Exploiting Multi-Word Units in Statistical Parsing and Generation

Conor Cafferkey

B.Sc.

A dissertation submitted in fulfilment of the
requirements for the award of

Master of Science

to the

Dublin City University

School of Computing

Supervisor: Prof. Josef van Genabith

September 2008

Declaration

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of Master of Science is entirely my own work and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work

Signed _____

(Conor Cafferkey)

Student ID 52017857

Date 30th September 2008

Abstract

Syntactic parsing is an important prerequisite for many natural language processing (NLP) applications. The task refers to the process of generating the tree of syntactic nodes with associated phrase category labels corresponding to a sentence.

Our objective is to improve upon statistical models for syntactic parsing by leveraging multi-word units (MWUs) such as named entities and other classes of multi-word expressions. Multi-word units are phrases that are lexically, syntactically and/or semantically idiosyncratic in that they are to at least some degree non-compositional. If such units are identified prior to, or as part of, the parsing process their boundaries can be exploited as islands of certainty within the very large (and often highly ambiguous) search space. Luckily, certain types of MWUs can be readily identified in an automatic fashion (using a variety of techniques) to a near-human level of accuracy.

We carry out a number of experiments which integrate knowledge about different classes of MWUs in several commonly deployed parsing architectures. In a supplementary set of experiments, we attempt to exploit these units in the converse operation to statistical parsing—statistical generation (in our case, surface realisation from Lexical-Functional Grammar f-structures). We show that, by exploiting knowledge about MWUs, certain classes of parsing and generation decisions are more accurately resolved. This translates to improvements in overall parsing and generation results which, although modest, are demonstrably significant.

Contents

Chapter 1

Introduction

In general terms, parsing is the task of deducing the hierarchical grammatical structure of a sequence of tokens. In natural language processing (NLP), the term usually refers to "shallow" syntactic parsing. The task of shallow (or skeletal) syntactic parsing is to generate the tree of syntactic nodes with associated phrase category labels corresponding to a sentence (whereas "deep" parsing also resolves non-local dependencies and associates richer relations between the syntactic nodes). Since syntactic structure is strongly correlated with semantic interpretation, parsing is a prerequisite for a number of NLP tasks. This makes it a potential corner-stone technology in applications such as grammar checking, text understanding, question answering (QA) and transfer-based approaches to machine translation (MT).

In many cases skeletal parses are used as a prerequisite to generate deeper linguistic representations. The syntax trees generated by a shallow parser can be mapped to a deep functional representation such as that provided by Lexical-Functional Grammar (LFG) (Kaplan and Bresnan, 1982, Bresnan, 2001). It has been shown that deep dependencies (or predicate-argument relations) generated from treebank-trained shallow parser output can outperform those generated by hand-crafted unification/constraint-based grammars (Cahill, 2004, Cahill et al., 2008).

The past decade has seen considerable advances in data-driven approaches to parsing—those that do not rely on hand-crafted grammar rules. Such models have

been trained on large-scale syntax tree-annotated corpora such as the Penn Wall Street Journal (WSJ) Treebank (Marcus et al., 1994).

State-of-the-art probabilistic (stochastic) parsing models such as the history-based lexicalised generative parsers of Collins (1999) and Charniak (2000) can achieve labelled syntactic bracket recall and precision in the region of 90% when trained and evaluated on the WSJ treebank. Reranking the candidate parse trees produced by such parsers using discriminative machine learning (ML) techniques has been shown to yield further modest gains in parse quality (e.g. Charniak and Johnson 2005).

In such models, it is commonly held that parse accuracy has reached an upper bound when we do not incorporate some form of additional knowledge, supplementary to the syntactic information encoded in the treebank resource. For example Agirre et al. (2008) incorporate word sense data as an additional source of knowledge. Additionally it has been noted that refinements relying solely on the Penn WSJ dataset run the risk of over-fitting to the specific idiosyncrasies of that corpus; which is largely of the same topic written in a particular "financial-speak" style. If we are to improve treebank-based probabilistic parsing and achieve a more general purpose, less domain-specific parsing architecture, then more generalised sources of knowledge are a necessity.

We believe that bringing together different knowledge sources that encode different types of linguistic information as part of, or as a complement to, a statistical model could yield an improved parser. This information might come from a number of sources: machine-readable dictionaries, knowledge-banks and ontologies, etc. and could be exploited by means of incorporating machine learning techniques to disambiguate problematic cases in parsing.

Another potential source of linguistic information might come from identifying and flagging certain classes of multi-word units (MWUs), such as named entities and other fixed and semi-fixed expressions. This would be achieved by employing machine learning-based classifiers in the initial stages of a parsing pipeline.

Multi-word units are phrases that are lexically, syntactically and/or semanti-

cally idiosyncratic in that they are to at least some degree non-compositional (Sag et al., 2002). The term is a broad one that encompasses a wide range of distinct, but related, phenomena: from truly fixed "lexicalised" expressions (e.g. "*by and large*", "*ad hoc*"), to semi-fixed expressions such as compound nominals ("*chief executive officer*", "*machine translation*"), to more syntactically flexible units such as phrasal verbs ("*write*"+"*up*", "*get*"+"*over*"). Borrowing some terminology from the information extraction (IE) field, we include the concept of named entities (NEs) in our definition of MWUs. These are fixed and semi-fixed expressions that refer to proper names ("*Pierre Vinken*", "*New York City*"), time expressions ("*two days ago*", "*October 19th*") and number expressions ("*one million dollars*", "*60 mph*").

The correct treatment of multi-word units presents a challenge in a wide variety of tasks, as much in purely statistical approaches to NLP as in more linguistically-grounded approaches (and all that fall in between). Their identification is a non-trivial consideration in information extraction (IE), question answering (QA), machine translation (MT), corpus construction, the specification of linguistic formalisms and—as we explore in this thesis—parsing. While it is clear that MWUs can present a challenge in parsing (and it is often assumed that incorporating some degree of MWU knowledge in parsing is beneficial) there is surprisingly little published research on the subject matter of this thesis. In this context, we believe that our work fills a conspicuous gap in the literature.

Our core research hypothesis is that, if we possess a means of accurately identifying MWUs, we can exploit knowledge about their syntactic (phrasal) boundaries in parsing as "ground truths" or "islands of certainty" within the parse chart. It is our intuition that this should help resolve certain important classes of ambiguous parsing decisions (such as the well-known stumbling-blocks of preposition-phrase attachment and co-ordination), yielding better overall parse accuracy. We also believe that improvements in efficiency (i.e. parse speed) should be brought about by taking advantage of this MWU information.

To test our hypotheses we will carry out a number of experiments which integrate

knowledge about different classes of MWUs. Given the broad range of phenomena that fall under the umbrella term "multi-word unit", the identification and treatment of MWUs is often a complex proposition. Luckily however, there exist certain classes of MWUs that can be quite readily identified in an automatic fashion—in some cases to a near-human level of accuracy. We will demonstrate that by exploiting information about these units, certain classes of parsing decisions are indeed more accurately and efficiently resolved by the parser. This translates to improvements in overall parsing results which are modest (at best a 1.8% reduction in error) but demonstrably significant. Among the specific research questions that we will address are:

1. **How should we integrate MWU information in a statistical parsing architecture?** We will look at two general approaches to integrating multi-word units in parsing: on the one hand, modifying (essentially, retokenising) the corpus used to train an existing parser such that MWUs are treated as single "words with spaces"; and on the other, implementing a constraints mechanism internal to the parser such that MWU boundaries are adhered to.

2. **What kinds of MWUs are useful in statistical parsing?** MWU is a broad term. In our case we we will look specifically at several classes of named entities—name expressions, time expressions and number expressions—as well as prepositional multi-word expressions.

3. **What effect does MWU information have across different parsers?** We test our approach with tree different parsing architectures: a "vanilla" probabilistic context-free grammar (PCFG); a history-based, lexicalised, generative parsing model; and a PCFG exploiting latent annotations.

In a sense, our research is as much an investigation of how well several existing parsing models already (implicitly) account for the various phenomena associated with multi-word units as an evaluation of explicitly incorporating MWUs in the model.

As a supplementary task, we will also explore the role that knowledge about the same MWUs can play in the converse operation to statistical parsing: statistical

sentence generation (specifically, surface realisation from LFG f-structures). To this end, we will present the results of some preliminary experiments that exploit MWUs in a statistical chart-based sentence generator (Cahill and van Genabith, 2006).

We have published parts of the research presented here in Hogan et al. (2007) and Cafferkey et al. (2007). To the best of our knowledge, this research is the first systematic investigation of the effect of MWU data in treebank-based, wide-coverage CFG parsing and generation.

1.1 Thesis structure

The remainder of this thesis is structured as follows:

Chapter 2 This chapter provides a summary of some of the prominent statistical approaches to natural language parsing—covering linguistic formalisms, parsing architectures, the current state of the art and principal areas of current research.

Chapter 3 Here we describe multi-word units, including named entities and other types of multi-word expressions. The chapter covers the characteristics of such units, their syntactic (non-)compositionality and semantic interpretation and methods for their identification. We also present the motivation for incorporating MWUs in the task of syntactic parsing.

Chapter 4 This chapter represents the core of our research—evaluating the effects of different approaches to incorporating both automatically-deduced and gold-standard MWU data in several parsing architectures. We analyse our results and discuss the implications of our work.

Chapter 5 Here we describe several experiments where we exploit multi-word units in the task of statistical sentence generation (specifically, surface realisation from LFG f-structures). We provide an introduction to Lexical-Functional Grammar (LFG) and

the task of surface realisation from LFG f-structures before presenting our results and analysis.

Chapter 6 We provide a comparison of our work with related research on multi-word units in several approaches to parsing, including dependency parsing and parsing with Lexical-Functional Grammar. We also discuss related work on constrained parsing.

Chapter 7 Finally, we summarise our work and offer conclusions and some future research directions.

Chapter 2

Statistical Parsing

Syntactic parsing is the process of recognising a sentence and assigning a syntactic structure to it. Data-driven, statistical approaches to parsing—employing supervised learning and large-scale syntactically-annotated corpora—have been among the best performing approaches to the task. In this chapter we will cover some of the more widely deployed statistical approaches to syntactic parsing. We will cover probabilistic context-free grammars (PCFGs), parsing algorithms, treebanks, the current state of the art and some areas of ongoing research. We will also briefly discuss some alternative linguistic formalisms for syntactic parsing.

2.1 Introduction

Parsing—the task of deducing the syntactic structure of a string—is the prerequisite for a range of natural language processing tasks. Amongst the tasks to which parsing has been applied are information extraction (e.g. Surdeanu et al., 2003), machine translation (Riezler and Maxwell, 2006, Charniak et al., 2003) and sentence compression (Turner and Charniak, 2005). For an overview of how parsing is employed in these and other tasks see Lease et al. (2006). There exist a number of approaches and linguistic formalisms for parsing—ranging from purely statistical models to hand-crafted, rule-based systems. Here we are concerned with data-driven, statistical approaches to "shallow" parsing (where the aim is to generate the phrase-

structure tree corresponding to a sentence—e.g. Figure 2.2). These models are usually trained on large-scale syntax tree-annotated corpora such as the Penn Wall Street Journal (WSJ) Treebank (Marcus et al., 1994).

2.2 Formal grammars

A range of concepts from formal language theory can be applied to natural language, and the concept of formal grammars is the basis for most approaches to parsing. A formal grammar G is defined as a quad-tuple (N, Σ, P, S) consisting of:

- A finite set N of nonterminal symbols (or "variables")
- A finite set Σ of terminal symbols that is disjoint from N
- A finite set P of production rules, each of the form
  $(\Sigma \cup N)^* N (\Sigma \cup N)^* \rightarrow (\Sigma \cup N)^*$
- A distinguished symbol $S \in N$ that is the start symbol

In a formal grammar, each production rule maps from one string of symbols to another, where the first string contains at least one nonterminal symbol. In the case that the second string is the empty string—that is, that it contains no symbols at all—λ is typically written. The language of a formal grammar $G = (N, \Sigma, P, S)$, denoted as (G), is defined as all those strings over Σ that can be generated by starting with the start symbol S and then applying the production rules in P until no more nonterminal symbols are present.

2.2.1 The Chomsky hierarchy

The Chomsky hierarchy (Table 2.1) is a taxonomy of classes of formal grammars (and their corresponding formal languages). Under the Chomsky classification, the overwhelming majority of the syntax of natural languages can be expressed by context-free grammars (those recognisable by a non-deterministic pushdown automaton). It

Grammar	Language	Automaton	Production rule form
Type 0	Unrestricted	Turing machine	$\alpha \rightarrow \beta$
Type 1	Context-sensitive	Linear-bounded automaton	$\alpha A \beta \rightarrow \alpha \gamma \beta$
Type 2	Context-free	Non-deterministic pushdown automaton	$A \rightarrow \gamma$
Type 3	Regular	Finite state automaton	$A \rightarrow a$ *or* $A \rightarrow aB$

Table 2.1: The Chomsky hierarchy; where α and β are (possibly empty) strings of terminal and nonterminal symbols, γ is a non-empty string of terminals and nonterminals, A and B are non-terminals, and a is a terminal symbol.

is therefore unsurprising that CFGs of one form or other are the underlying model on which the majority of natural language parsers are based.

2.3 Context-free grammars

A context-free grammar (CFG) is a grammar that imposes the restriction that the left-hand side of each production rule consists of a single nonterminal symbol only. Formally, a CFG is a 4-tuple (N, Σ, P, S) consisting of the following parameters:

1. A set of nonterminal symbols N
2. A set of terminal symbols Σ disjoint from N
3. A set of productions P, each of the form $A \rightarrow \beta$, where A is a nonterminal and $\beta \in (\Sigma \cup N)^*$
4. The designated start symbol $S \in N$

2.3.1 Probabilistic context-free grammars

Probabilistic context-free grammars (PCFGs) represent a simple augmentation to a CFG whereby each production rule has an associated probability p:

$$A \rightarrow \beta \, [p] \tag{2.1}$$

A PCFG is therefore a 5-tuple (N, Σ, P, S, D) where D is a function assigning probabilities to each production in P such that:

$$\forall i \sum_{j} p(A_i \rightarrow \beta_j) = 1 \tag{2.2}$$

Given a PCFG, the probability of a specific parse T for a sentence S is defined as the product of the probabilities of all of the production rules r used to expand each node n in the parse tree:

$$p(T, S) = \prod p(r(n)) \tag{2.3}$$

Therefore, the optimum parse for the the sentence S is the highest-probability tree in the set of possible parses $\tau(S)$:

$$\hat{T}(S) = \arg \max_{T \in \tau(S)} p(T) \tag{2.4}$$

PCFGs are the core model upon which many state-of-the-art parsers are based, and indeed many such sophisticated parsing models can be viewed merely as refinements of the PCFG model (Charniak, 1997).

2.3.2 Lexicalised context-free grammars

A head-lexicalised PCFG, as first described by Black et al. (1992), is based on the assumption that syntactic constituents can be associated with a lexical head. In such a grammar the lexical head of a given constituent is associated with a corresponding production rule, constituting a means whereby lexical (i.e. word) dependencies can be integrated into the model.

It has been demonstrated that lexical dependencies are useful in resolving certain classes of syntactic ambiguity (such as preposition-phrase attachment as documented by Hindle and Rooth, 1993).

The utility of lexicalisation is however limited by the fundamental sparsity of lexical dependency data inherent in the training corpora upon which parsing models

have been trained. Nonetheless, many of the best-performing statistical parsers (e.g. Collins 1999, Charniak 2000, Charniak and Johnson 2005) employ, and clearly benefit from, lexicalised parsing.[1]

2.4 The Penn WSJ Treebank

Statistical approaches to syntactic parsing typically require large amounts of training examples from which, say, a probabilistic context-free grammar can be derived. This is typically in the form of a large-scale syntactically-annotated corpus, or treebank. The most widely used such treebank has been the Wall Street Journal (WSJ) portion of the Penn Treebank (Marcus et al., 1993).

The Penn Treebank project was undertaken at the University of Pennsylvania beginning in 1989 with the aim of producing a large-scale annotated corpus of American English. The initial goal was the construction of a corpus annotated in terms of parts of speech and skeletal parse structure; this was performed in a semi-automatic manner whereby the outputs of part-of-speech tagger and syntactic parser were corrected by human annotators. The version of the corpus used here—Treebank Release 2—was completed in 1995.

Treebank Release 2 provides a corpus of one million words of 1989 Wall Street Journal material annotated in Treebank II bracketing style (Marcus et al., 1994). This combines part-of-speech, syntactic, and some semantic functional annotations.

A full listing of the Penn Treebank POS and syntactic tags can be found in Appendices A and B; Table 2.2 shows the set of Penn functional sub-tags; Figures 2.1 and 2.2 show a typical sentence annotated in Treebank II bracketing style.

We will discuss how to extract a probabilistic context-free grammar from the Penn WSJ Treebank in Chapter 4.

[1] This is underpinned by back-off and smoothing techniques, and through more sophisticated refinements of the PCFG model (e.g. history-based models).

Tag	Description	Tag	Description
TEXT CATEGORIES:		-SBJ	Surface subjects
-HLN	Headlines and datelines	-TPC	Topicalised and fronted constituents
-LST	List markers	-CLR	Closely related
-TTL	Titles	SEMANTIC ROLES:	
GRAMMATICAL FUNCTIONS:		-VOC	Vocatives
-CLF	True clefts	-DIR	Direction and trajectory
-NOM	Non-NPs that function as NPs	-LOC	Location
-ADV	Clausal and NP adverbials	-MNR	Manner
-LGS	Logical subjects in passives	-PRP	Purpose and reason
-PRD	Non-VP predicates	-TMP	Temporal phrases

Table 2.2: Treebank II functional sub-tags

```
(S
   (NP-SBJ ( Mr.) (NNP Baker) )
   (VP
      (VP
         (VBZ wears)
         (NP (DT a)
             (NP (NN tweed) (NN jacket) )
         )
         (PP-LOC (IN on)
            (NP (PRP$ his)
                (NP (NN ghostbusting) (NNS forays) )
            )
         )
      )
)
```

Figure 2.1: A syntactic analysis for the sentence "*Mr. Baker wears a tweed jacket on his ghostbusting forays*" annotated in Treebank II bracketing style (adapted from WSJ §04, sentence 347).

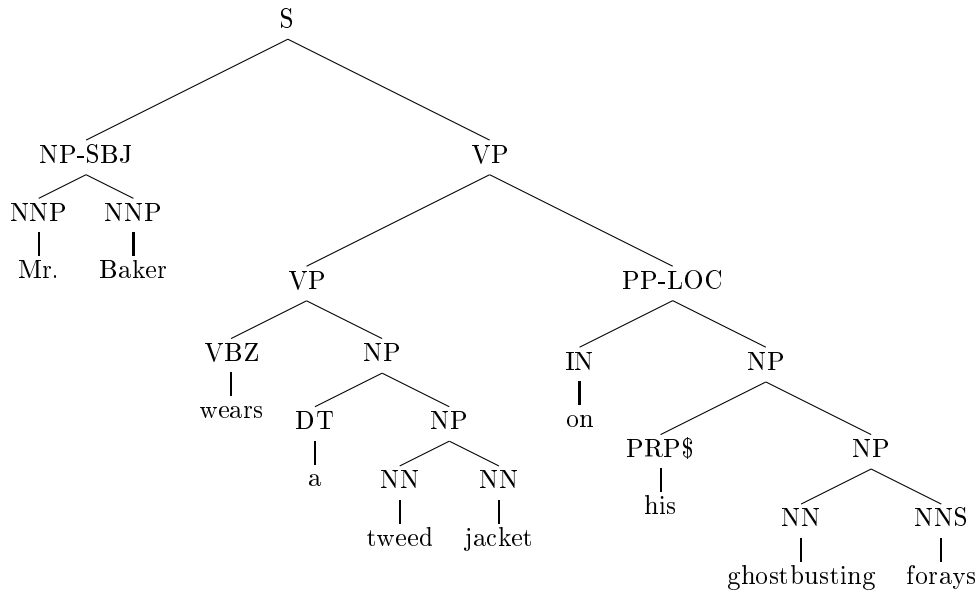

Figure 2.2: Graphical representation of analysis of the sentence "*Mr. Baker wears a tweed jacket on his ghostbusting forays*" from Figure 2.1.

2.5 Parsing with PCFGs

Parsing can be regarded as a search problem whereby we search through the space of possible parse trees to find the correct parse for a given sentence (this search space is defined by the grammar). One of the most commonly used algorithms for for context-free parsing is the Cocke-Younger-Kansami (CYK) algorithm (Aho and Ullman, 1972).

2.5.1 The CYK algorithm

The (probabilistic) CYK parser is a bottom-up dynamic programming algorithm that allows us to efficiently calculate (in cubic time) the most probable parse for a given sequence of word tokens $w_1 ... w_n$ and a probabilistic context-free grammar G.

