ NNS) .)))

PRP$ JJ NN_B0A0C000000000C0000000000 .
Desired parse : (S1 (FRAG (NP PRP$ JJ NN) .))
Actual parse : (S1 (SBAR (NP (NP PRP$ JJ NN) .)))

DT NN_000000000000000A0000000000 VBD -NONE- DT NNP_0000000000A00000000000000 NNS_C00D00A000000000000000000 IN NN_00000000000000000A0000000 CC NN_A00B000000000000000000000 CC JJ NNS_A00B0000000000000B0000000 .
Desired parse : (S1 (S (NP DT NN) (VP VBD (SBAR -NONE- (S (NP DT NNP) (VP NNS (PP IN (NP (NP NN CC NN) CC (NP JJ NNS))))))) .))
Actual parse : (S1 (S (NP DT NN) (VP VBD (NP -NONE- (NP DT NNP NNS)) (PP IN (NP NN (NP CC NN) (NP CC JJ NNS)))) .))

IN CD , NNP_000000000000000A0000000000 RB VBD -NONE- RB AUXG VBD -NONE- DT NN_A000000000000000000000B00 AUX JJ .

Desired parse : (S1 (S (PP IN (NP CD)) , (NP NNP) (ADVP RB) (VP VBD (S (NP -NONE-) (VP (ADVP RB) AUXG (VP VBD (SBAR -NONE- (S (NP DT NN) (VP AUX (ADJP JJ)))))))) .))
Actual parse : (S1 (S IN (NP CD , NNP) (ADVP RB) (VP VBD (SBAR -NONE- (SBAR (ADVP RB) (S (ADVP AUXG) (VP VBD (SBAR -NONE- (S (NP DT NN) (VP AUX (ADJP JJ))))))))) .))

NNP_0000000000A00000000000000 NNP_0000000000A00000000000000 NNP_0000000000A00000000000000 : $ CD CD -NONE- IN CD CD NN_00000000000000000000A0000 NN_0000000AC0000000000000000 NN_0000000000C000000A0000000 NNS_000000A0000000000A0000000 JJ NNP_000000000000000000000000A CD , CD , VBN -NONE- IN CD TO VB CD NN_00000000000000000000A0000 .
Desired parse : (S1 (NP (NP NNP NNP NNP) : (NP (NP (QP $ CD CD) -NONE-) (PP IN (NP (NP (QP CD CD) NN NN NN NNS) (ADJP JJ (NP NNP CD , CD)) , (VP VBN (NP -NONE-) (PP IN (NP CD)) (S (VP TO (VP VB (NP CD NN)))))))) .))
Actual parse : (S1 (S (NP (UCP NNP (NP NNP NNP) : (SBAR (QP $ CD CD) (S -NONE-))) (PRN IN (INTJ (UCP CD , (NP (NP CD) , VBN (NP -NONE-)) (SBAR IN (S (NP CD) (VP TO (VP VB (NP (ADJP CD) NN)))))) (NP (NP JJ NNP) (UCP (QP CD CD NN) (NP NN NN NNS)))))) .))

JJ NNS_00000B0000A00000000000000 VBN -NONE-
Desired parse : (S1 (NP (NP JJ NNS) (VP VBN (NP -NONE-))))
Actual parse : (S1 (NP JJ NNS VBN) -NONE-)

NNS_000000000000000A0000000000 VBD IN DT JJ NN_A000000000000000000B000000 IN NN_000B0A00000000000D0000000 NNS_000D000000000000A000000B RB AUX JJ , NNP_000000A000000000000000000 NNP_000000000000000A0000000000 VBD -NONE- -NONE- .
Desired parse : (S1 (S (S (NP NNS) (VP VBD (SBAR IN (S (NP (NP DT JJ NN) (PP IN (NP NN NNS))) (ADVP RB) (VP AUX (ADJP JJ)))))) , (NP NNP NNP) (VP VBD (SBAR -NONE- (S -NONE-))) .))
Actual parse : (S1 (S (NP NNS) (VP VBD (SBAR IN (S (NP (NP DT JJ NN) (PP IN (NP NN NNS))) (ADVP RB) (VP AUX (S (NP (NP (ADJP JJ)) , (NP NNP NNP)) (VP VBD (SBAR -NONE- (S -NONE-)))))))) .))

DT NN_000000A0000000000000A0000 VBD CD TO CD .
Desired parse : (S1 (S (NP DT NN) (VP VBD (NP CD) (PP TO (NP CD))) .))
Actual parse : (S1 (S (NP DT NN) (VP VBD (NP CD) (PP TO)) (S (NP CD) .)))

