Grammatical Feature Data-Oriented Parsing

Regina Finn

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

Master of Science (M.Sc.)

to the

Dublin City University
School of Computing

Supervisors: Prof. Andy Way, Dr. Mary Hearne

January 2007

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 by research and thesis 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: Regina Finn

(Candidate)

ID No.: 50079103

Date: 30.11.06

Abstract

LFG-DOP is a powerful, hybrid model of language processing where the tree representations of Data-Oriented Parsing (DOP) are augmented with the functional representations of Lexical Functional Grammar (LFG). The result is a robust parsing model which generates linguistically informed output. However, difficulties arise in the accurate implementation of fragmentation and sampling in this model. Due to these unresolved issues, there is currently no satisfactory implementation of the LFG-DOP model.

In this thesis, we propose a backing-off to Grammatical Feature-DOP (GF-DOP). The GF-DOP model differs from Tree-DOP and LFG-DOP in that the trees are annotated with selected features extracted from the f-structure, rather than explicitly linked to corresponding f-structure units. In this way, we make use of the information available to us in the f-structure, while avoiding the problems inherent in the implementation of LFG-DOP. We aim to improve the quality of the parses generated by modeling additional functional and feature information.

Experiments on the HomeCentre corpus have shown this model to be a valuable middle-ground between the two alternative models. GF-DOP has been shown to outperform the Tree-DOP model, as a result of its ability to identify and make use of grammatical features, while maintaining the integrity of the probability model.

Contents

Chapter 1

Introduction

In the field of parsing there are two fundamental approaches employed to system development. Following the first approach, systems are based on manually collected knowledge, normally provided by linguists. The second approach derives all knowledge automatically, extracting the necessary information from treebanks created by humans (such as the Penn II treebank and the HomeCentre corpus). Data-driven approaches yield systems which tend to be easier to create and considerably less expensive to maintain than those which adhere to the first approach.

One such parsing model in the data-driven paradigm is the Data-Oriented Parsing (DOP) model; Tree-DOP combines linguistics, statistics and rules, all of which are extracted automatically from an example base. Although previous investigations of this robust model have shown it to yield high quality parses (Bod, 2003a), (Hearne, 2005), the model is limited by the representations it assumes.

An extension to this model was proposed by Bod and Kaplan (1998); the integration of Lexical Functional Grammar (LFG) into the DOP model results in a powerful hybrid model of language called LFG-DOP. In theory, augmenting the Tree-DOP model with LFG results in a more linguistically enriched model of parsing. In reality, a satisfactory implementation of this model remains elusive. For this reason, we propose a new model which is an approximation of LFG-DOP, but avoids the associated implementational difficulties.

The primary contribution of this thesis is a new model which we call Grammatical Feature Data-Oriented Parsing, GF-DOP; we base the practical implementation of this model on Tree-DOP, but the theory on LFG-DOP. By combining certain elements of the two models, the new model benefits from each of their strengths while avoiding their principal weaknesses. We hypothesize that the GF-DOP model will be able to learn grammatical features and use them to improve the quality of the parses generated when compared to the Tree-DOP model. We empirically investigate this claim using treebanks for both English and French. From the experiments carried out, we see clear evidence to support the GF-DOP Hypothesis. Furthermore, we note that the model performs better overall for English than for French. Finally, we consider extensions for the GF-DOP model.

The remainder of this thesis follows the structure given below:

Chapter 2 The Data-Oriented Parsing (DOP) model, which is based on a cognitive theory set out by Scha (1990), and first introduced by Bod (1992), is described in this chapter. We present the Tree-DOP model before broaching some issues which arise in the implementation of the model. We also discuss some theoretical issues relating to inconsistencies in the probability model.

Chapter 3 The introduction of the Lexical Functional Grammar (LFG), (Kaplan and Bresnan, 1982), formalism to the Tree-DOP model forms a powerful hybrid model of language, known as LFG-DOP Bod and Kaplan (1998). In this chapter, we present the LFG formalism and discuss the augmentation of the Tree-DOP model with LFG. We outline the reasons why the satisfactory implementation of a LFG-DOP system is impeded, presenting the theoretical and practical issues involved. With this in mind, we consider the pros and cons of both the Tree-DOP and LFG-DOP models. Consequently, we propose a new model, GF-DOP, which combines the strengths of the two previous models, while avoiding difficulties which arise in their implementation.