The algorithm requires that the grammar be in Chomskey Normal Form (CNF); meaning that production rules are of the form $A \rightarrow B\,C$ or $A \rightarrow \alpha$, where A, B and C are nonterminal symbols and α is a terminal symbol (in other words, all non-preterminal productions are binary-branching). Since it is quite trivial to convert any context-free grammar to CNF, a CYK parser can be used to recognise any context-free

```
#initialisation
for all i,j,k
    p[i,j,k] = 0

#base case
for i = 1...n
    for k = 1...G
        if k → wi is in grammar
            p[i,i,k] = P(k → wi)

#recursive case
for s = 2...n
    for i = 1...n-s+1
    j = i+s-1
    for m = i...j-1
        for k = 1...G
            for k1 = 1...G
                for k2 = 1...G
                prob = p[i,m,k1] * p[m+1,j,k2] * P(k → k1 k2)
                if (prob > p[i,j,k])
                    p[i,j,k] = prob
                    B[i,j,k] = {m,k1,k2}
```

Figure 2.3: Pseudo-code for the CYK algorithm (based on that given in Cahill, 2004).

language.

Figure 2.3 gives the pseudo-code for a CYK implementation. First, we initialise a parse chart of size n by n. The base case then populates the diagonal of the chart (`[i][i]`) with unary productions—rules of the form $A \rightarrow a$, where A is the syntactic category (usually corresponding to a part of speech tag) associated with the word token a. Binary productions (rules spanning two constituents—$A \rightarrow A\,B$) will be considered during the recursive case.

The recursive case fills the chart bottom-up, left-to-right. At any chart position `[i][j]`, a rule $A \rightarrow A|a\ B|b$ may be inserted into the chart (with a corresponding probability) if there is already a rule with a LHS A at position `[i][k]` and a rule with LHS B at cell `[i+k][j-k]`, for all $i \leq k \leq j$.

The most common version of the algorithm determines the most probable rule at each stage, and retains that rule only (in this way we can efficiently calculate the most likely derivation). Once this process is completed, the sentence is recognised by the grammar if the subsequence containing the entire sentence is matched by the

Rule	Prob.	Rule	Prob.
S → NP VP	1.0	DT → *Mr.*	0.5
NP → DT NP	0.6	NNP →*Baker*	0.7
NP → NN NN	0.2	VBZ → *wears*	1.0
NP → NNP NNP	0.2	DT → *a*	0.5
VP → VBZ NP	1.0	NN → *tweed*	0.3
		NN → *jacket*	0.7
		NN → *Baker*	0.3

Figure 2.4: An example probabilistic context-free grammar.

	Mr 0	*Baker* 1	*wears* 2	*a* 3	*tweed* 4	*jacket* 5
0	DT = 0.5	NP = 0.6				S = 1.0
1		NNP = 0.7 NN = 0.3				
2			VBZ = 1.0			VP = 1.0
3				DT = 0.5		NP = 0.6
4					NN = 0.3	NP = 0.2
5						NN = 0.7

Figure 2.5: Probabilistic CYK parse chart for the sentence "*Mr. Baker wears a tweed jacket*".

start symbol.

By way of an example, Figure 2.5 illustrates the parse chart for the sentence "*Mr. Baker wears a tweed jacket*", given the PCFG grammar in Figure 2.4.

2.6 History-based (lexicalised, generative) parsing

In a history-based parsing model the parse tree is represented as a sequence of decisions, the decisions being made in some derivation of the tree. Each such decision has a probability and the product of the probabilities in a given derivation defines its likelihood over all possible derivations. History-based parsing models were first described by Black et. al (1992) as "history-based grammar models" (for a formal, generalised definition of history-based models see Bikel 2004). In its most general

form, a history-based parsing model states that all prior parse decisions can influence any later parse decisions in the derivation. In practice, however, only specific conditioning features are used in the history (such as path to the root node) to lend specific biases to the model. Here we will describe two such history-based approaches: that of Collins (1999) and that of Charniak (2000, 2005).

2.6.1 Collins (1999)

The three generative lexicalised parsing models described by Collins (1999) are examples of history-based models. Collins' model 1 extends a PCFG into a lexicalised dependency grammar-like framework, modelling dependencies between pairs of head words (bilexical statistics). Lexicalisation is performed using a set of head-finding heuristics based on those used by Magerman (1995).

Collins' model 2 additionally attempts to distinguish arguments from adjuncts and to model probabilities over subcategorisation frames. Another important linguistically-motivated feature of Collins models is that it treats "base NPs" (i.e. non-recursive noun phrases) quite differently from normal NPs (for details see Collins 1999, Bikel 2004).

Collins' model 3 further refines the model, making use of the trace annotations present in the Penn Treebank which describe *Wh*-movement. This particular augmentation, however, does not yield significant performance gains over model 2.

In Chapter 4 we will make use of Bikel's (2002) multilingual statistical parsing engine[2] emulating Collins' (1999) model 2.

2.6.2 Charniak (2000)

Charniak's (2000) "maximum entropy-inspired" parser employs a similar history-based approach to that of Collins. The major technical innovation of Charniak's parser is the use of a "maximum-entropy-inspired" model for conditioning and smoothing, where all probability distributions are heavily backed off and smoothed using

[2] available from `http://www.cis.upenn.edu/~dbikel/software.html`

Chen and Goodman's (1996) method.

2.6.2.1 Reranking: Charniak and Johnson (2005)

The Charniak and Johnson (2005) approach takes the n-most likely parses proposed by a modified version of the Charniak (2000) parser and employs a discriminative reranker—a maximum-entropy based estimator—to select the best parse. This method allows the incorporation of syntactic relationships that are more difficult to express in the generative model. This type of "pipeline" parsing architecture represents the current state of the art in statistical parsing.

2.7 PCFG with latent annotations

It was demonstrated by Klein and Manning (2003)—building on similar work carried out by Johnson (1998)—that an unlexicalised, "plain" PCFG parser can achieve performance that approaches that of history-based, lexicalised models if suitable linguistically-motivated heuristics are employed to augment the set of grammar symbols. For example, if the training corpus is modified such that the syntactic category of each node's immediate parent is appended to its syntactic label a significant improvement in accuracy is brought about.

Matsuzaki and Miyao (2005) introduced the idea of a "PCFG with latent annotations" (PCFG-LA), where fine-grained PCFG rules are induced using an expectation-maximisation (EM) algorithm to append "latent variables" to the nonterminal grammar symbols. This is similar in spirit to Klein and Manning's (2003) work, which uses manual feature selection to augment the grammar symbols. Petrov et al. (2006) expand on this idea with the Berkeley parser, which exploits the concept of a hierarchically-split PCFG.

2.7.1 Berkeley parser

The Berkeley Parser (Petrov et al., 2006)[3] employs an algorithm in which basic nonterminal symbols of a unlexicalised PCFG are alternately split and merged to maximise the likelihood of the training treebank. Their grammars automatically learn the kinds of linguistic distinctions exhibited in previous work with heuristic-based tree annotation.

The parser is trained by starting with a treebank-induced grammar and repeatedly splitting, merging and smoothing grammar symbols using an expectation maximisation algorithm.

2.8 Alternative linguistic formalisms for parsing

There exist many differing approaches to the analysis of natural language syntax and therefore there are many possible approaches to syntactic parsing. Among these are deep linguistic formalisms—such as HPSG, LFG and CCG—and dependency-based analyses.

2.8.1 Deep parsing

In "deep" parsing we generate phrase-structure trees similar to those that would be output by a skeletal parser but with a much richer set of grammatical relations between the nodes of the tree. These relations might include predicate-argument dependencies as well as other types of functional and semantic relations. In order to obtain accurate and complete representations deep parsers usually involve the representation and resolution of non-local dependencies (NLDs).

Examples of deep linguistic formalisms include Head-driven Phrase Structure Grammar (HPSG—Pollard and Sag, 1994), Lexical-Functional Grammar (LFG—Dalrymple, 2006), Tree-Adjoining Grammar (TAG— Joshi, 1987) and Combinatory Categorial Grammar (CCG—Steedman, 2000).

[3] available from `http://nlp.cs.berkeley.edu/`

In contrast to the statistical approaches to parsing with which we are concerned in this thesis, deep parsing has traditionally employed rule-based approaches based on hand-crafted grammars (a notable exception being the StatCCG parser for Combinatory Categorial Grammar).[4]

In this thesis we will perform a number of experiments exploiting multi-word units in sentence generation from LFG f-structures and, accordingly, we provide a summary of Lexical-Functional Grammar in Chapter 5.

2.8.2 Dependency parsing

An alternative formalism for syntactic parsing is dependency grammar. Dependency grammar parsing is the task of deriving the tree which represents grammatical relations—dependencies—between a given sequence of word tokens. Dependencies are defined in terms of direct grammatical relationships between the the word tokens in the sentence.

Dependency parsing is often said to combine many of the attributes of shallow (constituency) parsing with those of deeper approaches to parsing grounded in linguistic formalisms such as HPSG or LFG. Among the advantages of dependency grammars is their ability to naturally model non-nested constructions, which is important in freer-word order languages such as Czech, Dutch, and German (Nivre, 2005).

Figure 2.6 shows the sentence "*Mr. Baker wears a tweed jacket*" in dependency annotation (cf. Figure 2.2).

Among the current state of the art dependency parsers are Nivre et al.'s (2007) MaltParser, McDonald et al.'s (2006) MSTParser and "ensemble" approaches such as Sagae and Tsujii (2007). Dependency parsing has been the subject of a number of recent workshops and shared tasks including the CoNLL-X and CoNLL-2007 tasks (Buchholz and Marsi, 2006, Nivre et al., 2007a).

[4] More recently, a large amount of research has concentrated on automatic treebank-based acquisition of deep HPSG, CCG and LFG resources (Miyao et al., 2004, Hockenmaier, 2003, Cahill et al., 2008).

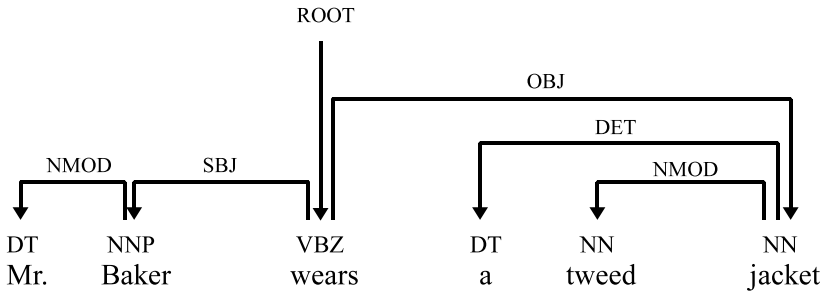

Figure 2.6: A dependency structure analysis for the sentence "*Mr. Baker wears a tweed jacket*" (from Figure 2.2).

Parser	% Recall	% Precision	F-score
Collins (1999) model 2	88.47	89.30	88.88
Bikel (2002) model 2 emulation	88.72	89.03	88.87
Charniak (2000)	89.6	89.5	89.55
Charniak and Johnson (2005)			90.1
Berkeley Parser (Petrov et al., 2006)	89.8	89.6	89.7

Table 2.3: Published state of the art results for statistical parsing of the Penn WSJ Treebank. Labelled recall, precision and f-score. All sentence lengths.

2.9 Summary and research directions

Some of the state-of-the art results reported in the literature for parsing the Penn WSJ corpus are given in Table 2.3.[5] Scores are given according to PARSEVAL labelled bracketing recall and precision measures (see Section 4.6).

We summarise some of the principal avenues of ongoing research in statistical parsing below.

2.9.1 Evaluation metrics

There have been many criticisms of the PARSEVAL bracketing recall and precision metrics (Abney et al., 1991) as a means of evaluating parser performance. Specific criticisms that have been leveraged are that the PARSEVAL measures penalise cer-

[5] The scores given here for Bikel's model 2 implementation and the Berkeley parser differ from the baseline scores that we will see in Chapter 4 due to different settings (in particular, we use gold-standard part-of-speech tags in our experiments).

tain attachment errors types too harshly, and that they are overly sensitive to the treebank annotation scheme. Other approaches to parser evaluation that have been proposed include leaf-ancestor (Sampson and Babarczy, 2003) and dependency-based evaluations (Carroll et al., 2002).

2.9.2 Domain adaptation

Parsers trained and repeatedly evaluated on the WSJ treebank run the risk of overfitting to that particular corpus, meaning that the parser generalises poorly to other domains. Domain adaptation (or genre portability) is thus an important topic. Some notable work on the subject of domain adaptation includes Sekine (1998), Gildea (2001) and Foster et al. (2007).

2.9.3 Semi-supervised training

Since there is only a finite amount of hand-annotated training data available (in the form of treebanks), semi-supervised learning is highly desirable in statistical parsing. Examples include Steedman et al. (2003) and McClosky et al. (2006). The approach taken for learning latent annotations in the Berkeley parser can also be seen as an instance of semi-supervised learning.

2.9.4 Extra-treebank linguistic information

It is our contention in this thesis that bringing together different knowledge sources that encode different types of linguistic information as part of, or as a complement to, the treebank-induced statistical model could yield an improved parser. This information might come from a number of sources: machine-readable dictionaries, knowledge-banks and ontologies, etc. and could be exploited by means of incorporating machine learning techniques to disambiguate problematic cases in parsing. Such approaches have shown promise in some areas; for example, Hogan (2007) shows that exploiting WordNet (Fellbaum et al., 1998) information can improve parsing co-ordinate structures.

Chapter 3

Multi-Words Units

This chapter defines the concept of multi-word units (MWUs)[1], which include (multi-word) named entities (NEs), compound nouns, compound function words, idioms, and many other forms of multi-word expressions (MWEs). We describe the classification of these units, their syntactic and semantic compositionality, methods for their identification and NLP applications that can benefit from the notion of MWUs. Finally, we present the case for incorporating knowledge of MWUs in statistical approaches to syntactic parsing.

3.1 Definition and classification

At a high level, we can define multi-word units as lexically, syntactically and/or semantically idiosyncratic phrases (i.e. sequences of two or more word tokens).[2] Such expressions can often be treated as single lexical units. MWUs are comprised of a large number of related but distinct phenomena.

The concept of MWUs is one that has been widely employed in information retrieval (IR), information extraction (IE) and question-answering (QA) systems where

[1] Since we will borrow terminology and technology from the information extraction field we use the more general term 'multi-word unit' rather than 'multi-word expression' which is commonly found in the computational linguistics literature. This is to be interpreted as MWU = MWE + NE, treating named entities as a separate class of MWU.

[2] In general these units are contiguous, composed of non-disjoint strings of word tokens, although this need not be the case.

collocations and named entities play important roles. These MWU types are only the tip of the iceberg however: the term can refer to a broad range of lexically, syntactically and/or semantically irregular strings of word tokens such as idioms, compound nouns, proper names, verb-particle constructions and light verbs. MWUs are found to varying degrees across all text genres and thus their identification, flagging, analysis and interpretation is potentially useful in almost all NLP applications. Although (as is the case in NLP in general) the phenomena associated with MWUs are most documented for English, they are by no means unique to English.

3.1.1 Types of multi-word units

As noted, the term "multi-word-unit" is a broad one. Here we will provide some description of the specific types of MWU to which we will refer. Table 3.1 provides some examples which illustrate some of the characteristics of these MWU classes.

We will discuss the main classes of MWUs in more detail below. Our classification of MWUs roughly follows that of Sag et al. (2002), consisting of four main types (in order of lexical rigidity): fixed expressions, semi-fixed expressions, syntactically-flexible fixed expressions and institutionalised phrases. We will treat multi-word NEs as a separate class, although there is some overlap with the aforementioned categories.

3.1.1.1 Fixed expressions

Fixed expressions are syntactically fully non-compositional and as such are lexically, morphologically and syntactically immutable. Examples include phrases such as "*by and large*" and foreign-language terms used in English such as "*ad hoc*". Although there certainly exists a large amount of such expressions, they are much less plentiful than other, more lexically flexible types of MWUs.

Syntactically rigid units:	
Fixed expressions	*by and large*
	for example
	ad hoc
Non-decomposable idioms	*kick the bucket*
	trip the light fantastic
	shoot the breeze
Compound nominals	*chief executive officer*
	car park
	machine translation

Syntactically-flexible units:	
Verb-particle constructions	*write + up*
	fall + off
	brush up + on
Decomposable idioms	*let the cat out of the bag*
	spill the beans
	sweep under the rug
Light verbs	*make + a mistake*
	give + a demo
	take + a nap

Named entities:	
Personal names	*Martha Matthews*
	Yoshio Hatakeyama
Organisations	*Rolls-Royce Motor Cars Inc.*
	Washington State University
Locations	*New York City*
	People's Republic of China
Time expressions	*October 19th*
	two years ago
	the 21st century
Quantities	*$2.7 million to $3 million*
	about 25%
	60 mph

Table 3.1: A possible taxonomy of multi-word unit types, with examples. For our purposes we treat named entities as a separate class of MWUs (although this distinction is not always clear-cut).

3.1.1.2 Semi-fixed expressions

The class of semi-fixed expressions—although for the most part lexically and syntactically rigid—are sometimes variable in terms of inflection, reflexive forms and determiner selection. Such expressions include non-decomposable idioms, compound nominals and proper names:

Non-decomposable idioms The syntactic composition of true idioms follows normal grammar rules, but they are semantically non-decomposable. As a consequence they tend to occur as syntactically rigid units. Examples include the oft-cited "*Kick the bucket*".

Compound nominals These are syntactically immutable units that inflect for number. Such units are plentiful in English, with examples such as "*chief executive officer*", "*machine translation*" and countless others. Compound nominals can sometimes be idiosyncratic in their inflections.

Proper names These are names of people, organisations, locations, etc.—equivalent to the class of ENAMEX-type named entities (see below).

3.1.1.3 Syntactically-flexible fixed constructions

Less rigid units such as verb-particle constructions (or phrasal verbs), decomposable idioms and light verbs are subject to a much greater degree of syntactic variability. This means that they are likely to require more complex treatments than the fixed and semi-fixed expressions that we have seen.

Verb-particle constructions A verb-particle construction is comprised of a verb together with one or more associated "particles", constituting a single semantic unit. Examples include "*write*"+"*up*", "*fall*"+"*off*" and "*brush up*"+"*on*". Any treatment of these units at the lexical level will prove problematic since they are not necessarily comprised of a contiguous string of tokens.

Decomposable idioms These are idiomatic expressions that are semantically decomposable, and can thus be subject to syntactic variability. Examples include "*Let the cat out of the bag*".

Light verbs A light verb is a verb participating in complex predication that has little semantic content of its own, but provides through inflection some details on the event semantics. Examples include "*make*"+"*a mistake*", "*give*"+"*a demo*" and "*take*"+"*a nap*".

3.1.1.4 Institutionalised phrases

Institutionalised phrases are syntactically and semantically compositional, but statistically frequent (e.g. "*at the weekend*"). Lexicalised statistical models of language should (at least theoretically) already capture the information required to account for the statistical bias of such expressions. Since these phrases are subject to full syntactic variability we will not consider them "true" MWUs in the sense with which this thesis is concerned.

3.1.1.5 Multi-word named entities

Named entities are single- or multi-word units that convey the names of specific people, organisations, locations, times or quantities. Strictly speaking, NEs refer to rigid designators—most commonly proper names; for example the names of people, organisations and locations. The term often also encompasses time and quantity expressions (which are not necessarily rigid designators).

The names of people, organisations, locations, etc. are referred to as name expressions (generally denoted ENAMEX); times, dates, days of the week, etc. are classified as time expressions (TIMEX); while quantities and other values are categorised as number expressions (NUMEX).

A number of type hierarchies have been defined for named entities : most notably the BBN hierarchy (Weischedel and Brunstein (2005)—designed for the QA task) and Sekine's extended hierarchy (Sekine et al., 2002).

The BBN Pronoun Coreference and Entity Type Corpus (Weischedel and Brunstein, 2005) supplements the Penn WSJ treebank (see Chapter 2) with annotations of the 29 NE types defined in the BBN hierarchy, including nominal-type NEs such as person, organisation, location, etc. as well as numeric types such as date, time, quantity and money. We will make use of this resource in Chapters 4 and 5.

3.2 (Non-)Compositionality and interpretation

The questions of syntactic and semantic compositionality and interpretation of MWUs come up to varying degrees in different NLP tasks. In applications that are linguistically-informed (i.e. not purely statistically-driven) it will not suffice to merely identify MWUs without some degree of further analysis. As we have seen, the degree of compositionality and thus syntactic and semantic interpretability varies depending on the class of MWU. The syntactic function performed by an MWU as a whole is often easily and systematically discernible—but this is not always the case. It follows that any computational treatment of MWUs should at least in some way be informed by the degree of (non-)compositionality of a given class of units.

Truly fixed expressions are essentially non-compositional and as such do not require any further analysis. It thus makes sense to treat them as immutable units (or "words with spaces"). Semi-fixed expressions can often be given a similar treatment. More flexible expressions require a more sophisticated, semantically-oriented approach however. We will partially explore this in Chapter 4, but the question of semantic interpretability is, for most part, beyond the scope of this thesis.

3.3 Identifying multi-word units

There exists are growing body of research on the identification and classification of MWUs of various types. This has been yielded by several workshops on the subject (at ACL-2006 and 2007, for example) and, in particular, the Multiword Expression Project led by Stanford University.