IN DT JJ NN_000D0000000000000000A0B NNP_0000000000A00000000000000 MD VB PRP$ NNS_00000000000000A0000000000 NNS_D00A0000000C00000B0000000 -NONE- TO VB NNP_0000000000A00000000000000 JJ NNS_C000000000000000A0000000 .
Desired parse : (S1 (S (PP IN (NP DT JJ NN)) (NP NNP) (VP MD (VP VB (NP PRP$ NNS) (NP NNS) (S (NP -NONE-) (VP TO (VP VB (NP NNP JJ NNS)))))) .))
Actual parse : (S1 (SBAR (S1 (S (PP IN (NP DT JJ NN NNP (ADVP MD) (VP VB (S1 (NP PRP$ NNS NNS) (S (NP -NONE-) (VP TO (VP VB (NP NNP JJ NNS)))))))))) .))

NNP_0000000000A00000000000000 NNP_0000000000A00000000000000 POS NN_00000000000000A0000000000 VBD IN CD JJ JJ NNS_00000000000000A0000000000 , VBG NNS_0000C00000B000A00000C0000 IN DT VBN NNP_0000000000A00000000000000 NNP_0000000000A00000000000000 NNP_0000000000A00000000000000 , MD AUX VBN -NONE- RB .
Desired parse : (S1 (S (NP (NP NNP NNP POS) NN) (VP VBD (SBAR IN (S (NP (NP CD JJ JJ NNS) , (PP VBG (NP (NP NNS) (PP IN (NP DT VBN NNP NNP NNP)))) ,) (VP MD (VP AUX (VP VBN (NP -NONE-) (ADVP RB))))))) .))
Actual parse : (S1 (S (S (NP (NP NNP NNP POS) NN) (VP VBD (PP IN (NP CD JJ JJ NNS)) , (VP VBG (NP NNS) (SBAR IN (S (NP DT) (VP VBN (NP NNP NNP NNP))))))) , (VP MD (VP AUX (VP VBN (S (NP -NONE-) (ADVP RB))))) .))

CC PRP_0000000000A00000000000000 MD VB RP -NONE- VBG RBR , CC VBG DT NN_0000000000DB0000000A00C00 JJR .
Desired parse : (S1 (S CC (NP PRP) (VP MD (VP VB (PRT RP) (S (NP -NONE-) (VP (VP VBG (ADVP RBR)) , CC (VP VBG (NP DT NN JJR)))))) .))
Actual parse : (S1 CC (S (NP PRP) (VP MD (VP VB (PRT RP (S (NP -NONE-) (VP VBG (PP RBR (S (NP ,) CC (NP VBG DT NN JJR)))))))) .)

References

[1] Cole, R., Mariani, J., Uszkoreit, H., Varile, G., Zaenen, A., Zue, V., and Zampoli, A. (1997). *Survey of the State of the Art in Human Language Technology*, Cambridge University Press and Giardini, ISBN: 0521592771

[2] Winograd, T. (1983). *Language as a Cognitive Process*, Addison-Wesley, Reading, MA.

[3] Dale, R., Moisl, H., and Somers, H. (2000). *Handbook of Natural Language Processing*, Marcel Dekker Inc, New York.

[4] Sells, P. (1985). *Lecture on Contemporary Syntactic Theories*. CSLI Series, Stanford.

[5] Charniak, E. (2000). A maximum-entropy-inspired parser. In *Proceedings of the conference of the North American Chapter of the Association for Computational Linguistics (NAACL-2000)*. PP 132-139.

[6] Collins, M. J., Duffy, N. (2002). New Ranking Algorithms for Parsing and Tagging: Kernels over Discrete Structures, and the Voted Perceptron. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pp 263-270.

[7] Bod, R. (2003). An Efficient Implementation of a New DOP Model. In *Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics (EACL-2003)*, Budapest, Hungary.

[8] Palmer-Brown, D., Tepper, J. A., and Powell, H. M. (2002). Connectionist Natural Language Parsing. *Trends in Cognitive Sciences*, **6(10)**, 437-442, Elsevier Science.

[9] Rohde, D. L. T., and Plaut, D. C. (2003). Connectionist models of language processing. *Cognitive Studies*, **10(1)**, 10-28.

[10] Frazier, L., Fodor, J. (1978). The sausage machine: A new two-stage parsing model. In *Cognition 6*. PP 291-325.

[11] Frazier, L. (1987). Sentence processing: Evidence from Dutch. In *Natural Language and Linguistic Theory 5*. PP 519-559.