Chapter 4 Having motivated the need for a new model in Chapter 3, we propose the Grammatical Feature-DOP (GF-DOP) model as an alternative. Based on a corpus of sentences annotated with c- and f-structures, the GF-DOP model extracts features from f-structures and appends them to c-structure node labels. The Tree-DOP model of parsing is then applied. The new model is an approximation of the linguistically sophisticated LFG-DOP model, but is as feasible to implement as the robust Tree-DOP model. This chapter describes the GF-DOP model in detail. We present the different features used for annotation and consider several different annotation approaches. We examine where the GF-DOP model fits into the DOP spectrum, as each of the three models' strengths and weaknesses are compared. Finally, we put forward the GF-DOP hypothesis; we propose that the new model will be able to learn grammatical features accurately, and make use of this information to produce more detailed, better quality parses than the Tree-DOP model.

Chapter 5 Following our introduction of the GF-DOP hypothesis, we outline a series of experiments which are used to investigate the actual performance of the new model. This chapter presents the experimental set up used, covering the data set, a detailed breakdown of features investigated and all preprocessing steps performed on the data set. We present the parser used in the experiments and describe the various evaluation measures which will be employed. Given that the GF-DOP hypothesis comprises two assertions[1], we will divide our experiments into two tasks, and present the results of our investigation over the following two chapters.

Chapter 6 This chapter looks at the GF-DOP model's performance on task one, grammatical feature detection accuracy, in both English and French. We examine the correlation between frequency of feature occurrences and detection accuracy scores.

[1]These assertions are that (1) the new model will be able to learn grammatical features accurately, and (2) make use of this information to produce more detailed, better quality parses.

Chapter 7 In this chapter we examine the GF-DOP model's performance on task two, parse accuracy, in both English and French. We observe how the combination of structure assignment and labeling accuracy combine to boost overall parse quality.

Chapter 8 Subsequent to our examination of the GF-DOP model's performance on these two individual tasks, we focus our attention on the model's overall performance. After a contrastive comparison of the model's achievements on the English and French, we consider the merits of the GF-DOP model when compared to other approaches to the same tasks.

Chapter 9 Finally, we conclude and suggest some avenues for future work.

Chapter 2

DOP: Data-Oriented Parsing

In Data-Oriented Parsing, we construct a parse for new input from previously parsed examples of language. This parsing approach combines linguistics, statistics and examples. In this chapter we describe the DOP model, in particular Tree-DOP, with an illustration of the four principal elements which must be defined. We discuss some implementational issues, such as fragment pruning, and consider several disambiguation strategies. Finally, we discuss theoretical issues related to inconsistencies in the probability model.

2.1 What is DOP?

DOP is an "experience-based approach to natural language parsing where input sentences are analysed by referencing prior analyses of similar sentences" (Hearne, 2005). The cognitive theory behind DOP was set out by Scha (1990), and its earliest implementation developed by Bod (1992). Scha proposed to implement "performance-based" grammars, rather than "competence-based" grammars, on the basis that humans process language based on their previous experiences, as opposed to a set of acquired grammar rules. He suggested that these performance grammars should "not only contain information about the structural possibilities of the general language system, but also 'accidental' details of the actual language use in a language

community, which determine the language experiences of an individual, and thereby influence what kind of utterances this individual expects to encounter, and what structures and meanings these utterances are expected to have," (Scha, 1990).

DOP exploits an example-base (a collection of annotated sentences) created from a monolingual corpus. The DOP Model comprises four elements which must be defined: how are the examples in the annotated corpus represented? How are fragments extracted from these representations? How are these fragments recombined to derive a parse for new input? How are parses for these new analyses ranked?

2.1.1 Representations

In our DOP system the example-base consists of syntactically labelled context-free phrase structure trees, such as those given in the treebank in Figure 2.1 (A). There may be more than one occurrence of any tree.