Approaches have ranged from hand-crafted, rule-based approaches to—particularly in the case of named entities—statistical methods. The two obviously are not completely disjoint, for example a statistical classifier (the latter case) might make use of MWU resources compiled using hand-crafted approaches (the former).

3.3.1 Simple approaches

Simple approaches such as identifying statistical co-occurrences of word tokens (collocations) or manually listing out sequences deemed to be MWUs (laboriously adding "words with spaces" to the lexicon of a rule based parser, for example) can be useful for certain, more lexically rigid MWU types. However such approaches scale poorly—they do not take into account the underlying phenomena at play and fail to account for the varying lexical, syntactic and semantic behaviours of different types of MWUs.

3.3.2 Rule-based approaches

Another method for MWU identification is to employ a rule-based approach to extract units from large-scale corpora in an automatic and semi-automatic fashion. This might be based on, for example, parts of speech and template matching. The dictionary of candidate prepositional multi-word expressions[3] that we will use in Chapters 4 and 5 was generated in this manner from the British National Corpus (BNC—Burnard, 2000).

3.3.3 Supervised learning

A more scalable solution is to employ supervised learning in the form of "semantic taggers" (sequential classifiers)—or cascades of such taggers—trained on large-scale MWU-annotated corpora. An example of this approach is that of Piao et al. (2003). Machine learning-based methods have, in particular, been applied to named entity recognition (see below).

[3] Based on resource from `http://mwe.stanford.edu`

[$_{\text{ENAMEX}}$Diamond Shamrock Offshore] 's stock rose [$_{\text{NUMEX}}$12.5 cents] [$_{\text{TIMEX}}$Friday] to close at [$_{\text{NUMEX}}$$ 8.25] in [$_{\text{ENAMEX}}$New York Stock Exchange] composite trading .

Figure 3.1: Output from a named entity recognition (NER) system for the sentence "*Diamond Shamrock Offshore's stock rose 12.5 cents Friday to close at $8.25 in New York Stock Exchange composite trading*".

System	% precision	% recall	f-score
baseline	71.91	50.90	59.61
Florian et al. (2003)	88.99	88.54	88.76
Chieu and Ng (2003)	88.12	88.51	88.31

Table 3.2: CoNLL-2003 language-independent NER task: English results. Precision, recall and f-score (harmonic mean of precision and recall).

3.3.4 Named entity recognition

Significant work has been carried out on the automatic identification (and classification) of named entities. The task of named entity recognition (NER), which is generally presented as a subtask of information extraction, has been the subject of a number of shared tasks and workshops. The problem is essentially a sequential classification task, to which both knowledge engineering- and machine learning-based approaches have been applied.

One of the first competitive evaluations of approaches to NER was as part of the Message Understanding Conferences (MUC-6, MUC-7) in 1995 and 1998. More recently, the CoNLL-2002 and CoNLL-2003 shared tasks evaluated language-independent NER systems which employed supervised learning using a wide range of machine-learning paradigms. The MUC and CoNLL tasks focused specifically on name expressions (ENAMEX), defining four broad subcategories: persons (denoted PER), locations (LOC), organisations (ORG), and miscellaneous (MISC). The performance achieved by the state-of-the-art systems is summarised in Tables 3.2 and 3.3.

Purely statistical approaches achieve performance below rule-based systems (although it should be noted that the datasets used to evaluate these differing approaches are not directly comparable). In both cases performance can approach near-human levels (e.g. on the MUC-7 task: human 97.6%, best NER 93%).

System	% precision	% recall	f-score
baseline			58.89
human	98	98	97.60
Mikheev et al. (1998)	95	92	93.39

Table 3.3: MUC-7 (1998) NER task. Precision, recall and f-score (harmonic mean of precision and recall).

Figure 3.1 shows a typical output from a named entity recogniser for the sentence "*Diamond Shamrock Offshore's stock rose 12.5 cents Friday to close at $8.25 in New York Stock Exchange composite trading*", where name expressions, number expressions and time expressions have been flagged.

In Chapters 4 and 5 we will make use of the output from the Chieu and Ng's (2003) NER system as evaluated in the CoNLL-2003 shared task. Chieu and Ng's system uses a maximum entropy sequential classifier to assign tags to unseen data, achieving almost 90% recall and precision. The English-language training and test data was derived from the Reuters corpus supplemented with gold-standard annotations of the four NE types defined in the task.

3.4 Multi-word units in NLP applications

The treatment of multi-word units presents a challenge in a wide variety of tasks, as much in purely statistical approaches to NLP as in more linguistically-grounded approaches (and all that fall in between). An obvious example is that of named entities, which are an important concept in information extraction (IE), and the related task of question answering (QA). Here the ability to accurately recognise rigid designators such as proper names is vital, and has been the impetus for much of the research on named entity recognition.

Other types of MWU are equally important in NLP, however this importance is often under-estimated. In addition to the tasks of IE and QA the identification and treatment of MWUs is a non-trivial component in (among other areas) machine translation (MT), corpus construction, the specification of linguistic formalisms and—as we explore in this thesis—parsing.

3.4.1 Exploiting multi-word units in parsing

It is often assumed that MWUs are useful in syntactic parsing as a means of resolving certain types of ambiguous cases, and indeed some parsing architectures attempt to exploit knowledge about limited classes of MWUs such as compound nouns. Despite this, there is a surprisingly small amount of research reported in the literature that puts the conjecture to the test (see Chapter 6 for a summary). In particular, we are not aware of any comprehensive work that investigates the utility of recognising different types of MWUs in data-driven, statistical approaches to syntactic parsing such as Collins' models.[4]

Identifying MWUs for parsing should permit us to partially address the problem of syntactic ambiguity at the lexical level. We can regard this as a means of reducing the effective number of word tokens, thereby reducing the overall complexity. Alternatively, it can be seen as a way of incorporating a form of additional (i.e. non-treebank) information into the parsing model. From either standpoint, one would reasonably expect that flagging MWUs prior to (or as part of) the parsing task should help in resolving ambiguous cases.

Figures 3.2 and 3.3 illustrate an example of how determining the boundaries of MWUs might improve parsing. Figure 3.2 gives a potential erroneous parse for the sentence "*In the meantime job losses continued*". If we can identify the MWU "*In the meantime*" and disallow hypotheses containing syntactic units (phrases) that overlap with this MWU (such as those that include the possible noun phrase "*the meantime job*") we are more likely to achieve the correct parse (Figure 3.3). A parser that respects the boundaries of MWUs in this manner could, in particular, go some way towards addressing the well-documented stumbling blocks of preposition-phrase attachment and co-ordination.

Figure 3.4 shows a hypothetical example where an MWU—the named entity "*People's Republic of China*"—creates a troublesome PP-attachment case. Here, this results in an erroneous parse tree where the preposition phrase "*of China*" is attached

[4] the exception being Nivre and Nilsson's (2004) work on MWUs in dependency parsing.

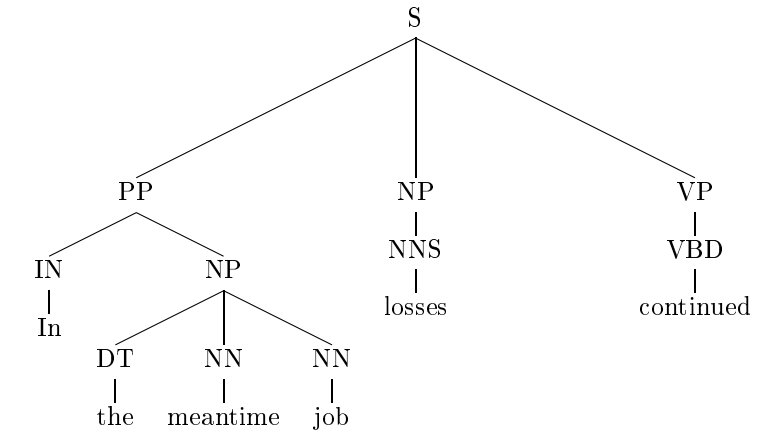

Figure 3.2: A potential erroneous parse tree for the sentence "*In the meantime job losses continued*". Here the boundary of the preposition phrase—the MWU "*In the meantime*"—has been incorrectly determined (the phrase wrongly includes the noun "*job*").

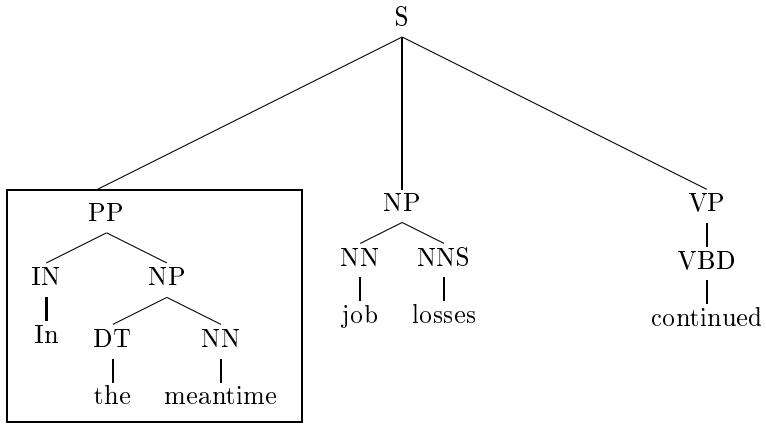

Figure 3.3: If we have identified the MWU "*In the meantime*" and, accordingly, we disallow phrases whose spans overlap with that of this unit, the correct parse tree is yielded.

at the wrong level. A parser made suitably "aware" of this class of MWUs should yield the correct parse tree shown in Figure 3.5.

As a final example, Figure 3.6 illustrates a potential erroneous co-ordination of the noun phrases in the sentence "*The Securities and Exchange Commission made an unequivocal ruling*". If we can identify the named entity "*Securities and Exchange Commission*" this previously ambiguous case becomes quite trivial, yielding the correct parse shown in Figure 3.7.

In the next chapter we will investigate a number of ways of putting knowledge about MWUs into practice in some of the statistical approaches to syntactic parsing that we have described in Chapter 2.

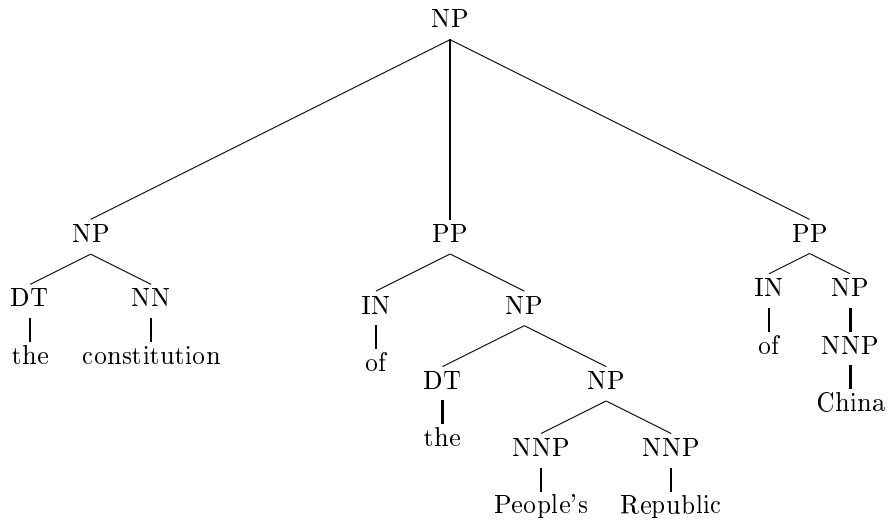

Figure 3.4: A potential erroneous parse tree for the noun phrase *"the constitution of the People's Republic of China"*, where the preposition phrase "*of China*" is incorrectly attached.

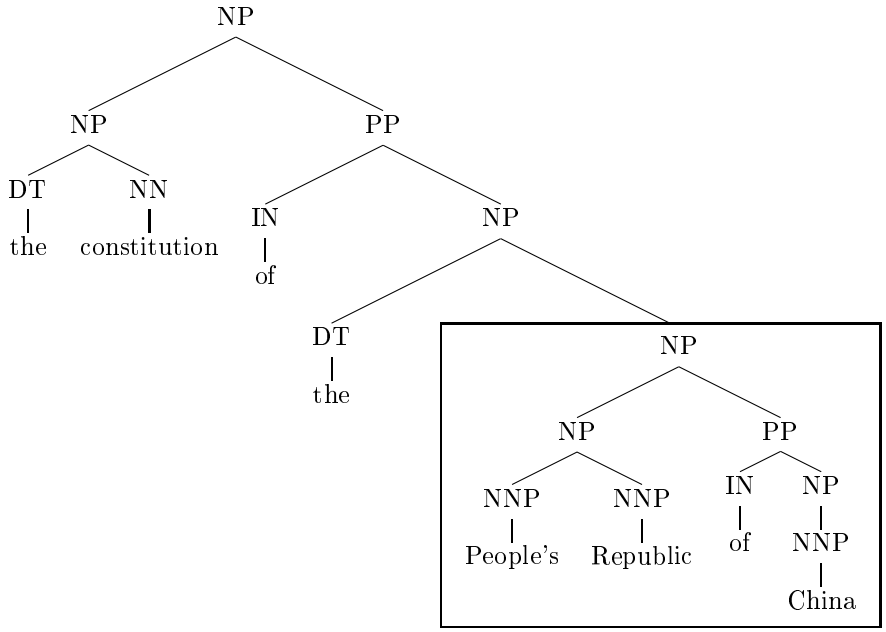

Figure 3.5: Identifying the MWU *"People's Republic of China"* simplifies the case in Figure 3.4, yielding a correct parse with the desired PP attachment.

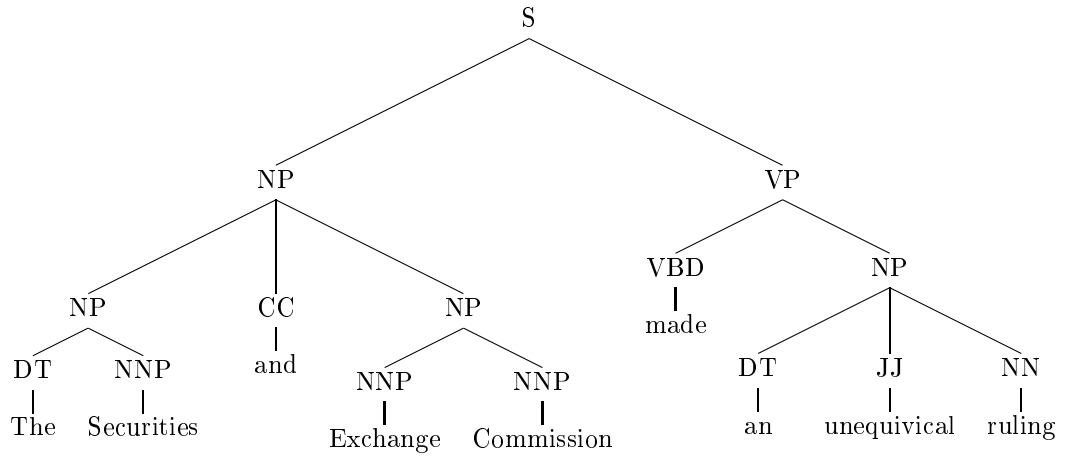

Figure 3.6: A potential erroneous parse tree for the sentence "*The Securities and Exchange Commission made an unequivocal ruling*". Here, the single unit "*Securities and Exchange Commission*" has been incorrectly parsed as two separate entities: "*Securities*" and "*Exchange Commission*".

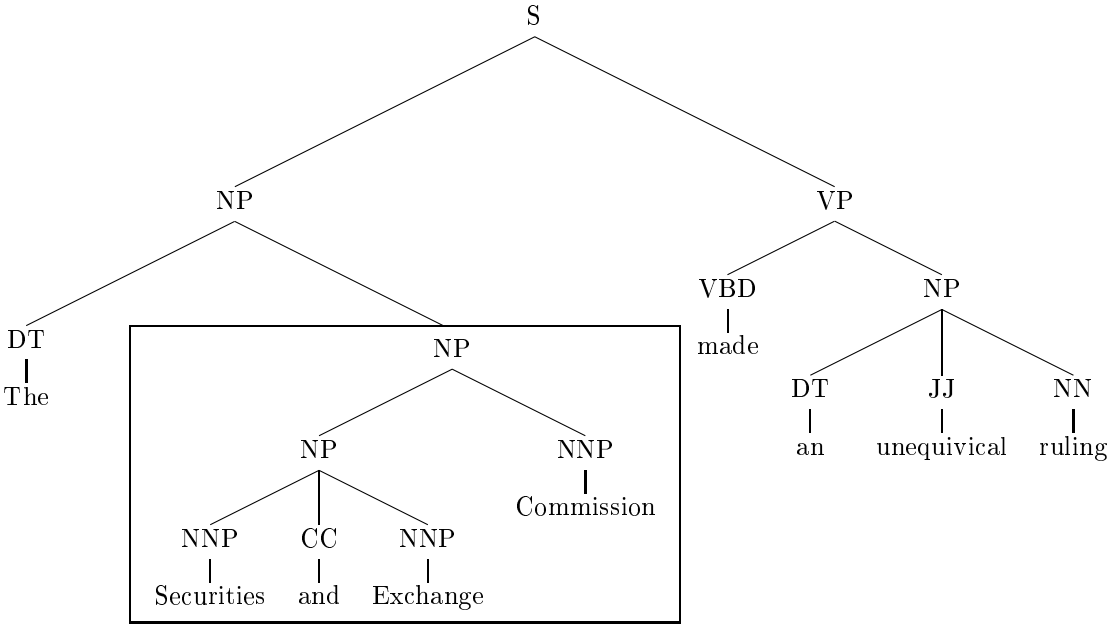

Figure 3.7: If we have flagged the multi-word unit "*Securities and Exchange Commission*" (an ENAMEX-type named entity), the correct parse is more forthcoming.

Chapter 4

Parsing with Multi-Word Units

In this chapter we describe a set of experiments designed to examine the impact of several approaches to exploiting multi-word units (MWUs) in syntactic constituency parsing. We provide a brief recap of our motivation, our core hypotheses and expectations. We then describe our experimental setup, our results and observations, and discuss some of the implications of our work. Although the small gains we achieve fall short of our initial optimistic expectations, they are consistent and demonstrably significant.[1]

4.1 Introduction

In Chapter 2 we discussed a number of architectures for statistical approaches to syntactic constituency parsing. We have also described (in Chapter 3) the concept of multi-word units and the problems that they pose in a variety of natural language processing (NLP) tasks. We speculated that integrating knowledge about MWUs in a statistical parsing model could yield improvements in both efficiency and accuracy. Specifically, we hope that in determining the boundaries of these fundamental units we can populate the parse chart with certain "ground truths" or "islands of certainty". Here we will put our core hypotheses to the test though a number of experiments.

[1] Parts of the work presented in this chapter have been published in Cafferkey et al. (2007).

4.2 Parsing architectures

We perform experiments using three different parsing architectures: a "vanilla" probabilistic context-free grammar (PCFG) parser; a history-based, lexicalised parsing model; and a PCFG with latent annotations.

For each of three parsers we use sections 02-21 of the Penn Wall Street Journal (WSJ) Treebank (39,832 sentences) as our training data, section 24 (1,346 sentences) as our development set, and section 23 (2,416 sentences) for our final evaluations.

4.2.1 "Vanilla" PCFG

The probabilistic context-free grammar is a fundamental concept in statistical approaches to natural language parsing and is the basis for other, more complex parsing architectures. As such, it represents a good starting point from which we can evaluate the effects of introducing the concept of MWUs in statistical parsing

For our PCFG-based experiments we use the BitPar parser (Schmid, 2004). This is an efficient implementation of a slightly modified version of the Cocke-Younger-Kasami (CYK) algorithm.