[12] Ferreira, F., and Clifton, C., Jr. (1986). The independence of syntactic processing. *Journal of Memory and Language*, 25, pp 348-368.

[13] MacDonald, M. C., Pearlmutter, N. J., and Seidenberg, M. S. (1994). The lexical nature of syntactic ambiguity resolution, *Psychological Review*, 101, pp 676-703.

[14] Tanenhaus, M., and Carlson, G. (1989). Lexical structure and language comprehension, In W.D. Marslen-Wilson (Ed.), *Lexical representation and process* (pp. 505-528), Cambridge, MA: MIT Press.

[15] Tanenhaus, M. K., and Trueswell, J. C. (1995). Sentence comprehension. In J. Miller and P. Eimas, eds., *Handbook of Perception and Cognition Vol. 11: Speech and Language*. New York: Academic Press.

[16] Merlo, P., and Stevenson, S., (eds), (2000). *The Lexical Basis of Sentence Processing: Formal, Computational and Experimental Issues*, John Benjamin Publ. Co.

[17] Miller, G., 1990, WORDNET: An online lexical database, *International Journal of Lexicography*, **3(4)**.

[18] Tepper, J. A., Powell, H., and Palmer-Brown, D. (2002). A corpus-based connectionist architecture for large-scale natural language parsing, *Connection Science*, **14(2)**.

[19] Tepper, J. A., Powell, H., and Palmer-Brown, D. (2001). Corpus-Based Connectionist Parsing, In *Proceedings of the Second Workshop on Natural Language Processing and Neural Networks*, National Center of Science, Tokyo, ISSN 1346-6682, pp 8-15.

[20] Garside, R. G., Leech, G. N., Varadi, T. (1987). Manual of Information to Accompany the Lancaster Parsed Corpus. Department of English, University of Oslo.

[21] Manning, C. D., and Schutze, H. (1999). *Foundations of Statistical Natural Language Processing*, The MIT Press, Cambridge, MA.

[22] Taraban, R. and McClelland, J. L. (1990). Parsing and comprehension: A multiple constraint view. In Comprehension Processes in Reading, (Balota, D. A., Flores d'Arcais, G. B. and Rayner, K., eds), Lawrence Erlbaum, Hillsdale, NJ, pp 231-263.

[23] Chomsky, N. (1956). Three models for the description of Language, *IRE Transactions on Information Theory*, **2**, pp 113-124.

[24] Chomsky, N. (1957). *Syntactic Structures*, The Hague: mouton.

[25] Chomsky, N. (1959). On certain formal properties of grammar, *Information and Control*, **1**, pp 91-112.

[26] Chomsky, N. (1965). Aspects of the Theory of Syntax, Cambridge, MA: MIT Press.

[27] Chomsky, N. (1986). Knowledge of Language: Its Nature, Origin, and Use. New York: Prager.

[28] Grishman, R. (1986). Computational Linguistics: an introduction. Cambridge University Press.

[29] Turing, A. M. (1950). Computing Machinery and Intelligence, *Mind: A Quarterly Review of Psychology and Philosophy*, 59, pp 433-460.

[30] Aho, A. V. (1968). Indexed grammars: an extension to context-free grammars. In *Journal of the Association for Computing Machinery*, **15(4)**, pp 647-671.

[31] Aho, A. V. (1969). Nested Stack Automata. In *Journal of the Association for Computing Machinery*, **16(3)**, pp 383-406.

[32] Grune, D., and Jacobs, C. J. H. (1990). *Parsing techniques – A Practical Guide*. Ellis Horwood, Chichester, England.

[33] Unger, S. H. (1968). A global parser for context-free phrase structure grammars, *Communications of the ACM*, **11(4)**, pp 240-247.

[34] Cocke, J., and Schwartz, J. T. (1970). Programming languages and their compilers: Preliminary notes. *Technical report*, Courant Institute of Mathematical Sciences, New York University.

[35] Younger, D.H. (1967). Recognition of context-free languages in time n^3, *Information Control*, **10(2)**, pp 189-208.

[36] Kasami, T., and Torii, K. (1969). A syntax-analysis procedure for unambiguous context-free grammars, *Journal of the ACM*, **16(3)**, pp 423-431.

[37] Sakai, I. (1962). Syntax in universal translation, In *proceedings of the 1961 International Conference on Machine Translation of Languages and Applied Language Analysis, Her Majesty's Stationery Office, London*, pp 593-608.

[38] Earley, J. (1970). An efficient context-free parsing algorithm, *Communications of the ACM*, **13(2)**, pp 94-102.