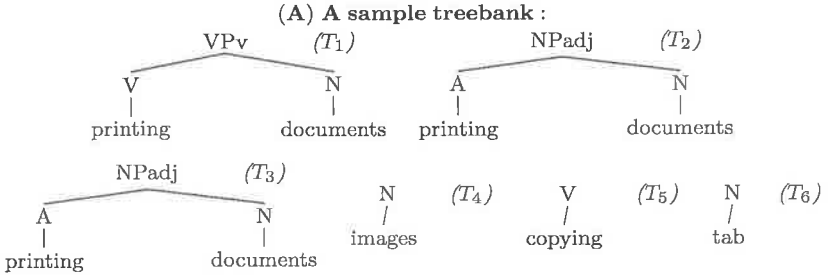
2.1.2 Fragmentation

During fragmentation, generalised fragments are extracted from the examples present in the treebank. A fragment t, extracted from T, is valid if

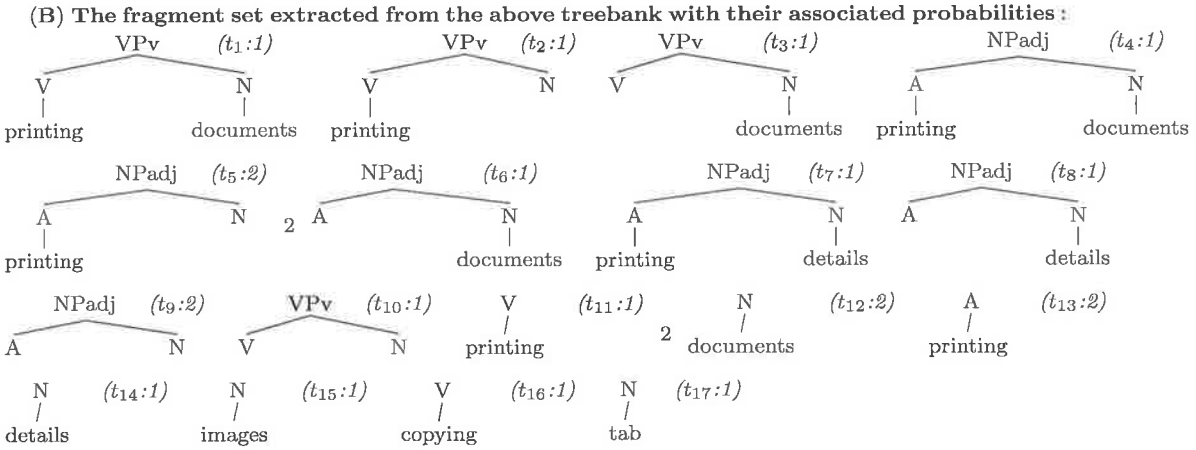
- every node in t is a node in T,
- each node in t has no children or the same number of children as the corresponding node in T,
- t comprises more than one node.

All valid fragments which can be extracted from Figure 2.1 (A) are given in Figure 2.1 (B). Examples of invalid fragments are given in Figure 2.2; fragment t_1 is invalid as it contains a node which was not present in the original tree T, Nmod. Fragment t_2 is invalid as the node VPv does not have the same number of children in this fragment as it had in the original tree T; node VPv had two children in the original

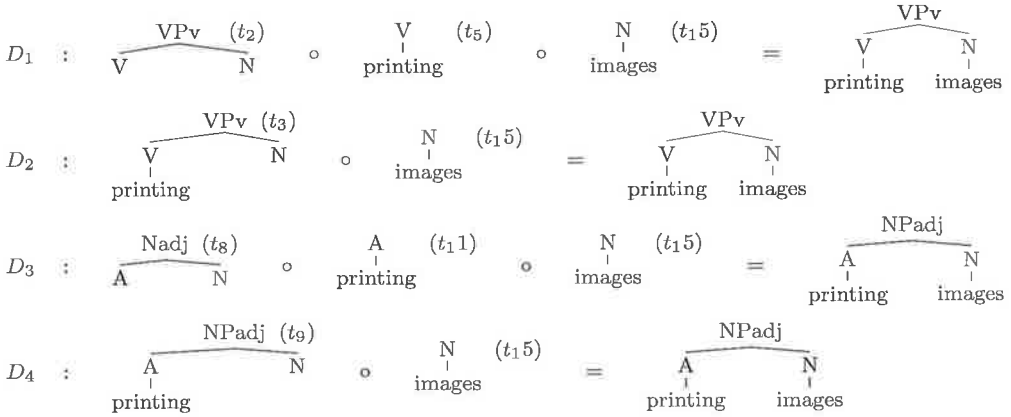
(A) A sample treebank :



(B) The fragment set extracted from the above treebank with their associated probabilities :



(C) Derivations for the string *printing images* :