Our baseline grammar is extracted from the training data (§02-21 of the corpus) using the well-known procedure described by Charniak (1997). Each PCFG production rule and its corresponding probability are simply read off from the WSJ syntax trees such that:

$$P(A \rightarrow \beta | A) = \frac{\text{count}(A \rightarrow \beta)}{\text{count}(A)} \tag{4.1}$$

Prior to extracting our grammar, we perform the following commonly-used preprocessing steps on the WSJ corpus data (see also Figure 4.1):

- add a "TOP" node to each sentence
- remove null terminals / trace annotations
- remove functional sub-labels

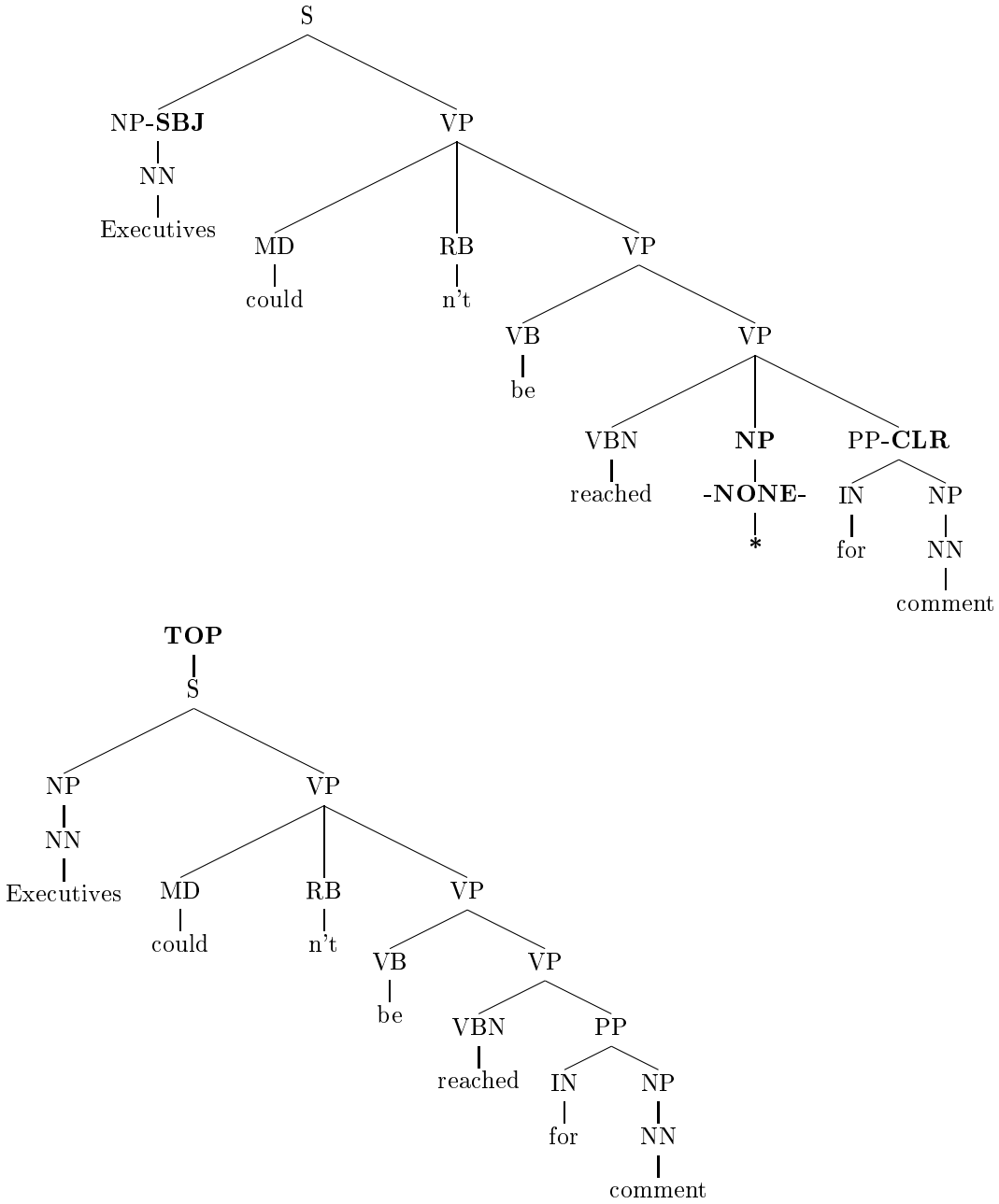

Figure 4.1: Preprocessing performed on WSJ corpus trees prior to extracting a PCFG grammar. We add a "TOP" node, and remove null terminals and functional sublabels (WSJ §02-21, tree 1172).

This results in a baseline grammar with 14,972 production rules[2], achieving a labelled bracketing f-score of 72.82 when evaluated on §23 of WSJ corpus (see Section 4.6; Table 4.4).

4.2.2 History-based, lexicalised model (Collins model 2)

Collins' (1999) describes three history-based, lexicalised generative parsing models (for a more in-depth discussion see Section 2.6.1). The most well known of these (model 2) has been widely deployed in a range of NLP applications.

For our experiments we use Bikel's (2002) parsing engine to emulate Collins' model 2 parser. We use Bikel's implementation of the Collins model as it is highly modular and more readily retrained and adjusted than Collins' original implementation. The Bikel parser also implements a framework for performing constrained parsing that we will make use of in Section 4.4.2.2.

When trained and evaluated on the WSJ treebank the Bikel parser baseline achieves an f-score of 88.66 (see Section 4.6; Table 4.4).

4.2.3 PCFG with latent annotations (Berkeley parser)

We use the Berkeley parser (Petrov et al., 2006) as an additional testbed for hypotheses. This parser uses an expectation maximisation (EM) technique to induce latent annotations during training of a PCFG-type parser (for further details see Section 2.4.2). The Berkeley parsing model represents the state of the art in statistical approaches to natural language parsing. In contrast to Collins' parsing model, the parser is unlexicalised.

For all experiments we use the parser's default settings for training and set thresholds for accuracy during parsing. When trained and evaluated on the WSJ treebank our baseline achieves an f-score of 90.06 (see Section 4.6; Table 4.4).

[2] We remove cyclical rules of the form $X \rightarrow X$ (a consequence of removing null terminals) from our grammar.

Source	Avg. MWUS per sent.	% words part of MWU	Avg. MWU length
BBN	1.22	4.79	2.64
NER	0.6	2.36	2.38
MWE	0.19	0.73	2.31

Table 4.1: MWU frequency and average length as they occur in §02-21 of the WSJ treebank, per MWU source.

4.3 MWU sources

We experiment with three different sources of MWU data: gold-standard named entity (NE) data from the BBN Pronoun Coreference and Entity Type Corpus; the named entity recognition (NER) system of Chieu and Ng (2003); and a dictionary of preposition-phrase expressions derived from the British National Corpus (BNC)[3]. Table 4.1 shows the frequency in the WSJ treebank of the MWUs from each source as well as the average number of word tokens per MWU as they occur in the treebank.

4.3.1 BBN corpus

As described in Chapter 3, the BBN Entity Type and Coreference Corpus (Weischedel and Brunstein, 2005) supplements the Penn WSJ Treebank with additional annotations of various classes of NEs: covering name expressions, number expressions and time expressions. Since the BBN corpus represents highly-comprehensive, gold-standard NE data we should be able to establish an approximate upper bound for the utility of exploiting multi-word NEs in syntactic parsing.

Before we could use the BBN corpus a certain amount of data clean-up was required: we encountered inconsistencies in tokenisation of the corpus versus the original WSJ treebank (these were in general systematic); and in sentence segmentation versus the original treebank (which, although for the most part systematic, were often erratic). We also fixed a number of erroneously-labelled entities as we came upon them.

The BBN NEs are the most plentiful of the MWU sources that we use, with an average of 2 MWUs per sentence (Table 4.1). Appendix C provides a list of the most

[3] Based on a resource from `http://mwe.stanford.edu/`

common NEs identified in the BBN corpus.

4.3.2 Chieu and Ng's (2003) NER system

In Chapter 3 we described the Chieu and Ng (2003) named entity recogniser. This NER system identifies NEs under four categories: persons (denoted PER), locations (LOC), organisations (ORG), and miscellaneous (MISC). These are roughly equivalent to the ENAMEX-type NEs identified in the BBN corpus. As we noted, the system achieves close to 90% recall and precision on the CoNLL-2003 test set (Reuters Corpus data).

The NEs identified by Chieu and Ng's system in the WSJ treebank are not as plentiful as the BBN corpus NEs (Table 4.1) and will obviously include a certain amount of noise. Appendix D provides a list of the 100 most common NEs identified by Chieu and Ng's system for the WSJ corpus.

4.3.3 Dictionary of multi-word expressions

As our final MWU source, we use a dictionary of candidate preposition-phrase multi-word expressions obtained from the Stanford Multi-Word Expression Project[4]. The list was derived semi-automatically from the British National Corpus (BNC).

Additionally, the BNC—in contrast to the WSJ Treebank—includes a set of MWUs that are treated as "words with spaces" (these include compound function words, foreign-language terms). We combine this set with the dictionary of preposition-phrase MWU candidates.

The units in our MWU dictionary are less plentiful in the WSJ corpus than those from the other sources, with an average of 0.19 MWUs per sentence (Table 4.1). Appendix E provides a list of the 100 most common such expressions occurring in the WSJ corpus.

[4]The list can be downloaded from `http://mwe.stanford.edu`

```
Diamond/NNP Shamrock/NNP Offshore/NNP 's/POS stock/NN
rose/VBD 12.5/CD cents/NNS Friday/NNP to/TO close/VB at/IN
$/$ 8.25/CD in/IN New/NNP York/NNP Stock/NNP Exchange/NNP
composite/NN trading/NN ./.
```

Figure 4.2: A sequence of part-of-speech annotated word tokens given as input to a parser.

```
Diamond_Shamrock_Offshore/NNP 's/POS stock/NN rose/VBD
12.5_cents/NNS Friday/NNP to/TO close/VB at/IN $_8.25/$
in/IN New_York_Stock_Exchange/NNP composite/NN trading/NN ./.
```

Figure 4.3: The parser input string from Figure 4.2 after retokenisation, featuring "words with spaces".

4.4 Integrating MWUs into parsing

We explore two general approaches to integrating MWUs in the parsing task: on the one hand, using a given parsing architecture "as is" and retokenising the training and test data such that MWUs are treated as single word tokens (Cafferkey et al., 2007); and on the other, prebracketing the training and test data such that a modified parser imposes phrase-boundary constraints on the parsing chart (cf. Glaysher and Moldovan, 2006).

It should be noted that, for our purposes, we are not overly concerned with differences between the several classes of MWUs used in our experiments—we merely exploit the fact that MWUs can be treated as atomic units, or as a means of imposing bracketing constraints, in parsing.

4.4.1 Corpus retokenisation

For the retokenisation approach we simply concatenate sequences of word tokens that have been flagged as MWUs—by, for example, an NER system—into single "words with spaces", whereby each new "word" assumes the POS tag of its head constituent[5]. Figure 4.2 shows a part-of-speech annotated sequence of word tokens to be passed to a parser; following MWU-retokenisation the sequence might resemble that in Figure 4.3,

[5] determined using Collins' (1999) head finding heuristics

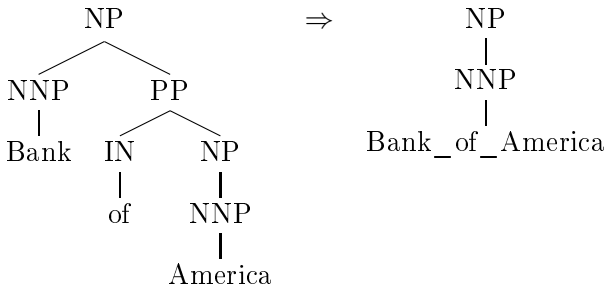

Figure 4.4: A retokenised training tree fragment. The entity "*Bank of America*" is treated as a single "word with spaces" and the corresponding syntax tree collapsed.

Grammar	# PCFG rules	Tokens	Types
baseline	14,972	950,028	44,389
BBN (all entity types)	13,793	870,484	57,221
NER (Chieu and Ng, 2003)	14,385	917,146	50,551
MWE (`mwe.stanford.edu`)	14,844	944,161	45,475
BBN + MWE	13,680	862,038	58,384
NER + MWE	14,256	907,413	51,413

Table 4.2: Number of PCFG production rules per retokenised corpus (WSJ §02-21): with BBN corpus entities; entities from Chieu and Ng's NER system; multi-word expressions from dictionary lookup; and MWE dictionary combined with BBN and NER, respectively.

where the NEs "*Diamond Shamrock Offshore*" (a name expression), *"12.5 cents"* and "*$ 8.25*" (number expressions), and "*New York Stock Exchange*" (a name expression) have become single tokens.

One question which we will explore in this approach is whether we should merely retokenise the test data (leaving the training data unaltered) or perform retokenisation on both. In the latter case, nodes in the training trees that dominate an MWU will be collapsed (Figure 4.4). This brings about a corresponding reduction in the number of production rules in a PCFG grammar (Table 4.2) as well as a decrease in the number of lexical tokens (together with an increase in the number of lexical types).

4.4.1.1 Re-inserting MWU subtrees

The retokenisation approach throws up a problem when we wish to perform evaluation: there will be fewer word tokens in the parser output than in the baseline parse.

```
( Diamond/NNP Shamrock/NNP Offshore/NNP ) 's/POS stock/NN
rose/VBD ( 12.5/CD cents/NNS ) Friday/NNP to/TO close/VB
at/IN ( $/$ 8.25/CD ) in/IN ( New/NNP York/NNP Stock/NNP
Exchange/NNP ) composite/NN trading/NN ./.
```

Figure 4.5: The parser input string from Figure 4.2 with prebracketed MWUs.

This means that we cannot draw an unbiased comparison with the baseline parse for a given sequence. To facilitate such a comparison we perform a supplementary experiment in which we re-expand collapsed MWU tokens after parsing with the corresponding gold tree fragment (the converse operation to that illustrated in Figure 4.4). This allows us to evaluate against our baseline with confidence, and to perform statistical significance testing (Section 4.7).

4.4.2 Constrained parsing

For the constrained parsing approach, we leave the WSJ corpus data unaltered and instead modify the parser itself such that it honours the syntactic boundaries of sequences of word tokens that have been flagged as MWUs[6]. A suitably modified parser will accept partially prebracketed input such as that in Figure 4.5.

In this scheme, the modified parser treats the input bracketings as constraints on the spans that are permitted to be added to the parsing chart. This seems an intuitively more satisfactory approach than the retokenisation method that we have discussed, since we maintain the original tokenisation of the training and test data (preserving the full lexicon, and production rule set). We perform experiments to determine whether we do in fact achieve better results.

4.4.2.1 Constrained parsing with PCFG

The changes to a CYK-type parser (such as BitPar) required to implement constrained parsing are quite straightforward: we modify the parser such that when an edge (span) that has been proposed is inconsistent with the constrained chart—that is, it overlaps with one or more constraint spans—we discard it and do not explore

[6] For a comparison of our approach to constrained parsing with related work, see Chapter 6.

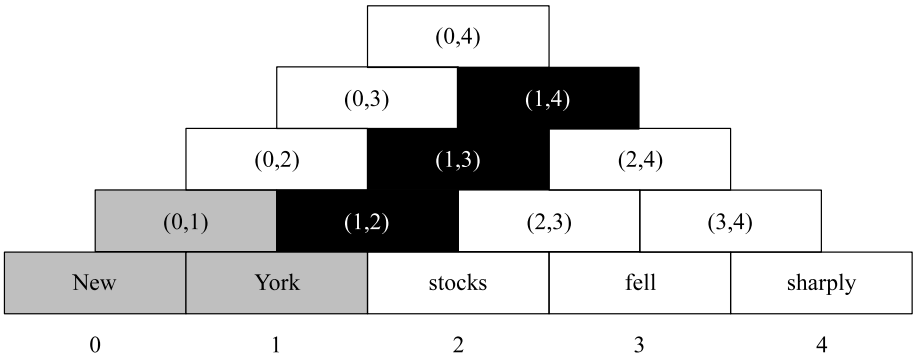

Table 4.3: A constrained CYK chart for the string "*New York stocks fell sharply*". Grey cells represent constraint spans. Hypotheses that include spans represented by the black cells will not be explored; for example, the possible noun phrase "*York stocks*" at span $(1, 2)$ is disallowed.

any hypotheses dependent on that span. Following the notation of Glaysher and Moldovan (2006), we define a span as pair $c = (s, t)$ where s is the index of the first word in the span and t is the index of the last word. Two spans (c_1 and c_2) are said to be overlapping iff $s_1 < s_2 \leq t_1 < t_2$ or $s_2 < s_1 \leq t_2 < t_1$. The pseudocode for the necessary modification to the parser can be found in Glaysher and Moldovan's paper[7].

Figure 4.3 illustrates the parse chart for the sentence *"New York stocks fell sharply"*, represented as a pyramid where each cell refers to a span (s, t). The greyed-out cells represent constraint spans, while the blackened cells are those whose spans overlap with constraint spans. Analyses that include the spans in the blackened cells are prohibited and thus only derivations consistent with the MWU bracketings can be generated.

4.4.2.2 Bikel parser constraints

Bikel's parser has a built-in constraints framework that allows us use prebracketed input. We modify the `PartialTreeConstraint` class that ships with the parser such that it enforces bracketing constraints in the same manner as the constrained PCFG

[7] They refer specifically to the `parse()` function of Collins' (1999) parser, but the approach generalises to the plain PCFG architecture.

parser above.

4.4.2.3 Berkeley parser

No facility is available in the Berkeley parser to allow prebracketed input (and we did not attempt to modify the parser to allow this). We are therefore unable to present results for constrained parsing using this architecture.

4.5 Other design considerations

In all cases we assume gold part-of-speech (POS) tags; that is, we use the tags assigned in the WSJ corpus.

We disregard MWUs that cross the original Penn bracketings. Also, we disregard MWUs that contain null terminals.

4.6 Experimental results

The performance of each parser (trained on §02-21 of the treebank) for each of our experiments was evaluated against §23 using the PARSEVAL bracketing recall and precision measures (Black et al., 1992). This was performed using the standard tool `evalb`[8]. Bracketing recall and precision are defined as follows:

$$\text{recall} = \frac{\#\text{ correct constituents in test parse}}{\#\text{ constituents in gold parse}} \tag{4.2}$$

$$\text{precision} = \frac{\#\text{ correct constituents in test parse}}{\#\text{ constituents in test parse}} \tag{4.3}$$

The results presented in this thesis are for labelled recall and precision; that is, we consider the syntactic category label assigned to each constituent in addition to its span. We also calculate the harmonic mean of the two (the f-score):

[8] available at: `http://nlp.cs.nyu.edu/evalb/`

Parser	Recall	Precision	F-score
PCFG (BitPar)	70.28	75.56	72.82
Bikel parser (Collins model 2)	88.61	88.71	88.66
Berkeley parser	90.03	90.31	90.17

Table 4.4: Baseline performance (labelled bracketing recall, precision and f-score) for the three parsing architectures: "vanilla" PCFG; history-based, lexicalised model; and PCFG with latent annotations (WSJ §23).

$$\text{f-score} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \tag{4.4}$$

It is standard practice to ignore punctuation tokens in evaluation and, for the sake of comparison with other work, we do likewise. We consider all sentence lengths. The baseline results for each of the parsing architectures is given in Table 4.4.[9]

4.6.1 Parsing with the retokenised corpus

For the retokenisation approach we will present results for the case in which we retokenise both the test and training data as well as those for retokenising the test data alone. We do not present results where the grammar gives less than full coverage. We also present scores for the case in which we re-expand collapsed MWUs after parsing—this provides a means of more accurately comparing our results against the baseline (see Section 4.4.1.1).

4.6.1.1 PCFG

Table 4.5 gives our results for parsing with a plain PCFG and MWU-retokenised data. The first row gives the results for our baseline grammar (as described in Section 4.2.1); the second gives the results of integrating MWUs identified by Chieu and Ng's NER system; the third gives the results of integrating MWUs from our dictionary of multi-word expressions; the fourth gives the results of integrating the ENAMEX type MWUs identified by the BBN corpus (see Table 4.6 for other MWU types identified in the BBN corpus); the final two rows give the results of combining the MWUs from the

[9] The baseline and subsequent results for Bikel's implementation of Collins' model 2 are different from those reported in Cafferkey et al. (2007) as we use the parser in gold POS mode for this thesis.

Source	Test + train retokenised			Test only retokenised		
	Recall	Precision	F-score	Recall	Precision	F-score
baseline	70.28	75.56	72.82	**70.28**	75.56	72.82
NER	70.27	75.75	72.91	70.19	75.75	72.87
MWE	70.13	75.53	72.73	69.72	75.21	72.36
BBN (ENAMEX only)	**70.39**	**75.93**	**73.05**	70.19	**75.81**	**72.90**
NER + MWE	70.11	75.72	72.81	69.64	75.40	72.40
BBN (ENAMEX) + MWE	70.25	75.90	72.96	69.67	75.49	72.46

Table 4.5: Plain PCFG results for retokenised corpus (WSJ §23).

dictionary of multi-word expressions with those derived from the NER system and the BBN corpus, respectively.

We achieve modest improvements in all cases except for that in which we retokenise based on our dictionary of multi-word expressions—which, in fact, proves detrimental to parsing performance. In each case, retokenising both the test and training data yields greater improvements than retokenising the test data alone. Our best result, with an f-score of 73.05, is achieved by retokenising both the test and training data based on the ENAMEX type MWUs identified in the BBN corpus.

When we look individually at the different classes of NEs identified in the BBN corpus—name expressions (ENAMEX), number expressions (NUMEX) and time expressions (TIMEX)—we observe that retokenising based on the NUMEX and TIMEX classes is detrimental to parsing performance (in contrast to the ENAMEX class, which yields the best overall results). We will investigate the reasons for this in Section 4.8.

The results obtained when, after parsing with the retokenised corpus, we re-expand all tokens that have been concatenated with their corresponding gold tree fragment are given in Table 4.7. The table follows the same format as Table 4.5 above, save that we only present results for the ENAMEX class of NEs from the BBN corpus (we have already seen that the other NE types are not useful in this retokenisation approach). The baseline is the same as that in Table 4.5.

BBN entity type	Test + train retokenised Recall	Precision	F-score	Test only retokenised Recall	Precision	F-score
ENAMEX	**70.39**	**75.93**	**73.05**	**70.19**	**75.81**	**72.90**
NUMEX	69.73	75.25	72.39			
TIMEX	70.05	75.46	72.66	70.07	75.54	72.70
all	69.80	75.65	72.61			

Table 4.6: Breakdown of BBN entity types for plain PCFG (WSJ §23).