[39] Tomita, M. (1986). *Efficient parsing for natural language*, Kluwer Academic Publishers, Boston, MA.

[40] Tomita, M. (1987). An efficient augmented-context-free parsing algorithm, *American Journal of Computational Linguistics*, **13(1-2)**, pp 31-46.

[41] Lewis II, P.M., and Stearns, R.E. (1968). Syntax-directed transduction, *Journal of the ACM*, **15(3)**, pp 465-488.

[42] Knuth, D.E. (1965). On the translation of languages from left to right, *Information Control*, **8**, pp 607-639.

[43] DeRemer, F., and Pennello, T.J. (1982). Efficient computation of LALR(1) look-ahead sets, *ACM Transactions on Programming Languages and Systems*, **4(4)**, pp 615-649.

[44] Ives, F. (1987). Response to remarks on recent algorithms for LALR lookahead sets, *ACM SIGPLAN Notices*, **22(8)**, pp 99-104.

[45] Rosenkrantz, D.J., and Lewis II, P.M. (1970). Deterministic left-corner parsing, In *IEEE Conference Record 11th Annual Symposium on Switching and Automata Theory*, pp 139-152.

[46] Small, S. (1981). Word Expert Parsing: A Theory of Distributed Word-Based Natural Language Understanding. Unpublished Ph.D Thesis, University of Maryland.

[47] Small, S., Cottrell, G., and Shastri, L. (1982). Toward connectionist parsing. In *Proceedings of the National Conference on Artificial Intelligence*, Pittsburgh, PA: AAAI, pp 247-250.

[48] Cottrell, G. W. (1985). Connectionist parsing. In *Proceedings of the 7th annual conference of the Cognitive Science Society*, Hillsdale, NJ: Lawrence Erlbaum Associates, pp 201-211.

[49] Cottrell, G. W. (1985). *A connectionist approach to word sense disambiguation*, Unpublished doctoral dissertation, Department of Computer Science, University of Rochester, Rochester, NY.

[50] Howells, T. (1988). VITAL: A connectionist parser. In *Proceedings of the 10th annual conference of the Cognitive Science Society*, Hillsdale, NJ: Lawrence Erlbaum Associates, pp 18-25.

[51] Waltz, D. L., and Pollack, J. B. (1985). Massively parallel parsing: A strongly interactive model of natural language interpretation, *Cognitive Science*, **9**, pp 51-74.

[52] Fanty, M. A. (1986). Context-free parsing with connectionist networks. In *Proceedings of AIP Conference on Neural Networks for Computers*, pp 140-145.

[53] Rager, J. E. (1992). Self-correcting connectionist parsing. In R.G. Reilly and N.E. Sharkey (Eds.), *Connectionist approaches to natural language processing*, Hillsdale, NJ: Lawrence Erlbaum Associates, pp 143-167.

[54] Selman, B., and Hirst, G. (1985). Connectionist parsing. In *Proceedings of the 7th annual conference of the Cognitive Science Society*, Isdale, NJ: Lawrence Erlbaum Associates, pp 212-221.

[55] Selman, B., and Hirst, G. (1994). Parsing as an energy minimisation problem, In G. Adriaens and U. Hahn (Eds.), *Parallel natural language processing*, Norwood, NJ: Ablex Publishing, pp 238-254.

[56] Fahlman, S. E., Hinton, G. E., and Sejnowski, T. J. (1983). Massively parallel architectures for AI: NETL, Thistle, and Boltzmann machines. In *Proceedings of the National Conference on Artificial Intelligence*, Washington, pp 109-113.

[57] Charniak, E., and Santos, E. (1987). A connectionist context-free parser which is not context-free, but then it is not really connectionist either, In *Proceedings of the 9th annual conference of the Cognitive Science Society*, Hillsdale, NJ: Lawrence Erlbaum Associates, pp 70-77.

[58] Rumelhart, D., and McClelland, J. (1986). On learning the past tenses of English verbs, In *Parallel Distributed Processing, Explorations in the Microstructure of Cognition*, vol 2, pp 216-271, MIT Press.

[59] McClelland, J. and Kawamoto, A. (1986). Mechanisms of sentence processing: assigning roles to constituents, In *Parallel Distributed Processing, Explorations in the Microstructure of Cognition*, vol 2, MIT Press.

[60] Hanson, S. J. and Kegl, J. (1987). PARSNIP: A connectionist network that learns natural language grammar from exposure to natural language sentences, In *Proceedings of the 9th Annual Conference of Cognitive Science*, pp 106-119.