(D) Calculation of probabilities of derivations for *printing images* :

$$P(D_1) = P(t_2) * P(t_5) * P(t_{15}) = \frac{1}{4} * \frac{1}{2} * \frac{1}{5} = \frac{1}{40}$$
$$P(D_2) = P(t_3) * P(t_{15}) * = \frac{1}{4} * \frac{1}{5} = \frac{1}{20}$$
$$P(D_3) = P(t_8) * P(t_{11}) * P(t_{15}) = \frac{2}{8} * \frac{2}{2} * \frac{1}{5} = \frac{1}{20}$$
$$P(D_4) = P(t_9) * P(t_{15}) * = \frac{2}{8} * \frac{1}{5} = \frac{1}{20}$$

(E) Summing derivation probabilities to calculate the most probable parse for *printing images* :

$$P(P_1) = P(D_1) + P(D_2) = \frac{1}{40} + \frac{1}{20} = \frac{3}{40}$$
$$P(P_2) = P(D_3) + P(D_4) = \frac{1}{20} + \frac{1}{20} = \frac{4}{40}$$

Figure 2.1: Illustration of (A) tree representations, (B) fragment extraction, (C) the composition process, (D) calculation of derivation probabilities and (E) parse ranking for the Tree-DOP model.

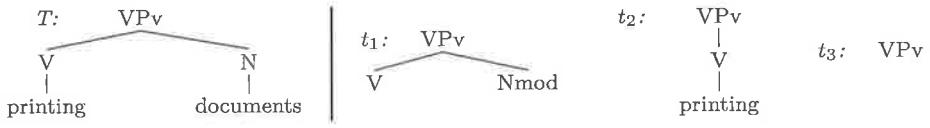


Figure 2.2: invalid fragments extracted from tree T.

tree. Fragment t_3 is invalid as it comprises a single node, while the minimum possible DOP fragment is one parent node with one child node.

Fragments are extracted from the treebank using the *root* and *frontier* operations, which are defined as follows:

- **Root:** given a copy of tree T, called T_{copy}, select a non-frontier, non-terminal node to be *root* and delete all except the subtree it dominates,
- **Frontier:** from tree T_{copy}, select a set of nodes to be *frontier* nodes and delete the subtrees they dominate.

For example, given T_{1copy}, a copy of tree T_1 in Figure 2.1 (A), we extract fragment t_{10} (Figure 2.1 (B)) in two steps; we begin by selecting node *VPv* as *root* and deleting all except the nodes dominated by *VPv*. Next we select V and N as the *frontier* set; we delete all subtrees dominated by these nodes. The result is the fragment t_{10}.

Note that the *frontier* set may be the empty set. For example, fragment t_1 (Figure 2.1 section (B)) was extracted from T_{2copy}, a copy of tree T_1 in Figure 2.1 section (A), by selecting *VPv* as *root* and the empty set as the *frontier* set.

2.1.3 Composition

To build a Tree-DOP derivation, a fragment is chosen at random to begin the derivation. This fragment may consist of either nodes labelled with syntactic categories only (such as fragment t_9 in Figure 2.1 (B)) or nodes labelled with a mixture of syntactic categories and words (such as fragment t_5 in Figure 2.1 (B)). Nodes labelled with syntactic categories constitute open substitution sites[1].

[1]An open substitution site is a node at which another fragment may be substituted.

Further fragments, whose root node labels match the current substitution site label, are composed at the open substitution sites until no further substitution sites remain and the derivation is complete. This is done using the composition operator, denoted by the symbol ∘. This is a left-most substitution operation. Several examples of derivations may be seen in Figure 2.1 (C).

If a fragment has more than one open substitution site, composition always takes place at the left-most site. In this way, we are assured that each derivation is unique. For example, given the order of the composition sequence in Figure 2.3, there is only one possible combination for the fragments shown.

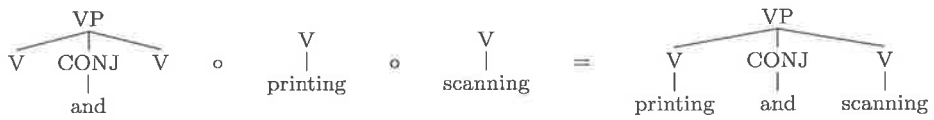


Figure 2.3: Left-most composition of a sequence of fragments which ensures that each derivation is unique.

2.1.4 The Probability Model

There is often more than one possible analysis for an input string, as can be seen in Figure 2.1 (C); there are two distinct derivations for the sentence *printing images*. These analyses are ranked in DOP using the associated probabilities, to ascertain which is the most likely analysis given the example-base.