Source	Test + train retokenised Recall	Precision	F-score	Test only retokenised Recall	Precision	F-score
baseline	70.28	75.56	72.82	**70.28**	75.56	72.82
NER	70.29	75.77	72.93	70.21	75.77	72.89
MWE	70.30	75.69	72.89	69.90	75.37	72.53
BBN (ENAMEX only)	70.45	75.99	73.12	70.26	**75.87**	**72.96**
NER + MWE	70.30	75.90	72.99	69.84	75.57	72.59
BBN (ENAMEX) + MWE	**70.49**	**76.11**	**73.19**	69.91	75.70	72.69

Table 4.7: Plain PCFG results for retokenised corpus with gold MWU subtrees re-inserted (WSJ §23).

4.6.1.2 Bikel parser

Table 4.10 gives our results for integrating MWUs with the retokenisation approach using the Bikel parser in Collins model 2 emulation mode. The first line is our baseline as described in Section 4.2.2. The remainder of the table follows the same format as that for our PCFG experiments except that, based on our previous results, we only include the ENAMEX class of NEs from the BBN corpus. Overall we achieve small gains similar in relative proportion to those observed in our plain PCFG experiments with our best result, an f-score of 88.82, obtained by retokenising both the training and test based on the BBN corpus name expressions.

Table 4.5 gives the results for the re-insertion experiment (for comparison with the baseline). Our best result here is a f-score of 88.85 when we retokenise the both the training and test based on the BBN corpus name expressions combined with the MWE dictionary.

Source	Test+train retokenised Recall	Precision	F-score	Test only retokenised Recall	Precision	F-score
baseline	88.61	88.71	88.66	**88.61**	88.71	88.66
NER	88.67	88.76	88.71	88.60	88.63	88.62
MWE	88.62	88.63	88.63	87.84	88.03	87.94
BBN (ENAMEX only)	**88.75**	**88.88**	**88.82**	88.60	**88.73**	**88.67**
NER + MWE	88.73	88.72	88.73	87.81	88.02	87.91
BBN (ENAMEX) + MWE	**88.75**	88.77	88.76	87.82	88.16	87.99

Table 4.8: Results for Bikel parser with retokenised corpus (WSJ §23).

BBN entity type	Test + train retokenised Recall	Precision	F-score	Test only retokenised Recall	Precision	F-score
ENAMEX	**88.75**	**88.88**	**88.82**	**88.60**	**88.73**	**88.67**
NUMEX	88.41	88.80	88.60			
TIMEX	88.49	88.63	88.56	87.87	88.05	87.96
all	88.50	88.86	88.68			

Table 4.9: Breakdown of BBN entity types for Bikel parser (WSJ §23)

Source	Test+train retokenised Recall	Precision	F-score	Test only Recall	Precision	F-score
baseline	88.61	88.71	88.66	88.61	88.71	88.66
NER	88.67	88.76	88.72	88.61	88.64	88.63
MWE	88.69	88.70	88.69	87.91	88.10	88.01
BBN (ENAMEX only)	88.77	**88.91**	88.84	88.62	88.76	88.69
NER + MWE	88.80	88.79	88.80	87.89	88.10	88.00
BBN (ENAMEX) + MWE	**88.84**	88.86	**88.85**	87.92	88.26	88.09

Table 4.10: Bikel parser results for retokenised corpus, with gold MWUs subtrees re-inserted (WSJ §23).

Source	Test+train retokenised			Test only retokenised		
	Recall	Precision	F-score	Recall	Precision	F-score
baseline	**90.03**	90.31	**90.17**	**90.03**	90.31	**90.17**
NER	89.71	90.20	89.95	89.83	90.31	90.07
MWE	89.82	90.11	89.97	88.71	89.14	88.93
BBN (ENAMEX only)	89.93	**90.33**	90.13	89.87	**90.39**	90.13
NER + MWE	89.51	89.96	89.73	88.47	89.09	88.78
BBN (ENAMEX) + MWE	89.72	90.19	89.95	88.48	89.15	88.82

Table 4.11: Berkeley parser with retokenised corpus (WSJ §23)

4.6.1.3 Berkeley parser

Table 4.11 shows our results for the retokenisation approach with the Berkeley parser. The first row is our baseline as described in Section 4.2.3, the remainder of the table follows the same format as those for the PCFG and Bikel parsers above. Here, parsing with the retokenised corpus in fact had, in most cases, a negative effect on performance. Since we did not observe any discernible gains we do not present results for re-inserting the gold MWU subtrees.

4.6.2 Constrained parsing

For the constrained parsing approach we present the results for only the PCFG and Bikel parsers (as previously noted, we did not attempt to implement constrained parsing with the Berkeley parser).

4.6.2.1 PCFG

Table 4.12 shows our results when we do constrained parsing using the PCFG parser modified as per Section 4.6.2.1. The first row is our baseline (the same as that from the previous PCFG experiments); the second is when we impose constraints on the parse chart based on the NER system; the third based on the MWE dictionary; the fourth based on the BBN corpus NEs; and the final two rows are when we combine the MWE dictionary with the NER system and the BBN corpus, respectively. As with the retokenisation experiments, we also examine the individual contributions of the different classes of NEs identified in the BBN corpus (Table 4.13). Our best improve-

Source	Recall	Precision	F-score
baseline	70.28	75.56	72.82
NER	70.45	75.68	72.97
MWE	70.35	75.65	72.90
BBN	70.74	75.93	73.24
NER+MWE	70.52	75.77	73.05
BBN+MWE	**70.81**	**76.01**	**73.32**

Table 4.12: Plain PCFG with constrained parsing (WSJ §23).

BBN entity type	Recall	Precision	F-score
ENAMEX	70.50	75.74	73.03
NUMEX	70.38	75.65	72.92
TIMEX	70.41	75.66	72.94
all	**70.74**	**75.93**	**73.24**

Table 4.13: Breakdown of BBN entity types for constrained PCFG (WSJ §23).

ment was achieved by constraining the parser based on the BBN corpus NEs combined with the dictionary of multi-word expressions, achieving an f-score of 73.32). This was the largest overall relative increase observed across all of our experiments (see Section 4.8).

4.6.2.2 Bikel parser

Table 4.14 shows the Bikel parser (Collins model 2) using the span constraint that we implemented (Section 4.4.2.2). The baseline is the same as that in the earlier Bikel parser experiments and, as before, we provide a breakdown of the influence of the different classes of NEs from the BBN corpus (Table 4.15). Our best improvement was achieved by constraining the parser based on the BBN corpus NEs combined with the dictionary of multi-word expressions, yielding an f-score of 88.84.

MWU source	Recall	Precision	F-score
baseline	88.61	88.71	88.66
NER	88.52	88.58	88.55
MWE	88.67	88.82	88.74
BBN	88.70	88.86	88.78
NER + MWE	88.58	88.69	88.64
BBN + MWE	**88.74**	**88.94**	**88.84**

Table 4.14: Bikel parser with constrained parsing (WSJ §23).

BBN entity type	Recall	Precision	F-score
ENAMEX	88.65	88.78	88.72
NUMEX	88.63	88.76	88.70
TIMEX	88.63	88.74	88.69
all	**88.70**	**88.86**	**88.78**

Table 4.15: Breakdown of BBN entity types for constrained Bikel parser(WSJ §23).

Experiment	F-score	Base f-score	% error reduction	p-value
1. Constr. PCFG, BBN + MWE	73.32	72.82	1.84	0.00009
2. Constr. Bikel, BBN + MWE	88.84	88.66	1.59	0.00009
3. Constr. PCFG, BBN	73.24	72.82	1.55	0.00009
4. Retoke. Bikel, BBN (ENAMEX)	88.82	88.66	1.41	0.00099
5. Constr. Bikel BBN	88.84	88.66	1.06	0.00009

Table 4.16: Top five largest relative reductions in error

4.7 Statistical significance

We performed statistical significance testing using a stratified shuffling test with 10,000 iterations.[10] Despite the modest scale of the improvements, most were found to be highly statistically significant (typically to a level of $p \leq 0.0001$). We give p-values for the five best-performing experiments in Table 4.16.

4.8 Further discussion

4.8.1 Overview

Across the different parsing architectures, MWU sources, and approaches to handling MWUs, we achieved visible gains. However these improvements tended to be very small, falling short of our initial optimistic expectations. Notwithstanding, the results are by and large consistent with our intuitions about each given experiment and, we believe, vindicate our argument for the identification of MWUs to inform parsing. Table 4.16 gives the five experiments that yielded the best improvements over the respective baselines. For each we give the f-score, baseline f-score, percentage error reduction and p-value.

[10] available from `http://www.cis.upenn.edu/~dbikel/software.html`

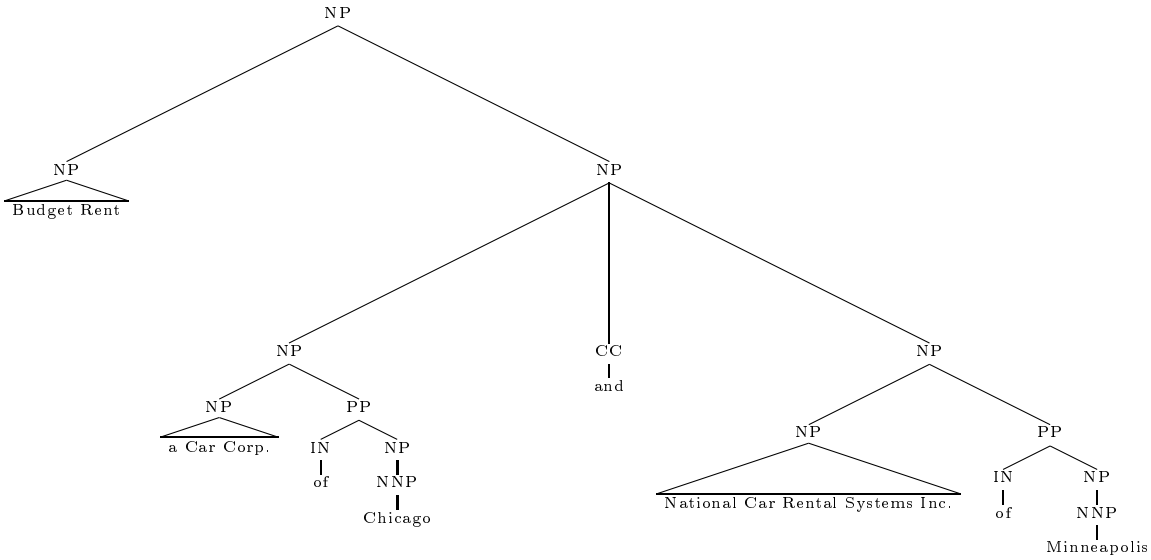

Figure 4.6: Co-ordinated entities containing preposition phrases cause the non MWU-aware Bikel parser to produce an incorrect parse for the sentence fragment "*Budget Rent a Car Corp. of Chicago and National Rental Systems Inc. of Minneapolis*" (WSJ §23, sent. 2088).

Figure 4.6 illustrates a case where the non MWU-aware Bikel parser generates an incorrect parse for the phrase "*Budget Rent a Car Corp. of Chicago and National Rental Systems Inc. of Minneapolis*". The combination of PP-attachment ambiguity and co-ordination makes this phrase particularly difficult for the parser. Since the MWU-aware parser knows that "*Budget Rent a Car Corp.*" and "*National Rental Systems Inc.*" are named entities, this case becomes straightforward and, indeed, the MWU-aware parse generates the correct parse (Figure 4.7).

Figure 4.8 gives an example where the non MWU-informed PCFG parser produces an entirely incorrect parse for the phrase "*Up to now only specific aspects have been challenged*". Exploiting the MWU dictionary lookup, however, the multi-word expression "*Up to now*" is identified and the parser guided to the optimum parse (Figure 4.9).

Knowledge about MWUs doesn't always lead to the correct parse however. In the case of the incorrect parse tree produced by the baseline Bikel parser in Figure 4.10, exploiting MWUs with the constrained parser still yields an erroneous (though different) result (Figure 4.12). The correct parse tree is shown in 4.11.

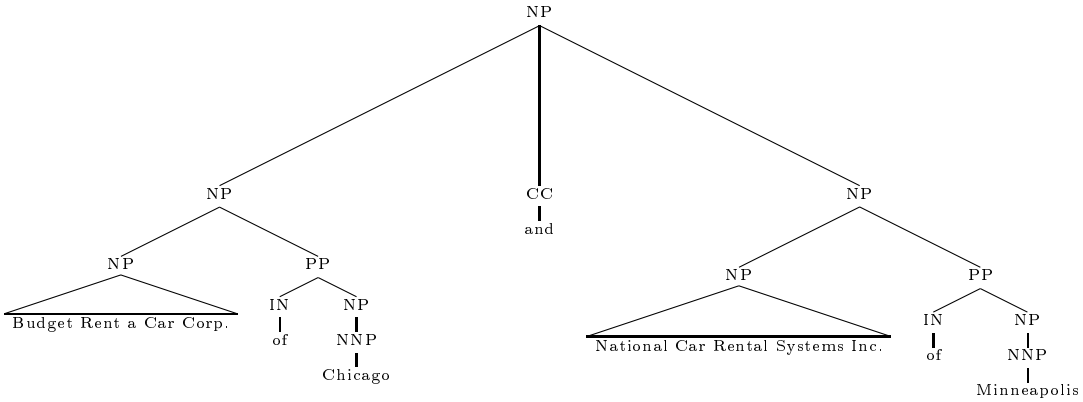

Figure 4.7: Correct parse tree produced by the MWU-aware Bikel parser (constrained parsing with BBN corpus MWUs) for the sentence fragment in Figure 4.6.

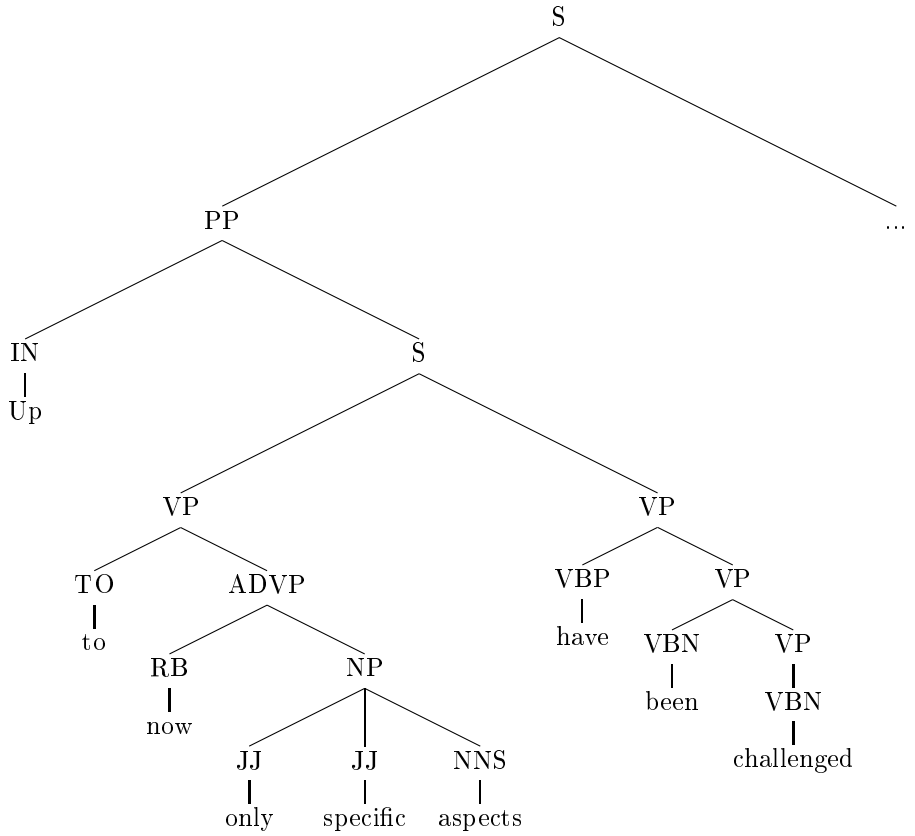

Figure 4.8: Another example, this time from the PCFG parser, where the multi-word expression "*Up to now*" is incorrectly bracketed (WSJ §23, sent. 1989)

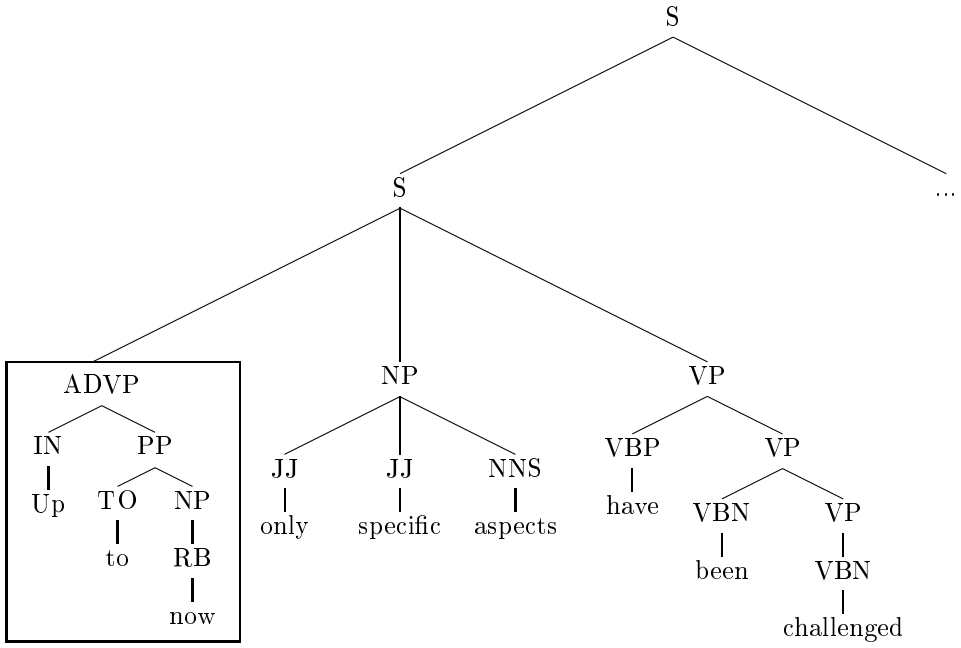

Figure 4.9: Correct parse for the example in Figure 4.8. The MWU *"Up to now"* (highlighted) receives the correct bracketing (PCFG parser with MWE dictionary).

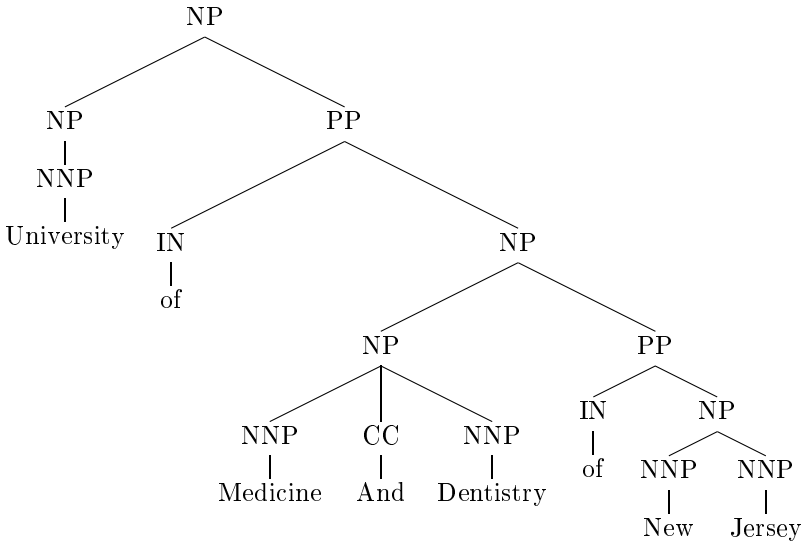

Figure 4.10: Incorrect parse tree for the sentence fragment "*University of Medicine and Dentistry of New Jersey*" produced by the non MWU-aware Bikel parser where the preposition phrase *"of New Jersey"* has been attached at the wrong level (WSJ §23, sent. 2088).

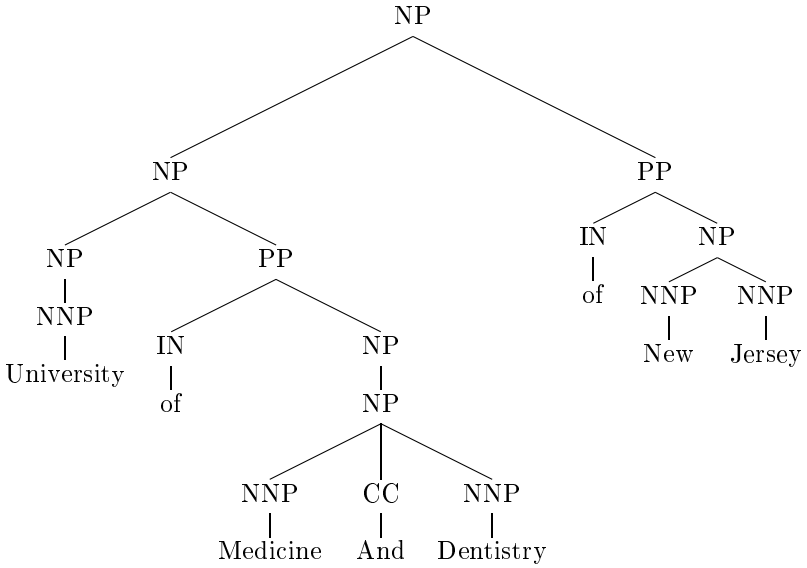

Figure 4.11: Correct parse tree for the sentence in Figure 4.10.