[61] Apolloni, B. (1992). Learning to solve PP-attachment ambiguities in natural language processing through neural networks, In *IEEE Transactions on Neural Networks*, pp 199-205.

[62] Archambault, D., and Bassano, J. (1994). A neural network for supervised learning of natural language grammar. In *IEEE Transactions on Neural Networks*, pp 267-273.

[63] Kwasny, S. C. and Faisal, K. A. (1990). Connectionism and determinism in a syntactic parser, *Connection Science*, **2**, pp 63-82.

[64] Cleeremans, A., Servan-Schreiber, D., McClelland, J. (1989). Finite state automata and simple recurrent networks, In *Neural Computation*, **1(3)**, pp 372-381.

[65] Moisl, H. (1992). Connectionist finite state language processing, In *Connection Science*, **4(2)**, pp 67-91.

[66] Elman, J. L. (1990). Finding structure in time, *Cognitive Science*, **14**, pp 179-211.

[67] Jordan, M. I. (1986). Attractor dynamics and parallelism in a connectionist sequential machine, In *Proceedings of the 8th Annual Conference of the Cognitive Science Society*, pp 531-545.

[68] Ghahramani, Z., Allen, R. (1991). Temporal processing with connectionist networks. In *IEEE Transactions on Neural Networks*. PP 541-546.

[69] Tsoi, A. C., and Back, A. (1994). Locally recurrent globally feedforward networks, a critical review of architectures. In *IEEE Transactions on Neural Networks*, **5(2)**, pp 229-239.

[70] Pollack, J. B. (1990). Recursive Distributed Representations, *Artificial Intelligence*, **46**, pp 77-105.

[71] Weber, V., and Wermter, S. (1996). Using hybrid connectionist learning for speech/language analysis, In S. Wremter, E. Riloff, and G. Scheler (Eds.), *Lecture notes in artificial intelligence 1040: Connectionist, statistical, and symbolic approaches to learning for natural language processing*, Berlin: Springer-Verlag, pp 87-101.

[72] Wermter, S., and Weber, V. (1994). Learning fault-tolerant speech parsing with screen, In *Proceedings of the 12th National Conference on Artificial Intelligence*, Seattle, WA: AAAI, pp 670-675.

[73] Wermter, S., and Weber, V. (1997). SCREEN: Learning a flat syntactic and semantic spoken language analysis using artificial neural networks, *Journal of Artificial Intelligence Research*, **6**, pp 35-85.

[74] Jain, A. N., and Waibel, A. H (1990). Incremental parsing by modular recurrent connectionist networks, In D. Touretzky (Ed.), *Advances in neural information processing systems 2*, pp 364-371, San Mateo, CA: Morgan Kaufmann.

[75] Stevenson, S. (1994). *A competitive attachment model for resolving syntactic ambiguities in natural language parsing*, Unpublished doctoral dissertation, Department of Computer Science, University of Maryland.

[76] Stevenson, S., and Merlo, P. (1997). Lexical structure and parsing complexity, *Language and Cognitive Processes*, **12**, pp 349-399.

[77] Berg, G. (1992). A connectionist parser with recursive sentence structure and lexical disambiguity, In *Proceedings of the 10th National Conference on Artificial Intelligence*, San Jose, CA: AAAI, pp 32-37.

[78] Henderson, J. B. (1994). Connectionist syntactic parsing using temporal variable binding, *Journal of Psycholinguistic Research*, **23(5)**, pp 353-379.

[79] Henderson, J. B. (1994). *Description based parsing in a connectionist network*, Unpublished doctoral dissertation, University of Pennsylvania, Philadelphia, PA.

[80] Henderson, J. B. (1996). A connectionist architecture with inherent systematicity, In *Proceedings of the 18th annual conference of the Cognitive Science Society*, Hillsdale, NJ: Lawrence Erlbaum Associates, pp 574-579.

[81] Henderson, J. B., and Lane, P. C. R. (1998). A connectionist architecture for learning to parse, In *Proceedings of the 17th International Conference on Computational Linguistics and the 36th annual meeting of the Association for Computational Linguistics (COLING-ACL '98)*, University of Montreal, Canada.

[82] Lane, P. C. R., and Henderson, J. B. (1998). Simple synchrony networks: Learning to parse natural language with temporal synchrony variable binding, In *Proceedings of the 1998 International Conference on Artificial Neural Networks*, Skovde, Sweden, pp 615-620.

[83] Lane, P. C. R., and Henderson, J. B. (2003). Towards effective parsing with neural networks: Inherent generalisations and bounded resource effects, *Applied Intelligence*, **19**, pp 83-100.

[84] St. John, M. F., and McClelland, J. L. (1992). Parallel constraint satisfaction as a comprehension mechanism, In R. G. Reilly and N. E. Sharkey (Eds.), *Connectionist approaches to natural language processing*, Hillsdale, NJ: Lawrence Erlbaum Associates, pp 97-136.

[85] Harm, M. W., Thornton, R., and MacDonald, M. C. (2000). A distributed, large scale connectionist model of the interaction of lexical and semantic constraints in syntactic ambiguity resolution, In *Proceedings of the 13th annual CUNY Conference on Human Sentence Processing*, La Jolla, CA.

[86] Kemke, C. (1996). A hybrid approach to natural language parsing, In *Proceedings of ICANN '96*, pp 875-880.

[87] Kemke, C. (2002). A constructive approach to parsing with neural networks – the hybrid connectionist parsing method, In *Proceedings of the 15th Canadian Conference on Artificial Intelligence*, Calgary, Alberta, Canada.

[88] Sharkey, A. J. C., and Sharkey, N. E. (1992). Connectionism and natural language, In *Connectionist Natural Language Processing*, (Ed. N. Sharkey), Part 20, Intellect, pp 1-10.

[89] Miikkulainen, R. (1995). Subsymbolic parsing of embedded structures, In *Computational Architectures Integrating Neural and Symbolic Processes*, (Ed. R. Sun), Kluwer Academic Publishers, Boston, MA.

[90] Blaheta, D., and Charniak, E. (2000). Assigning function tags to parsed text, In *Proceedings of NAACL, 2000*, pp 234-240.

[91] Darken, C., and Moody, J. (1992). Towards faster stochastic gradient search, *Advances in Neural Information Processing Systems*, **4**, pp 1009-1016, San Mateo, CA: Morgan Kaufmann.

[92] Jacobs, R. A. (1988). Increased rates of convergence through learning rate adaptation, *Neural Networks*, **1(4)**, pp 295-307.

[93] Lawrence, S., Giles, C. L., and Tsoi, A. C. (1996). What size neural network gives optimal generalisation? Convergence properties of backpropagation, *Technical Report UMIACS-TR-96-22 and CS-TR-3617*, Institute for Advanced Computer Studies, University of Maryland.

[94] Haykin, S., 1999, Neural Networks: A Comprehensive Foundation (2 ed), Upper Saddle River, NJ: Prentice-Hall, ISBN: 0132733501

[95] Geman, S., Bienenstock, E., and Doursat, R. (1992). Neural networks and the bias/valence dilemma, *Neural Computation*, **4**, pp 1-58.

[96] Stone, M. (1974). Cross-validation choice and assessment of statistical predictions, *Journal of the Royal Statistical Society*, **B(36)**, pp 111-133.

[97] Morgan, N., and Bourlard, H. (1990). Continuous speech recognition using multilayer perceptrons with hidden Markov models, In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp 413-416.

[98] Collins, M. J. (2000). Discriminative reranking for natural language parsing. In *Proceedings of the 17th International Conference on Machine Learning*, pp 175-182.

[99] Magerman, D. M. (1995). Statistical decision-tree models for parsing, In *Proceedings of the 33rd Annual Meeting of The Association for Computational Linguistics*, pp276-283.

[100] Ratnaparkhi, A. (1996). A maximum entropy model for part-of-speech tagging, In *Proceedings of the Conference on Empirical Methods in Natural Language Processing, 1996*.

[101] Marcus, M. P., Santorini, B., and Marcinkiewicz, M. A. (1993). Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, **19**, pp 313-330.

[102] Prechelt, L. (1998). Automatic Early Stopping Using Cross Validation : Quantifying the Criteria, *Neural Networks*, **11(4)**, pp 761-767.

[103] Harrison et al. (1991). Evaluating syntax performance of parser/grammars, In *Proceedings of the Natural Language Processing Systems Evaluation Workshop (Technical Report RL-TR-91-362)*, Berkeley, CA.

[104] Williams, R.J., Peng, J. (1990). An efficient gradient-based algorithm for on-line training of recurrent network trajectories. *Neural Computation* **2**, pp 490--501.

[105] Bengio, Y. and Simard, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, **5(2)**, pp 157-166.

[106] Sharkey, N., Sharkey, A. and Jackson, S. (2000) Are SRN's sufficient for modelling language acquisition? *In:* P. Broeder and J. Murre, *Models of Language Acquisition: Inductive and Deductive Approaches*. Oxford University Press. pp 33-54.