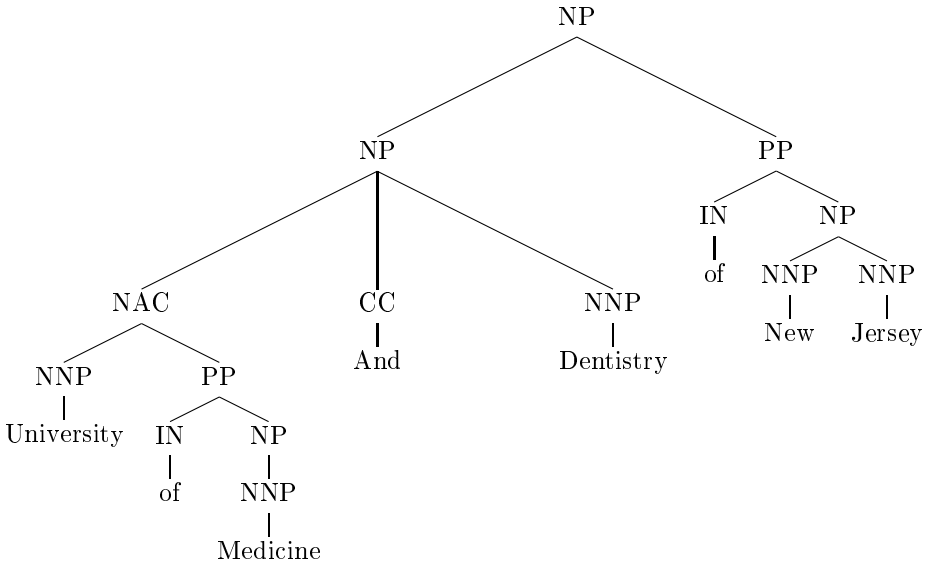

Figure 4.12: Still erroneous (though different) parse tree produced by the MWU-aware Bikel parser (constrained parsing with BBN corpus MWUs)

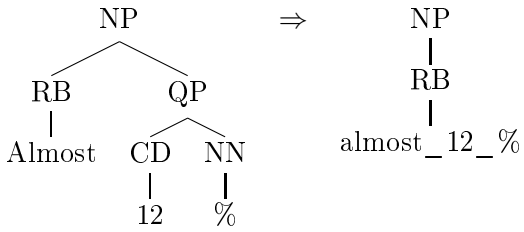

Figure 4.13: Less-than-optimum POS tag ("RB"—adverb) assigned to the concatenated word in the retokenisation approach (a better POS tag in this case might be might be "CD"—cardinal number).

4.8.2 Specific observations

4.8.2.1 Corpus retokenisation versus constrained parsing

Constrained parsing represents an intuitively more satisfactory approach than corpus retokenisation. In the retokenisation approach one of the problems is the difficulty of assigning a new POS. We used Collins' head finding rules which didn't always assign an optimum tag (particularly in the case of the BBN corpus-derived NUMEX and TIMEX entities—see Figure 4.13). Moreover, in the case of the preposition-phrase multi-word units it is in fact usually undesirable to treat them as a single lexical entry.

In the case of corpus retokenisation, it is clear that it is preferable to retokenise both the test and training data. It is evident, however, that the optimum results will be achieved using prebracketing and constrained parsing.

4.8.2.2 BBN corpus

The TIMEX and NUMEX entities are poorly suited to our task; even though we are dealing with gold-standard data. The entity boundaries are often inconsistent with the treebank bracketings. Also, in these cases, our approach often assigns a less than optimum POS tag to the concatenated word units (as above). This would be easily remedied by modifying the heuristics that identify the head constituent word of such MWUs; however, based on our results, using constrained parsing would represent a better option.

4.8.2.3 Bikel parser

Given its lexicalised nature, the Bikel parser will be more susceptible to data sparsity problems brought about by the retokenisation approach. Nonetheless we achieve gains similar in relative proportion to those observed in the plain PCFG experiments.

4.8.2.4 Berkeley parser

Disappointingly, we did not achieve any discernible improvements over the baseline in our experiments with the Berkeley parser. Any 'gains' observed were very small and, in fact, retokenising the corpus usually had a *negative* impact. It is perhaps the case that the node splitting and merging performed in training implicitly adapts the model such that the idiosyncrasies of the MWUs with which we are concerned are already sufficiently accounted for. It might still however be the case that implementing parser-internal constraints could prove fruitful.

Chapter 5

Sentence Generation with MWUs

In the preceding chapter we explored a number of approaches to exploiting multi-word units (MWUs) in statistical approaches to syntactic parsing. Here, as a secondary task we present several experiments where we make use of MWU information in the converse operation to statistical parsing: statistical generation. Specifically, we attempt to exploit information about the MWU types that we have previously seen in the task of sentence generation (or surface realisation) from Lexical-Functional Grammar (LFG) f-structures.[1] In this chapter we will provide a brief introduction to LFG, outline the task of sentence generation and its applications, describe our experimental setup and results obtained, and conclude with a commentary on their implications.

5.1 Lexical-Functional Grammar

Lexical-Functional Grammar (Kaplan, 1995) is a constraint-based theory of grammar, which analyses strings in terms of c(onstituency)-structure and f(unctional)-structure (Figure 5.1). C-structure is defined in terms of CFGs, and f-structures are recursive attribute-value matrices which represent abstract syntactic functions (such as SUBJect, OBJect, OBLique, COMPlement (sentential), ADJ(N)unct), agreement,

[1] This was joint work with Deirdre Hogan. Parts of the research presented in this chapter have been published in Hogan et al. (2007) and Cafferkey et al. (2007).

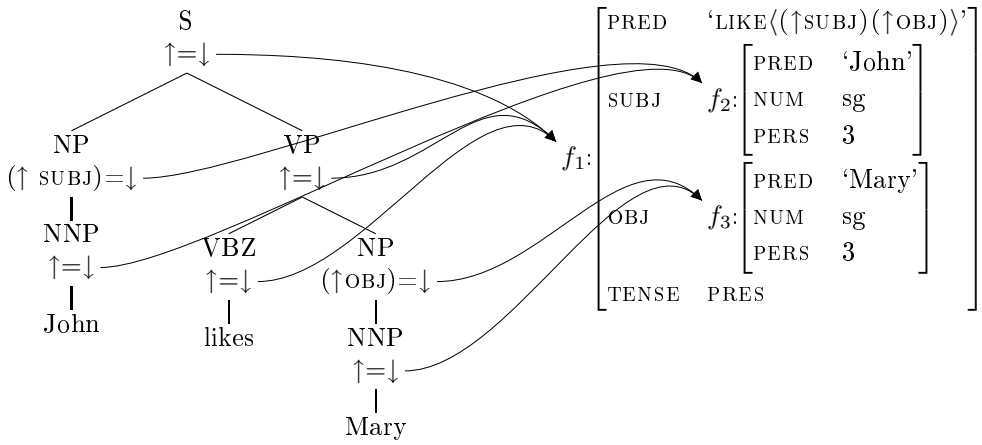

Figure 5.1: An LFG c-structure (on the left) and f-structure (on the right) for the sentence "*John likes Mary*". ϕ-links denoted by curvy arrows.

control, long-distance dependencies and some semantic information (e.g. tense, aspect). C-structures and f-structures are related in a projection architecture in terms of a piecewise correspondence ϕ. The correspondence is indicated in terms of the curvy arrows pointing from c-structure nodes to f-structure components in Figure 5.1.

Given a c-structure node n_i, the corresponding f-structure component f_j is $\phi(n_i)$. F-structures and the c-structure / f-structure correspondence are described in terms of functional annotations on c-structure nodes (CFG grammar rules). An equation of the form $(\uparrow F) = \downarrow$ states that the f-structure associated with the mother of the current c-structure node ($\uparrow$) has an attribute (grammatical function) (F), whose value is the f-structure of the current node ($\downarrow$). The up- and down-arrows are shorthand for $\phi(M(n_i)) = \phi(n_i)$ where n_i is the c-structure node annotated with the equation and M is the mother function on CFG tree nodes.

For a full description of Lexical-Functional Grammar, refer to Kaplan (1995).

5.1.1 Parsing into LFG: Cahill et al. (2004, 2008)

The automatic LFG annotation algorithm of Cahill et al. (2004, 2008) provides a means of annotating CFG trees (such as those found in the Penn WSJ treebank, or those generated by a parser such as Collins') with LFG f-structure information

based on a set of rules, heuristics and automatically-extracted subcategorisation information. This allows the rapid induction of LFG resources from treebanks. The performance of an English LFG grammar induced in this manner is comparable to or exceeding that of the hand-crafted English ParGram grammar (Cahill et al., 2008).

Cahill et al. (2004) present two approaches for using a PCFG grammar to parse into LFG[2]: a *pipeline* architecture, and an *integrated* parsing architecture. In the *pipeline* architecture, a PCFG-type grammar is extracted from a treebank and used to parse unseen text into trees, the resulting trees are automatically annotated with f-structure equations, and the corresponding f-structure produced by a constraint solver. In the *integrated* architecture, the treebank trees are first automatically annotated with f-structure information, a PCFG-type grammar is then extracted with rules containing the f-structure information, unseen text is parsed into trees with f-structure annotations, which are then passed to the constraint solver to produce the f-structure.

5.2 Sentence generation from f-structures

Sentence generation (or surface realisation) is the task of generating meaningful, grammatically correct and fluent text from some abstract semantic or syntactic representation of the sentence—in our case, LFG f-structures. It is an important and growing field of natural language processing with applications in areas such as transfer-based machine translation (Riezler and Maxwell, 2006) and sentence condensation (Riezler et al., 2003).

In our specific case the surface generation task is, given an f-structure, to generate the corresponding surface string.

[2]That is, LFG *approximations* (context-free grammars are not sufficiently expressive to fully describe constraint-based formalisms such as LFG).

5.2.1 History-based statistical chart generator: Hogan et al. (2007)

We will use a history-based statistical chart generator (Hogan et al., 2007) to perform the sentence generation task. This is an augmented version of Cahill and van Genabith's (2006) chart-based generator which is, in turn, based on the methodology of Kay (1996). The generator achieves state-of-the-art results.

The generator maximises the probability of a tree given an f-structure (Eqn. 5.1), and the surface string generated is the yield of the highest probability tree.

$$Tree_{best} := argmax_{Tree} P(Tree|FStr) \tag{5.1}$$

In the case of the Cahill and van Genabith (2006) generator, f-structure annotated CFG production rules (LHS $\rightarrow$ RHS) are conditioned on their LHSs and on the set of features/attributes $Feats = \{a_i|\exists v_j(\phi(X))a_i = v_j\}$ where attribute a_i have an associated value v_i (Eqn. 5.2).

$$P(Tree|\text{F-}Str) := \prod_{\substack{X \rightarrow Y \ in \ Tree \\ Feats = \{a_i|\exists v_j(\phi(X))a_i = v_j\}}} P(X \rightarrow Y|X, Feats) \tag{5.2}$$

The probability of a tree is decomposed into the product of the probabilities of the f-structure annotated production rules contributing to the tree. Conditional probabilities are estimated by maximum-likelihood estimation. The generator effectively turns the f-structure annotated PCFGs from the integrated parsing architecture of Cahill et al. (2004, 2008) into probabilistic generation grammars.

The Hogan et al. (2007) generator expands on this architecture, implementing a history-based model that increases the conditioning context in PCFG-style rules by including the grammatical function of the f-structure parent. This is a means of breaking down some of the independence assumptions inherent in the Cahill and van Genabith model.

Gold Sentence:

> By this time, it was 4:30 a.m. in New York, and Mr. Smith fielded a call from a **New York** customer wanting an opinion on the British **stock market**, which had been having troubles of its own even before Friday's New York market break.

Generator Output:

> By this time, in New York, it was 4:30 a.m., and Mr. Smith fielded a call from **New** a customer **York**, wanting an opinion on the **market** British **stock** which had been having troubles of its own even before Friday's New York market break.

Figure 5.2: MWU boundaries and word ordering fragmented by the generator. Gold string above, and generator output below.

5.2.2 MWUs in sentence generation

We observed that in the surface strings output by the generator, MWUs can often be fragmented. Given this, we speculate that the identification of MWUs may be useful in the generation task as a means of reducing complexity and imposing word-order constraints.

Take the example in Figure 5.2. In this case, the multi-word units "*New York*" and "*stock market*" are fragmented in the generator output. If such MWUs were treated as single units ("words-with spaces") or, alternatively, if we impose constraints on the generator such that the boundaries and word ordering of the units are strictly adhered to, this should help improve generation accuracy.

5.3 Experimental design

The automatic LFG f-structure annotation algorithm of Cahill et al. (2004, 2008) was used to produce the f-structures for development, test and training sets to be used with the generator (Hogan et al., 2007). As was the case for parsing (Chapter 4), sections 02-21 of the WSJ treebank were used to train the generator, section 24 was used as a development set and section 23 was used for final test results.

We employ the same three sources of MWU data used in our parsing experiments:

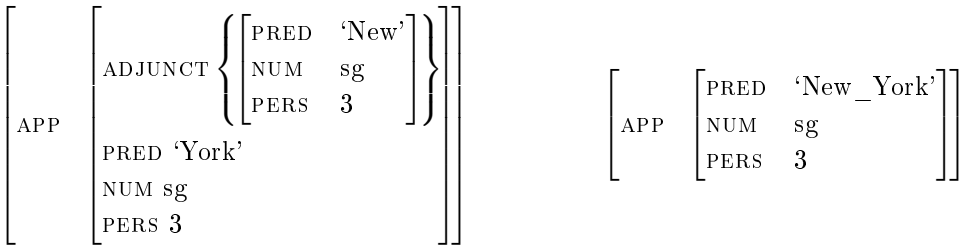

Figure 5.3: Original f-structure (on the left) for the MWU "*New York*" and the equivalent f-structure from the retokenised corpus (on the right).

the BBN corpus (Weischedel and Brunstein, 2005); Chieu and Ng's (2003) named entity recogniser; and a dictionary of MWU candidates[3]. Refer to Section 4.3 for more details on the MWU sources.

We perform two types of experiments: on the one hand, retokenising the WSJ corpus data according to word sequences that have been flagged as MWUs (cf. Section 4.4.1); and, on the other hand, retaining the original tokenisation but adding supplementary annotations that delimit the boundaries of the units—allowing us to impose constraints within the generator (cf. Section 4.4.2).

5.3.1 Retokenisation

We retokenise the Penn WSJ corpus in the same manner as previously done for our parsing experiments (Chapter 4). As was the case for parsing, we perform experiments for the case where we retokenise both the test and training data; and also for the case where we retokenise the test set but train on the original, unretokenised treebank. Figure 5.3 illustrates the effect of retokenisation, showing the original f-structure and the retokenised f-structure for the MWU "*New York*".

5.3.2 Constrained generation

In the constrained generation approach, a mechanism is introduced to the generation algorithm which penalises the generation of sequences of words which violate the internal word order of MWUs. The input is marked up in such a way that, although MWUs are no longer concatenated into single words, the generator can determine

[3] As before, based on a resource from `mwu.stanford.edu`

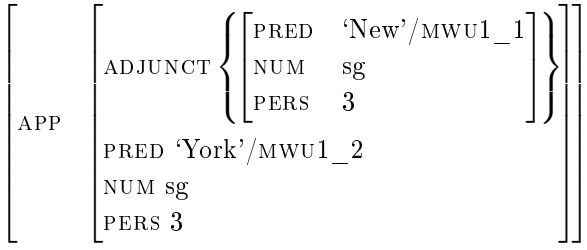

Figure 5.4: F-structure for the MWU "*New York*" using the MWU-markup method used for constrained parsing.

which items are part of the MWUs. Figure 5.4 provides an example of an f-structure marked up in this way. The tag MWU1_1, for example, indicates that the f-structure fragment is part of an MWU with id number 1 and that the item corresponds to the first word of the MWU.

5.4 Results

We evaluate the performance of the generator in terms of the BLEU metric (Papineni et al., 2001) and a simple string accuracy measure. We also give the generator's coverage over the test set (the percentage of input f-structures that generate a string).

The BLEU metric is typically used in the field of machine translation to measure translation accuracy. The metric was specifically designed to approximate human judgement with respect to the quality of a given translation versus a set of human-standard reference translations. As we use it here, BLEU can be seen as an approximate measure of sentence fluency and interpretability.

The string accuracy scores given are based on the simple string edit distance between the generator output and the gold standard sentence.

Our results include strings yielded by partial outputs produced by the generator (cases where a sequence of tree fragments are generated rather than a single, complete tree that spans the full sentence).

The baseline generator (trained on automatically-produced f-structures for sections 02-21 of the WSJ treebank) achieves a BLEU score of 66.52, a string accuracy of 68.69 and 98.18% coverage when evaluated against section 23 of the WSJ treebank.

MWU source	Test + train retokenised BLEU	Edit dist.	% cov.	Test only retokenised BLEU	Edit dist.	% cov.
baseline	0.6724	0.6989	98.18	0.6724	0.6989	98.18
NER	0.6753	0.7032	99.92	0.6775	0.7030	99.92
MWE	0.6663	0.6952	99.92	0.6700	0.6976	99.92
BBN (all)	**0.6815**	0.7052	99.96	**0.6856**	**0.7066**	99.96
NER + MWE	0.6696	0.6975	99.96	0.6743	0.7012	99.96
BBN (all) + MWE	0.6808	**0.7071**	**100**	0.6755	0.7020	**100**

Table 5.1: Results for generation with retokenised corpus (WSJ §23)

Entity type	Test + train retokenised BLEU	Edit dist.	% cov.	Test only retokenised BLEU	Edit dist.	% cov.
ENAMEX	0.6787	0.7022	**99.96**	0.6783	0.704	**99.96**
NUMEX	0.6784	0.7039	99.88	0.6733	0.6986	99.88
TIMEX	0.6784	0.7039	99.88	0.6776	0.7014	99.88
all	**0.6815**	**0.7052**	**99.96**	**0.6856**	**0.7066**	**99.96**

Table 5.2: BBN breakdown by entity type (WSJ §23)

5.4.1 Retokenisation

Table 5.1 shows our results for the retokenisation approach. The first row is our baseline generator; the second gives the results of retokenising according to the MWUs identified by Chieu and Ng's NER system; the third gives the results of retokenising according to the MWUs from the Stanford dictionary of multi-word expressions; the fourth gives the results of retokenising according to the MWUs identified by the BBN corpus; the final two rows give the results of combining the MWUs from the dictionary of multi-word expressions with those derived from the NER system and the BBN corpus, respectively. The best result for each column is shown in bold face. Table 5.2 gives a breakdown of the contribution of each of the types of entity identified in the BBN corpus.

Our best BLEU score (0.6856) was achieved with the BBN NEs when we retokenise only the test set (training the generator on the unretokenised WSJ corpus). When we combine our MWE list with the BBN NEs (retokenising both the test and training sets) we achieve the best string edit distance score (0.7071) and full coverage over the test set.

MWU source	BLEU	Edit dist.	% coverage
baseline	0.6724	0.6989	98.18
NER	0.6756	**0.701**	**99.88**
MWE	0.6740	**0.701**	99.79
BBN	0.6785	0.7010	**99.88**
NER + MWE	0.6771	0.7028	99.79
BBN + MWE	**0.6800**	**0.7027**	99.79

Table 5.3: Results for generation with internal constraints (WSJ §23)

5.4.2 Constrained generation

Table 5.3 shows the results achieved when we use the MWU-markup and constrained generation approach. The first row is our baseline generator (the same as that from the previous experiments); the second is when we impose constraints based on the NER system; the third based on the MWE dictionary; the fourth based on the BBN corpus NEs; and the final two rows are when we combine the MWE dictionary with the NER system and the BBN corpus, respectively.

The best result for each column is shown in bold face.

The constrained generation approach did not perform as well as the retokenisation approach. We achieve the best results with the BBN corpus NEs combined with MWE dictionary lookup: BLEU score of 0.68, string edit distance of 0.7027 and 99.79% coverage.

5.4.3 Summary

Overall, the best **coverage** (100%) was achieved by retokenising both the test and training data according to the BBN corpus-derived NEs combined with the dictionary of MWE candidates, this also yielded the best overall **string edit distance** score (0.7071).

The best overall **BLEU score** (0.6856) was achieved by retokenising the test data according to the BBN corpus-derived NEs (with the generator trained on the original, unretokenised training set). Near-full coverage (99.96%) was achieved in this case.

These improvements were found to be statistically significant using a bootstrap resampling test with 10,000 shuffles to level $p = 0.006$ and $p = 0.00009$, respectively.