[107]Gers, F. A. and Schmidhuber, J. (2001). Long Short-Term Memory learns context free and context sensitive languages. In V. Kurkova et.al., editor, *Proceedings of the ICANNGA 2001 Conference*, volume 1, pages 134--137, Wien, NY. Springer.

[108] Hochreiter, S. and Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, **9(8)**:1735--1780.

[109] Sharkey, N. E. (1991). Connectionist representation techniques, *Artificial Intelligence Review*, **5**, pp 143-167.

[110] Browne, A., and Sun, R. (2001). Connectionist inference models, *Neural Networks*, **14**, pp 1331-1355.

[111] Tepper, J. A. (2000). Corpus-based Connectionist Parsing, Unpublished Ph.D Thesis, The Nottingham Trent University.

[112] Johansson, S., Leech, G., Goodluck, H. (1978). *Manual of Information to Accompany the Lancaster-Oslo/Bergen Corpus of British English*. Department of English, University of Oslo. Also see ICAME Journal **16**, pp 124.

[113] Garside, R. (1987). The CLAWS word-tagging system. In *The Computational Analysis of English: A corpus-based approach*. (Ed. R. G. Garside, G. N. Leech, G. Sampson). London : Longman.

[114] Garside, R.G., Leech, G.N., Sampson G. (1987). *The Computational Analysis of English: A corpus-based approach*. London: Longman.

[115] Sheiber, S. M. (1983). Sentence disambiguation by a shift-reduce parsing technique. In *Computer Speech and Language*, pp 297-323.

[116] Sheiber, S. M. (1992). *Constraint-Based Grammar Formalisms*. MIT Press, Cambridge, Massachusetts.

[117] Buitelaar, P. (1998). A lexicon of underspecified semantic tagging. In *Proceedings of the ACL-SIGLEX Workshop "Tagging Text with Lexical Semantics: Why, What and How?"* Washington, DC. pp 25-33.

[118] Tikhonov, A. N. (1963). On solving incorrectly posed problems and method of regularisation. *Doklady Akademii Nauk*, USSR, **v151**, pp 501-504.

[119] Marcus M., Kim, G., Marcinkiewicz, M., et al. (1994). The Penn Treebank: Annotating Predicate Argument Structure. In *Proceedings of ARPA Speech and Natural Language Workshop*.

[120] Resnik, P. (1995). Disambiguating noun groupings with respect to WordNet senses. In *Proceedings of the Third Workshop on Very Large Corpora*, Cambridge, MA. pp 54-68

[121] Charniak, E., Johnson, M. (2005). Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of the 43rd meeting of the Association for Computational Linguistics*, Ann Arbor, MI.

[122] Mayberry, III, M., Miikkulainen, R. (2003). Incremental nonmonotonic parsing through semantic self-organisation. In *Proceedings of the 25th Annual Conference of the Cognitive Science Society*, Mahawa, NJ.: Erlbaum. pp 798-803.

[123] Ceusters, W., Rogers, J., Consorti, F., Rossi-Mori, A. (1999). Syntactic-semantic tagging as a mediator between linguistic representations and formal models: an exercise in linking SNOMED to Galen. *Artificial Intelligence in Medicine*, **v15**, pp 5-23.

[124] Buitelaar, P., Alexandersson, J., Jaeger, T., Lesch, S., Pfleger, N., Raileanu, D. (2001). An unsupervised semantic tagger applied to German. In *Proceedings of Recent Advances in NLP (RANLP)*, Tzigov Chark, Bulgaria.

[125] Pustejovsky, J., Boguraev, B., Verhagen, M., Buitelaar, P., Johnson, M. (1997). Semantic indexing and typed hyperlinking. In *Proceedings of the American Association for Artificial Intelligence Conference*, Spring Symposium, NLP for WWW. pp 120-128.

[126] Fellbaum, C., Grabowski, J., Landes, S. (1997). Analysis of a hand-tagging task. In *Proceedings of the ACL-SIGLEX Workshop "Tagging Text with Lexical Semantics: Why, What, and How?"* Washington, DC. pp 34-40.

[127] Palmer, M., Dang, H., Rosenzweig, J. (2000). Sense tagging the Penn Treebank. In *Proceedings of the 2nd International Conference on Language Resources and Evaluation (LREC-2000)*, Athens, Greece.

[128] Kingsbury, P., Palmer, M., Marcus, M. (2002). Adding semantic annotation to the Penn Treebank. In *Proceedings of the Human Language Technology Conference*.