5.5 Discussion and implications

In our evaluation of incorporating MWUs in surface generation from LFG f-structures we have demonstrated that moderate improvements in generator accuracy can be achieved. For automatically acquired MWUs, we found that this could best be achieved by concatenating input items when producing the f-structure input to the generator, while training the input generation grammar on the original (i.e. non MWU-concatenated) sections of the treebank.

While our investigations here have shown that there exists potential to exploit MWUs as a means of informing both word ordering and selection in surface realisation from f-structures, there remains much room for additional experiments and analysis.

Referring back to Figure 5.2, the compound-noun MWU "stock exchange" is not identified by our approaches (because such non-NE compound nominals are not present in our training data). It is therefore clear that there is scope for improvements by the identification of such compound nouns (and other, additional MWU types).

While the constrained generation approach performed reasonably well, it is surprising that it yielded scores lower than those achieved with the retokenisation approach (where we retokenise the test data alone, leaving the training set unaltered). Although we did not conduct thorough investigation, we believe that the constraints mechanism might be interacting poorly with the generation model (or vice versa). If suitable improvements were made, we expect that the performance could equal (in fact, might exceed) that of the retokenisation approach. This is a potential avenue of further research.

Chapter 6

Comparison with Related Work

In this chapter we will compare our research to similar work that we believe to be relevant.[1] As we have previously noted, there is a surprisingly small amount of work published on the subject of exploiting knowledge about multi-word units (MWUs) in syntactic parsing. Moreover, the majority of such work falls in the realm of hand-crafted, rule-based approaches to parsing rather than the data-driven, statistical methods with which we are concerned. Although parallels can be drawn between such approaches and our own research, we believe that our work fills a conspicuous gap in the literature.

6.1 Introduction

Multi-word units have traditionally been integrated into hand-crafted grammars in the form of "words with spaces", primarily as a means of improving coverage and robustness. While there have been a number of investigations into the utility of leveraging MWUs in the context of these hand-crafted, rule-based approaches to parsing (which we will discuss below), there has been very little work carried out which evaluates the utility of identifying MWUs for statistical parsing.

[1] Since our research does not specifically deal with MWU *identification* (for the most part we exploit existing methods and resources) we will not discuss related work on that issue here (see Chapter 3 for an overview of the subject).

6.2 MWUs in shallow parsing

Our experiments have fallen under the category of shallow (or "skeletal") syntactic parsing, as opposed to "deep" parsing based on linguistic formalisms such as LFG or HPSG (see Section 6.3 below). We are not aware of any research that deals with evaluating the effect of leveraging MWUs in our specific context (data-driven syntactic constituency parsing). Indeed, there has been only a very limited amount of work carried out with respect to MWUs in *statistical* approaches to parsing in general. The most notable of such research, and most analogous to our own work, is that of Nivre and Nilsson (2004) who exploit multi-word units in the loosely related task of statistical dependency parsing.

6.2.1 Dependency parsing: Nivre and Nilsson (2004)

Dependency parsing is the task of deriving the tree which represents grammatical relations—dependencies—between a given sequence of word tokens. In dependency grammars, no nonterminal nodes are present between a head and its dependent; that is, dependencies are defined in terms of direct grammatical relationships between the the nodes of the sequence. Dependency parsing is often said to combine many of the attributes of shallow (constituency) parsing with those of deeper approaches to parsing grounded in linguistic formalisms such as HPSG or LFG. (Section 2.8.2 provides a more detailed look at the subject of dependency parsing.)

Nivre and Nilsson (2004) investigate the influence of identifying certain classes of MWUs with respect to dependency parsing of Swedish using a parser employing memory-based learning. Their approach to integrating MWUs in the parsing process is roughly equivalent to our retokenisation method.

The types of MWUs with which their investigations are concerned are those that have been flagged in the Talbanken05 (Swedish treebank) corpus (Nivre et al., 2006). These are quite diverse: they include named entity-type MWUs such as multi-word names and number expressions; as well as various types of compound function words including adverbs, prepositions, subordinating conjunctions, determiners and pro-

	Non-lexicalised parser		Lexicalised parser	
	AS	LAS	AS	LAS
baseline	83.0	76.1	84.7	80.7
MWU	83.5	77.4	85.6	81.6

Table 6.1: Summary of Nivre and Nilsson's (2004) results: AS = attachment score; LAS = labelled attachment score.

nouns.

Nivre and Nilsson (2004) evaluate in terms of attachment score: the proportion of tokens (excluding punctuation) that are assigned the correct head. They report results for both the unlabelled and labelled case of the attachment score (AS and LAS, respectively). Table 6.1 summaries their experimental results (consult the paper for further results and discussion). They achieve gains that they determine to be quite positive, corresponding to up to 5% error reduction (which, although seemingly modest, is significant when the relative scarcity of the MWUs is taken into account).

6.3 MWUs in deep parsing

In "deep" parsing we generate phrase-structure trees similar to those that would be output by a skeletal parser but with a much richer set of grammatical relations between the nodes of the tree. These relations might include predicate-argument dependencies as well as other types of functional and semantic relations. Examples of deep linguistic formalisms include HPSG (Pollard and Sag, 1994), LFG, (Dalrymple, 2006), TAG (Joshi, 1987) and CCG (Steedman, 2000).

In contrast to the statistical approaches to parsing with which we are concerned in this thesis, deep parsing has traditionally employed rule-based approaches based on hand-crafted grammars (a notable exception being the StatCCG parser for Combinatory Categorial Grammar).[2] In research carried out relating to incorporating MWUs in these deep parsing architectures the primary goal has typically been to provide a semi-automated way of increasing the lexicon size and thus increasing a grammar's coverage and robustness. We will summarise some of this work below.

[2]Many rule-based approaches now incorporate statistical disambiguators, however.

	% coverage	# solutions	Best f-score
baseline	76	482	82
with NEs	78	263	86

Table 6.2: Summary of Kaplan et al.'s (2003) results

6.3.1 LFG: Kaplan et al. (2003)

Lexical Functional Grammar (see Section 5.1 for an overview) is a constraint-based theory of grammar which analyses strings in terms of constituency structures and functional structures. Constituency structures are defined in terms of context-free grammars while functional structures are recursive attribute-value matrices that represent abstract syntactic functions, agreement, control, long-distance dependencies and some semantic information.

Kaplan and King (2003) perform an experiment using the English ParGram grammar (Butt and King, 1999) incorporating named entities—specifically, proper names—based on gold-standard data as parsing constraints (this is analogous to our experiments using the ENAMEX class of NEs from the BBN corpus in Chapter 4).

Evaluating against the PARC-700 Dependency Bank (King et al., 2003), a subset of the Penn WSJ Treebank that has been annotated with f-structures, they report some substantial gains across the board: increased coverage, reduced ambiguity and improved accuracy (Table 6.2).

Although they allude to additional experiments that they have performed using automatically-acquired NEs, they present results for gold standard NE data only.

6.3.2 HPSG: Villavicencio et al. (2007)

A good deal of the literature on identifying multi-word units has come from efforts associated with Head-driven Phrase Structure Grammar (HPSG). The Multiword Expression Project[3] led by Stanford University is a good example of this (we make use of MWU resources developed under the auspices of this project in our experiments in Chapters 4 and 5). Despite this, there are very few papers that deal specifically

[3] `http://mwu.stanford.edu/`

	# items	# parsed	# avg. analyses	% coverage
ERG	674	48	335.08	7.1
ERG with MWE	674	153	285.01	22.7

Table 6.3: Summary of Villavicencio et al.'s (2007) results

with methodologies for incorporating such MWUs in a HPSG parsing architecture.

One pertinent example found in the literature however is that of Villavicencio et al. (2007) who perform experiments in which they exploit automatically identified MWUs with the English Resource Grammar (ERG—Copestake and Flickinger, 2000). They report an impressive increase in coverage on a subcorpus of the BNC from 7% to 21% (see Table 6.3).

6.3.3 CCG, TAG and others

Sag et al. (2004) report that there are ongoing attempts to integrate MWUs to varying degrees in grammars based on several other deep linguistic formalisms including Combinatory Categorial Grammar (CCG) and Tree-Adjoining Grammar (TAG).

6.4 Related work on constrained parsing

Our approach to constrained parsing is based on that of Glaysher and Moldovan (2006) who use a modified version of the Collins parser. In their experiments, they use a HMM-based syntactic chunker (partial parser) to identify chunks which are used to enforce restrictions on the spans permitted to be added to the parse chart such that parse derivations are consistent with these syntactic chunks (this is exactly how we treat our MWUs in our constrained parsing experiments in Chapter 4). They report an almost threefold increase in the efficiency (i.e. speed) of the parser while incurring a minimal loss in accuracy.

6.4.1 Kulick et al. (2006)

Kulick et al. (2006) report on research where they exploit partial prebracketing and constrained parsing with a view to expedite treebank construction. They perform

experiments with the Penn Treebank and the Penn-Helsinki Parsed Corpus of Early Modern English (Kroch and Taylor, 1999) and, like us, they make use of the constraints framework built in to Bikel's (2002) parser. They encounter some issues where the constraint mechanism interacts poorly with some of the complex and interacting parameters of Collins' (1999) parsing model—these issues do not apply to our approach however, since we use a much more straightforward technique in which we are not concerned with the syntactic categories of our constraint spans. That is to say, in their approach the parse chart is seeded with specific labelled spans prior to parsing (they also implement "negative constraints"). In our approach we do not rigidly impose such spans but rather impose restrictions on which spans are permitted to be added to the chart by stipulating that a span may not *overlap* with our constraint spans (it may however, again unlike the Kulick et al. approach, subsume the constraint span).

Chapter 7

Conclusions

Our core hypothesis in this thesis has been that we can exploit knowledge about certain types of multi-word units (MWUs) as a means to improve statistical parsing. We speculated that this should improve both parsing accuracy and efficiency. More broadly, we believe that syntactic parsing can be improved by incorporating other types of extra-treebank linguistic information.

We have described several approaches to parsing natural language and placed our work within this context, focusing on statistical approaches to "shallow" syntactic parsing. We have defined the concept of multi-word units—sequences of word tokens that are (to at least some degree) syntactically non-compositional—and have described some of the phenomena that can be classified as such: including various types of fixed and semi-fixed expressions, and named entities (NEs).

In the core of our research we have described a number of experiments that put our intuitions to the test. Specifically we assessed the impact of exploiting MWU data in three different parsing architectures, comparing two general approaches: corpus retokenisation and constrained parsing. In the former, we alter the corpus on which the parser is trained and evaluated (the Penn WSJ treebank) such that MWUs are treated as single word tokens; while in the latter we retain the original tokenisation of the corpus and instead impose partial phrase-boundary constraints during parsing. We have also reported on additional experiments where we make

use of the same MWU information in the converse operation to statistical parsing—statistical sentence generation. Finally we have compared our research with similar and related work on multi-word units in syntactic parsing and on constrained parsing in general. To the best of our knowledge, the research reported here is the first systematic investigation of the effect of MWU data in treebank-based, wide-coverage CFG parsing and generation.

Our experimental results have shown that incorporating MWUs in statistical parsing can indeed be beneficial in some cases, and we have discussed concrete examples where MWU data addresses specific classes of syntactic ambiguity and improves overall parse quality. However, the gains that we have reported have been modest at best (we achieve a reduction of 1.8% in parser error for gold-standard data). Our supplementary experiments with sentence generation exploiting MWUs yielded gains of a similar scale to those observed in our parsing experiments. This is somewhat disappointing; but, at the same time, is testimony to the value of statistical models in natural languages processing: it turns out that the statistical parsing and generation architectures that we have employed in our experiments already do a quite good job of accounting for the phenomena associated with multi-word units. It could be argued that the very fact that the specific types of MWUs that we employed are quite readily-identifiable with a high degree of accuracy means that they are usually unlikely to cause major problems for a statistical parser.

The small scale of the improvements that we observed in our experimental results are in stark contrast to the case of hand-crafted, rule-based approaches to parsing—where the effect of incorporating MWU data (e.g. NEs) has been to yield substantial improvements in parsing performance (Kaplan and King, 2003).

State-of-the-art results for parsing have reached the region of 90%+ f-measure. We believe that if further improvements in parsing are to be brought about that generalise across corpora and text domains then it is a necessity to incorporate other (non-treebank) linguistic information to statistical parsing models. The methodologies described for incorporating MWUs (in particular constrained parsing) are general

and could be applied to other types of linguistic phenomena (e.g. syntactic chunks as per Glaysher and Moldovan, 2006). It is our contention that machine-learning based classifiers—such Chieu and Ng's (2003) maximum entropy-based named entity recogniser—could be a useful way of incorporating other forms of linguistic information into a parsing pipeline, either as a pre- or postprocessing module.

Even though the gains in parsing performance reported in this thesis are modest, our work fills a conspicuous gap in the research literature as we are not aware of similar work published that deals specifically with multi-word units in statistical approaches to syntactic (constituency) parsing and generation.

7.1 Future work

In our research we have dealt with different types of named entities (name expressions, time expressions and number expressions) and with certain types of lexically rigid prepositional multi-word expressions but, as we described in Chapter 3, there exist many other classes of multi-word units. From this perspective, our research has merely scratched the surface of the possibilities for integrating knowledge of multi-word units in statistical models of syntactic parsing. Another potential source of MWU data, for example, is WordNet (Fellbaum et al., 1998) (where 41% of lexical entities in version 1.7 are multi-word). In addition, we might try to account for more complex phenomena such verb-particle constructions and light verbs. Other approaches to integrating the data in the parsing architecture could also be explored (e.g. using MWU data as feature variables in a reranker).

Bibliography

S. Abney, S. Flickenger, C. Gdaniec, C. Grishman, P. Harrison, D. Hindle, R. Ingria, F. Jelinek, J. Klavans, M. Liberman, et al. Procedure for quantitatively comparing the syntactic coverage of English grammars. *Proceedings of the workshop on Speech and Natural Language*, pages 306–311, 1991.

Eneko Agirre, Timothy Baldwin, and David Martinez. Improving parsing and PP attachment performance with sense information. In *Proceedings of ACL-08: HLT*, pages 317–325, Columbus, Ohio, June 2008. Association for Computational Linguistics.

Alfred V. Aho and Jeffrey D. Ullman. *The Theory of Parsing, Translation, and Compiling, Vol 1: Parsing, Vol 2: Compiling.* Prentice Hall, 1972.

D.M. Bikel. Design of a Multi-Lingual, Parallel-Processing Statistical Parsing Engine. *Proceedings of the Second International Conference on Human Language Technology Research*, pages 178–182, 2002.

D.M. Bikel. *On the Parameter Space of Generative Lexcicalized Statistical Parsing Models.* PhD thesis, University of Pennsylvania, 2004.

E. Black, F. Jelinek, J. Lafferty, DM Magerman, R. Mercer, and S. Roukos. Towards History-based Grammars: Using Richer Models of Context in Probabilistic Parsing. *Proceedings of the February 1992 DARPA Speech and Natural Language Workshop. Arden House, NY*, 1992.

J. Bresnan. *Lexical-Functional Syntax.* Blackwell, Oxford, 2001.

S. Buchholz and E. Marsi. CoNLL-X shared task on multilingual dependency parsing. *Proc. of the Tenth Conference on Computational Natural Language Learning*, pages 189–210, 2006.

L. Burnard. Users Reference Guide for the British National Corpus. Technical report, Oxford University Computing Services, 2000.

M. Butt and T. King. *A Grammar Writer's Cookbook*. MIT Press, 1999.

C. Cafferkey, D. Hogan, , and J. van Genabith. Multi-Word Units in Treebank-Based Probabilistic Parsing and Generation. *Proceedings of the 10th International Conference on Recent Advance in Natural Language Processing (RANLP-2007), Borovets, Bulgaria*, 2007.

A. Cahill. *Parsing with Automatically Acquired, Wide-Coverage, Robust, Probabilistic LFG Approximations*. PhD thesis, Dublin City University, 2004.

A. Cahill and J. van Genabith. Robust PCFG-Based Generation Using Automatically Acquired LFG Approximations. *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL*, pages 1033–1040, 2006.

A. Cahill, M. Burke, R. O'Donovan, S. Riezler, J. van Genabith, and A. Way. Wide-Coverage Deep Statistical Parsing Using Automatic Dependency Structure Annotation. *Computational Linguistics*, 34(1):81–124, 2008.

J. Carroll, A. Frank, D. Lin, D. Prescher, and H. Uszkoreit. Beyond PARSEVAL-Towards Improved Evaluation Measures for Parsing Systems. *Workshop at the 3rd International Conference on Language Resources and Evaluation LREC-02., Las Palmas*, 2002.

E. Charniak. Statistical Parsing With a Context-Free Grammar and Word Statistics. *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, pages 598–603, 1997.

E. Charniak. A Maximum-Entropy-Inspired Parser. *ACM International Conference Proceeding Series*, 4:132–139, 2000.

E. Charniak and M. Johnson. Coarse-tofine n-best parsing and maxent discriminative reranking, 2005.

E. Charniak, K. Knight, and K. Yamada. Syntax-Based Language Models for Statistical Machine Translation. *MT Summit IX*, 2003.

S.F. Chen and J. Goodman. An Empirical Study of Smoothing Techniques for Language Modeling. *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pages 310–318, 1996.

H.L. Chieu and H.T. Ng. Named Entity Recognition with a Maximum Entropy Approach. 2003.

M. Collins. *Head-Driven Statistical Models for Natural Language Parsing.* PhD thesis, University of Pennsylvania, 1999.

A. Copestake and D. Flickinger. An Open-Source Grammar Development Environment and Broad-Coverage English Grammar Using HPSG. *Proceedings of LREC*, 2, 2000.

M. Dalrymple. Lexical Functional Grammar. 2006.

C. Fellbaum et al. *WordNet: An Electronic Lexical Database.* Cambridge, Mass: MIT Press, 1998.

R. Florian, A. Ittycheriah, H. Jing, and T. Zhang. Named Entity Recognition through Classifier Combination. In *Proceedings of CoNLL-2003*, volume 58, 2003.

J. Foster, J. Wagner, D. Seddah, and J. van Genabith. Adapting WSJ-Trained Parsers to the British National Corpus Using In-Domain Self-Training. *IWPT 2007.*

D. Gildea. Corpus variation and parser performance. *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing*, pages 167–202, 2001.

E. Glaysher and D. Moldovan. Speeding up Full Syntactic Parsing by Leveraging Partial Parsing Decisions. *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 295–300, 2006.

D. Hindle and M. Rooth. Structural Ambiguity and Lexical Relations. *Computational Linguistics*, 19(1):103–120, 1993.

J. Hockenmaier. *Data and Models for Statistical Parsing with CCG*. PhD thesis, Ph. D. thesis, School of Informatics, University of Edinburgh, 2003.

D. Hogan. Coordinate Noun Phrase Disambiguation in a Generative Parsing Model. *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 680–687, 2007.

D. Hogan, C. Cafferkey, A. Cahill, and J. van Genabith. Exploiting Multi-Word Units in History-Based Probabilistic Generation. *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 267–276, 2007.

M. Johnson. PCFG Models of Linguistic Tree Representations. *Computational Linguistics*, 24(4):613–632, 1998.

A.K. Joshi. An Introduction to Tree Adjoining Grammars. *Mathematics of Language*, 1:87–115, 1987.

R.M. Kaplan. The Formal Architecture of Lexical-Functional Grammar. *Formal Issues in Lexical-Functional Grammar*, pages 7–27, 1995.

R.M. Kaplan and J. Bresnan. Lexical Functional Grammar. *Bresnan, J., editor*, pages 173–281, 1982.

R.M. Kaplan and T.H. King. Low-Level Mark-Up and Large-Scale LFG Grammar Processing. *Proceedings of the LFG 2003 Conference, Saratoga Springs, New York*, 2003.

M. Kay. Chart Generation. *Proceedings of the 34th Conference of the Association for Computational Linguistics*, pages 200–204, 1996.

T.H. King, R. Crouch, S. Riezler, M. Dalrymple, and R. Kaplan. The PARC 700 Dependency Bank. *Proceedings of the Fourth International Workshop on Linguistically Interpreted Corpora (LINC-03)*, pages 1–8, 2003.

D. Klein and C.D. Manning. Accurate Unlexicalized Parsing. *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 423–430, 2003.

A. Kroch and A. Taylor. The Penn-Helsinki Parsed Corpus of Middle English. *University of Pennsylvania*, 1999.

S. Kulick, D. Bikel, and A. Kroch. Treebank Construction by Levels Using Constrained Chart Parsing. *Proceedings of the Fifth International Conference on Treebanks and Linguistic Theories, Prague, Czech Republic*, 2006.

M. Lease, E. Charniak, M. Johnson, and D. McClosky. A Look At Parsing and Its Applications. *Proceedings of the Twenty-First National Conference on Artificial Intelligence (AAAI-06)*, pages 16–20, 2006.

D.M. Magerman. Statistical Decision-Tree Models for Parsing. *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, 276283, 1995.

M. Marcus, G. Kim, M. Marcinkiewicz, R. MacIntyre, A. Bies, M. Ferguson, K. Katz, and B. Schasberger. The Penn treebank: Annotating predicate argument structure, 1994.

M.P. Marcus, M.A. Marcinkiewicz, and B. Santorini. Building a Large Annotated Corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313–330, 1993.

T. Matsuzaki and Y. Miyao. Probabilistic CFG with Latent Annotations. *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 75–82, 2005.

D. McClosky, E. Charniak, and M. Johnson. Effective self-training for parsing. *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 152–159, 2006.

R. McDonald, K. Lerman, and F. Pereira. Multilingual Dependency Analysis With a Two-Stage Discriminative Parser. *Proceedings of CoNLL*, 2006.

A. Mikheev, C. Grover, and M. Moens. Description of the LTG system used for MUC. In *Seventh Message Understanding Conference: Proceedings of a Conference*, 1998.

Y. Miyao, T. Ninomiya, and J. Tsujii. Corpusoriented grammar development for acquiring a Headdriven Phrase Structure Grammar from the Penn Treebank. *Proc. IJCNLP*, 4, 2004.

J. Nivre. Dependency grammar and dependency parsing. *MSI report*, 4071, 2005.

J. Nivre and J. Nilsson. Multiword Units in Syntactic Parsing. *MEMURA 2004 - Methodologies and Evaluation of Multiword Units in Real-World Applications, Workshop at LREC 2004*, pages 39–46, 2004.

J. Nivre, J. Nilsson, and J. Hall. Talbanken05: A Swedish Treebank with Phrase Structure and Dependency Annotation. *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC2006)*, 2006.

J. Nivre, J. Hall, S. Kubler, R. McDonald, J. Nilsson, S. Riedel, and D. Yuret. The CoNLL 2007 Shared Task on Dependency Parsing. *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL*, pages 915–932, 2007a.

J. Nivre, J. Hall, J. Nilsson, A. Chanev, G. Eryigit, S. Kübler, S. Marinov, and

E. Marsi. MaltParser: A Language-Independent System for Data-Driven Dependency Parsing. *Natural Language Engineering*, 13(02):95–135, 2007b.

K. Papineni, S. Roukos, T. Ward, and W.J. Zhu. BLEU: A Method for Automatic Evaluation of Machine Translation. *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, 2001.

S. Petrov, L. Barrett, R. Thibaux, and D. Klein. Learning Accurate, Compact, and Interpretable Tree Annotation. *ACL 44/COLING*, 21:433–440, 2006.

S.S.L. Piao, P. Rayson, D. Archer, A. Wilson, and T. McEnery. Extracting Multiword Expressions With a Semantic Tagger. In *Proceedings of the ACL 2003 workshop on Multiword expressions: analysis, acquisition and treatment-Volume 18*, pages 49–56. Association for Computational Linguistics Morristown, NJ, USA, 2003.

C.J. Pollard and I.A. Sag. *Head-Driven Phrase Structure Grammar*. University Of Chicago Press, 1994.

S. Riezler and J.T. Maxwell. Grammatical Machine Translation. *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 248–255, 2006.

S. Riezler, T.H. King, R. Crouch, and A. Zaenen. Statistical Sentence Condensation using Ambiguity Packing and Stochastic Disambiguation Methods for Lexical-Functional Grammar. *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 118–125, 2003.

I.A. Sag, T. Baldwin, F. Bond, A. Copestake, and D. Flickinger. Multiword Expressions: A Pain in the Neck for NLP. *Proc. of the 3rd International Conference on Intelligent Text Processing and Computational Linguistics (CICLing-2002)*, pages 1–15, 2002.

Kenji Sagae and Jun'ichi Tsujii. Dependency parsing and domain adaptation with LR

models and parser ensembles. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 1044–1050, 2007.

G. Sampson and A. Babarczy. A test of the leaf-ancestor metric for parse accuracy. *Natural Language Engineering*, 9(04):365–380, 2003.

H. Schmid. Efficient Parsing of Highly Ambiguous Context-Free Grammars with Bit Vectors. *Proceedings of the 20th International Conference on Computational Linguistics*, 2004.

S. Sekine. The Domain Dependence of Parsing. *Fifth Conference on Applied Natural Language Processing*, pages 96–102, 1998.

S. Sekine, K. Sudo, and C. Nobata. Extended Named Entity Hierarchy. *Proceedings of the LREC-2002 Conference*, pages 1818–1824, 2002.

M. Steedman. *The Syntactic Process*. MIT Press, 2000.

M. Steedman, M. Osborne, A. Sarkar, S. Clark, R. Hwa, J. Hockenmaier, P. Ruhlen, S. Baker, and J. Crim. Bootstrapping statistical parsers from small datasets. *The Proceedings of the Annual Meeting of the European Chapter of the ACL*, pages 331–338, 2003.

M. Surdeanu, S. Harabagiu, J. Williams, and P. Aarseth. Using Predicate-Argument Structures for Information Extraction. *Proceedings of ACL 2003*, pages 8–15, 2003.

J. Turner and E. Charniak. Supervised and Unsupervised Learning for Sentence Compression. *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 290–297, 2005.

A. Villavicencio, V. Kordoni, Y. Zhang, M. Idiart, and C. Ramisch. Validation and Evaluation of Automatically Acquired Multiword Expressions for Grammar Engineering. *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 1034–1043, 2007.

R. Weischedel and A. Brunstein. BBN Pronoun Coreference and Entity Type Corpus. *Linguistic Data Consortium*, 2005.

Appendix A

Penn Treebank POS Tags

Tag	Description	Tag	Description
CC	Co-ordinating conjunction	PRP$	Possessive pronoun
CD	Cardinal number	RB	Adverb
DT	Determiner	RBR	Comparative adverb
EX	Existential "there"	RBS	Superlative adverb
FW	Foreign word	RP	Particle
IN	Preposition	SYM	Symbol
JJ	Adjective	TO	"to"
JJR	Comparative adjective	UH	Interjection
JJS	Superlative adjective	VB	Base-form verb
LS	List item	VBD	Past-tense verb
MD	Modal	VBG	Gerund verb
NN	Singular noun	VBN	Past-participle verb
NNS	Plural noun	VBP	Non-3sg present-tense verb
NNP	Singular proper noun	VBZ	3sg present-tense verb
NNPS	Plural proper noun	WDT	*Wh*-determiner
PDT	Predeterminer	WP	*Wh*-pronoun
POS	Possessive ending	WP$	Possessive *wh*-
PRP	Personal pronoun	WRB	*Wh*-adverb

Tag	Description	Tag	Description
$	Dollar sign	)	Right parenthesis
#	Pound sign	,	Comma
“	Left quote	.	Sentence-final punctuation
”	Right quote	:	Mid-sentence punctuation
(	Left parenthesis		

Appendix B

Penn Treebank Syntactic Labels

Tag	Description
ADJP	Adjective phrase
ADVP	Adverb phrase
CONJP	Conjunction phrase
FRAG	Fragment
INTJ	Interjection
NAC	Not a constituent
NP	Noun phrase
NX	Head sub-phrase of complex noun phrase
PP	Prepositional phrase
QP	Quantifier phrase
RRC	Reduced relative clause
S	Simple declarative clause (sentence)
SBAR	Clause introduced by complementiser
SBARQ	Question introduced by *wh*-word
SINV	Inverted declarative sentence
SQ	Inverted yes-no question
UCP	Unlike co-ordinated phrase
VP	Verb phrase

Tag	Description
WHADJP	*Wh*-adjective phrase
WHADVP	*Wh*-adverb phrase
WHNP	*Wh*-noun phrase
WHPP	*Wh*-prepositional phrase
X	Constituent of unknown or uncertain category

Appendix C

List of Most Common BBN Corpus Entities

Here we give the most common named entities identified by the BBN corpus for sections 02-21 of the WSJ Treebank (39,832 sentences, 1,014,129 word tokens)—as used in our parsing and generation experiments (described in Chapters 4 and 5, respectively). We list the 100 most common name expressions (denoted ENAMEX), number expressions (NUMEX) and time expressions (TIMEX).

C.1 100 most common name expressions (ENAMEX)

	count
1. new york	501
2. vice president	337
3. new york stock exchange	224
4. san francisco	203
5. chief executive officer	174
6. big board	150

7.	white house	128
8.	hong kong	118
9.	los angeles	117
10.	soviet union	80
11.	securities and exchange commission	78
12.	west germany	72
13.	west german	71
14.	moody 's[1]	67
15.	hurricane hugo	64
16.	dow jones	64
17.	chief operating officer	60
18.	supreme court	59
19.	chapter 11[2]	54
20.	navigation mixte	54
21.	merrill lynch	53
22.	sci tv	52
23.	prime minister	51
24.	bay area	49
25.	lloyd 's	47
26.	federal reserve	47

[1] the possessive case marker ('*s*) is treated as a distinct token in the Penn Treebank
[2] as in bankruptcy

27.	east germany	47
28.	justice department	47
29.	fannie mae	46
30.	new york city	46
31.	sotheby 's	45
32.	vice chairman	43
33.	the wall street journal	43
34.	chief financial officer	40
35.	british air	40
36.	qintex australia	39
37.	new jersey	39
38.	south africa	38
39.	freddie mac	34
40.	world series	32
41.	south korea	32
42.	american stock exchange	32
43.	philip morris	31
44.	international business machines corp.	31
45.	general electric co.	31
46.	control data	31
47.	american express	30

48.	merrill lynch capital markets	30
49.	new york-based	30
50.	air force	29
51.	attorney general	29
52.	shearson lehman hutton	29
53.	united airlines	29
54.	general motors corp.	28
55.	commerce department	28
56.	first boston	28
57.	european community	27
58.	time warner	27
59.	brooks brothers	26
60.	bloomingdale 's	25
61.	las vegas	25
62.	morgan stanley	25
63.	george bush	25
64.	ual corp.	25
65.	pinkerton 's	25
66.	bay bridge	24
67.	qintex entertainment	24
68.	state department	24

69.	ford motor co.	24
70.	north america	24
71.	drexel burnham lambert inc.	24
72.	first boston corp.	23
73.	american airlines	23
74.	eastern europe	23
75.	du pont	23
76.	costa rica	23
77.	bear stearns	23
78.	salomon brothers inc.	22
79.	east bloc	22
80.	gulf power	22
81.	social security	22
82.	bond corp.	22
83.	east german	22
84.	san jose	21
85.	new zealand	21
86.	british airways	21
87.	moody 's investors service inc.	21
88.	goldman , sachs & co.	21
89.	morgan stanley & co.	20

90.	general motors	20
91.	telerate systems inc	20
92.	red cross	20
93.	standard & poor 's corp.	20
94.	federal reserve board	19
95.	merrill lynch & co.	19
96.	federal national mortgage association	19
97.	k mart	19
98.	shearson lehman hutton inc.	19
99.	mcdonald 's	19
100.	communist party	19

C.2 100 most common number expressions (NUMEX)

		count
1.	10 %[3]	91
2.	15 %	89
3.	50 %	79
4.	8 %	67
5.	9 %	64
6.	20 %	64
7.	5 %	63

[3] the percentage and dollar symbols are treated as a distinct tokens in the Penn Treebank

8.	7 %	61
9.	$ 1 billion	58
10.	12 %	56
11.	$ 1 million	54
12.	25 %	52
13.	$ 1,000	52
14.	$ 200 million	51
15.	30 %	49
16.	$ 100 million	46
17.	40 %	46
18.	$ 150 million	45
19.	2 %	43
20.	6 %	41
21.	$ 50 million	40
22.	$ 500 million	40
23.	11 %	39
24.	4 %	37
25.	16 %	35
26.	12.5 cents	34
27.	25 cents	34
28.	$ 2 billion	33

29. $ 10 million	33
30. 22 %	33
31. 3 %	33
32. 8.50 %	32
33. 14 %	32
34. 1 1/4	31
35. 13 %	31
36. 8 3/4 %	30
37. 0.2 %	30
38. 18 %	30
39. 17 %	30
40. only one	30
41. 60 %	29
42. 50 cents	29
43. 51 %	28
44. $ 15 million	28
45. about half	28
46. $ 1	28
47. at least three	28
48. 8 11/16 %	28
49. 1 3/8	27

50. $ 10,000	27
51. 35 %	26
52. $ 20 million	25
53. $ 3 billion	25
54. $ 5,000	24
55. $ 400 million	24
56. $ 40 million	23
57. 80 %	23
58. 33 %	23
59. $ 350 million	23
60. 1 1/2	22
61. $ 100,000	22
62. more than half	22
63. 19 %	21
64. 10 cents	20
65. 1 1/8	20
66. 75 cents	20
67. $ 4 billion	20
68. $ 30 million	20
69. 23 %	20
70. five cents	20

71. 2.5 %	20
72. 100 %	19
73. 4.6 %	19
74. $ 300 million	19
75. $ 8 million	19
76. $ 4 million	19
77. $ 1.1 billion	19
78. $ 500,000	18
79. 37.5 cents	18
80. 5.5 %	18
81. 0.3 %	18
82. $ 1.3 billion	18
83. 1 %	18
84. 4.5 %	18
85. 8.45 %	17
86. $ 300-a-share	17
87. 1 7/8	17
88. $ 1.5 billion	17
89. 70 %	17
90. 1 3/4	17
91. 44 %	17

92.	28 %	17
93.	24 %	16
94.	$ 25 million	16
95.	$ 250 million	16
96.	49 %	16
97.	21 %	16
98.	45 %	16
99.	55 %	15
100.	$ 750 million	15

C.3 100 most common time expressions (TIMEX)

		count
1.	last year	311
2.	a year earlier	294
3.	this year	284
4.	last week	246
5.	the third quarter	158
6.	a year ago	145
7.	last month	126
8.	next year	125
9.	this week	123
10.	the quarter	117

11.	earlier this year	99
12.	the day	86
13.	the nine months	85
14.	the year	71
15.	recent years	61
16.	this month	56
17.	six months	54
18.	three months	50
19.	earlier this month	50
20.	oct. 13	49
21.	the fourth quarter	49
22.	the second quarter	48
23.	next month	46
24.	the latest quarter	44
25.	one month	44
26.	last friday	43
27.	30 days	43
28.	five years	43
29.	a year	38
30.	sept. 30	38
31.	two years ago	37

32.	three years	36
33.	two years	35
34.	these days	35
35.	the weekend	34
36.	one year	34
37.	next week	31
38.	early next year	30
39.	this summer	29
40.	fiscal 1990	29
41.	the week	29
42.	two months	29
43.	late yesterday	27
44.	year end	27
45.	60 days	26
46.	the past year	26
47.	recent weeks	26
48.	90 days	25
49.	fiscal 1989	24
50.	oct. 31	23
51.	last spring	23
52.	three years ago	22

53.	a day	21
54.	four years	21
55.	recent months	21
56.	each year	21
57.	the end of the year	20
58.	the previous year	20
59.	the past 30 days	20
60.	the year-earlier quarter	19
61.	the month	19
62.	two days	19
63.	a month	19
64.	several years	19
65.	the first nine months	19
66.	a week	18
67.	10 years	18
68.	october 1987	18
69.	the years	18
70.	the past two years	18
71.	two weeks ago	18
72.	nov. 15	18
73.	the 1970s	18

74.	late friday	17
75.	the year-ago quarter	17
76.	the latest week	17
77.	nov. 1	17
78.	last summer	16
79.	the early 1980s	16
80.	this fall	16
81.	the first year	16
82.	seven years	16
83.	that day	15
84.	the 1980s	15
85.	late monday	15
86.	the 1990s	15
87.	the past five years	15
88.	three days	15
89.	nov. 30	15
90.	next spring	14
91.	15 years	14
92.	four months	14
93.	recent days	14
94.	september 1988	14

95. the past century 14

96. dec. 15 14

97. the past decade 14

98. last thursday 14

99. every day 14

100. the next few years 13

Appendix D

100 Most Common NER-Identified Entities

Here we list the most common named entities identified by the Chieu and Ng named entity recogniser (NER) for sections 02-21 of the WSJ Treebank (39,832 sentences, 1,014,129 word tokens)—as used in our parsing and generation experiments (described in Chapters 4 and 5, respectively).

	count
1. new york	513
2. san francisco	225
3. new york stock exchange	189
4. wall street	182
5. dow jones	177
6. big board	142
7. white house	128
8. hong kong	122
9. los angeles	122

10.	soviet union	80
11.	mr. bush[1]	78
12.	securities and exchange commission	78
13.	west germany	72
14.	moody 's	67
15.	merrill lynch	60
16.	supreme court	57
17.	navigation mixte	55
18.	standard & poor	52
19.	lloyd 's	52
20.	federal reserve	50
21.	sci tv	50
22.	new york city	49
23.	new jersey	47
24.	east germany	47
25.	president bush	47
26.	mr. guber	44
27.	the wall street journal	44
28.	fannie mae	44
29.	mr. gorbachev	41

[1] unlike the BBN corpus' annotators, the NER system considers personal titles (*mr.*, etc.) as NE consituents

30.	south africa	40
31.	qintex australia	38
32.	justice department	38
33.	british air	37
34.	american stock exchange	34
35.	philip morris	33
36.	freddie mac	33
37.	mr. noriega	33
38.	new hampshire	33
39.	south korea	32
40.	mr. lawson	32
41.	west german	31
42.	mr. krenz	31
43.	world series	31
44.	international business machines corp.	31
45.	air force	30
46.	united airlines	29
47.	mr. peters	29
48.	mr. roman	28
49.	american express	28
50.	commerce department	28

51.	merrill lynch capital markets	28
52.	general motors corp.	27
53.	time warner	27
54.	bay bridge	26
55.	brooks brothers	26
56.	costa rica	26
57.	control data	26
58.	las vegas	25
59.	morgan stanley	25
60.	george bush	25
61.	ual corp.	25
62.	north america	25
63.	new zealand	24
64.	state department	24
65.	mr. steinhardt	24
66.	red cross	24
67.	qintex entertainment	23
68.	ford motor co.	23
69.	american airlines	23
70.	du pont	23
71.	bear stearns	23

72.	first boston	23
73.	drexel burnham lambert inc.	23
74.	mr. jones	22
75.	general electric co.	22
76.	eastern europe	22
77.	salomon brothers inc.	22
78.	gulf power	22
79.	social security	22
80.	st. louis	22
81.	united states	22
82.	new york-based	22
83.	smith barney	21
84.	federal national mortgage association	21
85.	general motors	21
86.	san jose	21
87.	first boston corp.	21
88.	british airways	21
89.	moody 's investors service inc.	21
90.	mr. rey	21
91.	goldman , sachs & co.	21
92.	bond corp.	21

93.	mr. breeden	20
94.	mr. engelken	20
95.	mr. corry	20
96.	morgan stanley & co.	20
97.	telerate systems inc	20
98.	north american	20
99.	shearson lehman hutton	20
100.	mr. dinkins	20

Appendix E

100 Most Common Dictionary-Identified MWUs

Here we list the most common multi-word units identified by looking up a dictionary of candidates for sections 02-21 of the WSJ Treebank (39,832 sentences, 1,014,129 word tokens)—as used in our parsing and generation experiments (described in Chapters 4 and 5, respectively).

	count
1. more than	510
2. because of	390
3. such as	343
4. at least	299
5. for example	175
6. a lot	172
7. as well as	142
8. in addition	129
9. rather than	116

10.	for instance	83
11.	in fact	80
12.	even though	78
13.	instead of	78
14.	even if	76
15.	no longer	70
16.	a little	69
17.	less than	67
18.	of course	67
19.	as well	66
20.	up to	66
21.	as a result	63
22.	at par	63
23.	no one	61
24.	at all	50
25.	in order	49
26.	each other	48
27.	in the past	45
28.	for sale	45
29.	a bit	43
30.	in part	38

31.	at the time	37
32.	at home	34
33.	about half	32
34.	for the year	31
35.	after all	30
36.	in general	30
37.	for the company	29
38.	as high	29
39.	since then	28
40.	on the other hand	28
41.	in turn	27
42.	in the market	27
43.	in the world	26
44.	by contrast	26
45.	per share	26
46.	all but	24
47.	to work	24
48.	under way	24
49.	in effect	24
50.	over the weekend	22
51.	at the moment	22

52.	in place	22
53.	in particular	22
54.	in principle	21
55.	so that	21
56.	on the market	21
57.	on average	21
58.	as if	20
59.	by the company	19
60.	in the end	19
61.	to date	19
62.	for a while	19
63.	in this case	19
64.	on the issue	18
65.	over time	17
66.	in the future	17
67.	as a whole	17
68.	no doubt	17
69.	in prison	17
70.	as far	17
71.	to maturity	17
72.	just about	16

73.	once again	16
74.	on abortion	16
75.	close to	16
76.	in the country	16
77.	as president	16
78.	to the company	15
79.	to come	14
80.	due to	14
81.	in court	14
82.	as usual	14
83.	by the government	14
84.	in the meantime	14
85.	in any case	14
86.	in the case	14
87.	around the world	14
88.	that is	13
89.	at first	13
90.	to three	13
91.	for some time	13
92.	even when	13
93.	in the region	13

94.	one 's	12
95.	in mind	12
96.	in the area	12
97.	at this time	12
98.	in this country	12
99.	on the line	12
100.	now that	12