[129] Miller, G., Chodorow, M., Landes, S., Leacock, C., Thomas, R. (1994). Using a semantic concordance for sense identification. In *Proceedings of the ARPA Workshop on Human Language Technology*, Plainsboro, NJ. pp 240-243.

[130] Chang, E., Huang, C., Ker, S., Yang, C. (2002). Induction of classification from lexicon expansion: assigning domain tags to WordNet entries. In *Proceedings of COLING-2002*, Taipei, Taiwan.

[131] Dill, S., Eiron, N., Gibson, D., Gruhl, D., Guha, R., Jhingran, A., Kanungo, T., Rajagopalan, S., Tomkins, A., Tomlin, J., Zien, J. (2003). SemTag and Seeker: Bootstrapping the semantic web via automated semantic annotation. In *Proceedings of the 12th International Conference on World Wide Web*, Budapest, Hungary.

[132] Zelle, J., Mooney, R. (1993). Learning semantic grammars with constructive inductive logic programming. In *Proceedings of the 11th National Conference on Artificial Intelligence*, Washington, DC. pp 817-822.

[133] Lowe, J., Baker, C., Fillmore, C. (1997). A frame-semantic approach to semantic annotation. In *Proceedings of the ACL-SIGLEX Workshop "Tagging Text with Lexical Semantics: Why, What, and How?"* Washington, DC. pp 18-24.

[134] Saarinen, S., Bramley, R. B., Cybenko, G. (1992). Neural Networks, backpropagation, and automatic differentiation, In A. Griewank and G. F. Corliss (Eds.), *Automatic Differentiation of Algorithms: Theory, Implementation, and Application*, Philadelphia: SIAM, pp 31-42.

[135] Jaeger, H., Maass, W., Principe, J. (2007). Special Issue on echo state networks and liquid state machines. Neural Networks, **20(3)**, pp 287 – 289.

[136] Chomsky, N. (1982). *Some Concepts and Consequences of the Theory of Government and Binding*. Cambridge, Mass.:MIT Press.

[137] Gazdar, G. (1983). Phrase structure grammars and natural languages. In *Proceedings of the 8th International Joint Conference on Artificial Intelligence*. pp 556-565.

[138] Bresnan, J. (1982). The Mental Representation of Grammatical Relations. Cambridge, Mass.:MIT Press.

[139] Gazdar, G., Klein, E. H., Pullum, G. K., Sag, I. A. (1985). *Generalised Phrase Structure Grammar*. Oxford: Blackwell, and Cambridge, MA: Havard University Press.

[140] Pollard. C., Sag, I. (1994). *Head-Driven Phrase Structure Grammar*. CSLI Series, Stanford. The University of Chicago Press.

[141] Mel'cuk, I. (1988). *Dependency Syntax: Theory and Practice*. State University of New York Press, Albany, New York.

[142] Wood, M. M. (1993). *Categorial Grammar*. Routledge.

[143] Schabes, Y. (1990). *Mathematical and Computational Aspects of Lexicalised Grammars*. Unpublished PhD thesis, University of Pennsylvania, Department of Computer Science.

[144] Cook, W. A. (1989). *Case Grammar Theory*. Georgetown University Press, Washington, DC.

[145] Fillmore, C. R. (1968). The case for case. In *Universals in Linguistics Theory*. Bach, E., Harms, R. (Eds). Holt, Rinehart and Winston.

[146] Jacobs, P., Rau, L. (1993). Innovations in text interpretation. In *Artificial Intelligence*. **63(1-2)**. pp 143-191.

[147] Marquez, L., Carreras X., Litkowski, K. C., Stevenson, S. (2008). Semantic Role Labelling: An Introduction to the Special Issue, *Computational Linguistics*, **34(2)**, pp 145-159.

[148] Van Valin, R. D., LaPolla, R. (1997). *Syntax: structure, meaning and function*. Cambridge, UK: CUP.

[149] Toutanova, K., aghighi, A., Manning, C. D. (2008). A Global Joint Model for Semantic Role Labelling, *Computational Linguistics*, **34(2)**, pp 161-191.

[150] Moschitti, A., Pighin, D., Basili, R. (2008). Tree Kernels for Semantic Role Labelling, *Computational Linguistics*, **34(2)**, pp 193-224.

[151] Punyakanok, V., Roth, D., Yih, W. (2008). The importance of Syntactic Parsing and Inference in Semantic Role Labelling, *Computational Linguistics*, **34(2)**, pp 257-287.

[152] Pradhan, S. S., Ward, W., Martin, J. H. (2008). Towards Robust Semantic Role Labelling, *Computational Linguistics*, **34(2)**, pp 290-310.