- The probability of a fragment is its relative frequency in the example-base, as in equation (2.1):

$$P(f_x) = \frac{|f_x|}{\sum_{root(f)=root(f_x)} |f|} \tag{2.1}$$

- The probability of a derivation is the product of the probabilities of the fragments which are used to build that derivation, given in equation (2.2):

$$P(D_x) = \prod_{f \in D_x} P(f) \tag{2.2}$$

- The probability of a parse is the sum of the probabilities of the derivations which yield that exact parse, as in equation (2.3):

$$P(T_x) = \sum_{D \text{ yields } T_x} P(D) \tag{2.3}$$

Calculating the most probable derivation is not the same as calculating the most probable parse. There may be several valid derivations derived for a given string. Some of these may yield identical parses. We sum over the probabilities of each derivation of a parse to calculate the probability for that particular parse.

Figure 2.1 (D) shows the calculation of probabilities of derivations for the sentence *printing images*. In Figure 2.1 (C), we see that there were four derivations. We calculate the probability for each of these derivations by multiplying the probabilities of the fragments used to build each derivation.

Re-examining the resulting derivations in Figure 2.1 (C), we see that there are only two distinct parses: D_1 and D_2 yield the same parse (which we will call P_1), as do D_3 and D_4 (which we will call P_2). To calculate the probability of each of these parses we sum the probabilities of the derivations which yield that parse: summing the probabilities of D_1 and D_2, $\frac{1}{40} + \frac{1}{20}$, we get the probability of P_1, $\frac{3}{40}$. The probability of P_2 is equal to the sum of the probabilities of D_3 and D_4: $\frac{1}{20} + \frac{1}{20} = \frac{1}{10}$ (or $\frac{4}{40}$). The second parse, P_2 is the most probable parse for the sentence *printing images* given the example-base in Figure 2.1 (A). This calculation is illustrated in Figure 2.1 (E).

2.2 Implementing DOP

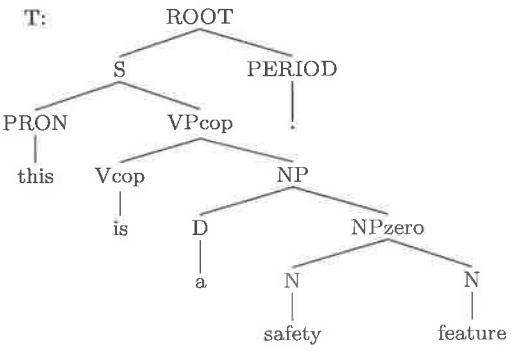
2.2.1 Fragmentation of the treebank

As mentioned in section 2.1.1, examples of previously parsed language are represented as context-free phrase-structure trees. From these examples, all possible fragments in a tree T are extracted. The number of fragments F which can be

projected from a given node $Node_T$, which has n children, $C_{T1} \ldots C_{Tn}$, can be calculated from the formula given in equation (2.4):

$$F(Node_T) = (F(C_{T1}) + 1) * \ldots * (F(C_{Tn}) + 1) \tag{2.4}$$

The number of fragments which can be extracted from the treebank is the sum of the number of fragments which can be projected from each node in the treebank. As can be seen in Figure 2.4, even a small tree can project a large number of fragments. Generating these fragments for every tree in a treebank results in a very large number of fragments: the English side of the HomeCentre corpus, which comprises only 980 trees, yields 312,132,787,415 DOP fragments.



F(ROOT)	=	(F(S)+1)	*	(F(PERIOD)+1)	=	94
F(S)	=	(F(PRON)+1)	*	(F(VPcop)+1)	=	46
F(PRON)	=				=	1
F(VPcop)	=	(F(Vcop)+1)	*	(F(NP)+1)	=	22
F(Vcop)	=				=	1
F(NP)	=	(F(D)+1)	*	(F(NPzero)+1)	=	10
F(D)	=				=	1
F(NPzero)	=	(F(N)+1)	*	(F(N)+1)	=	4
F(N)	=				=	1
F(N)	=				=	1
F(PERIOD)	=				=	1
Total fragments	=	94 + 46 + 22 + 10 + 4 + 1 + 1 + 1 + 1 + 1 + 1			=	182

Figure 2.4: Calculation of the fragments projected from each node in a tree, used to calculate the total number of fragments extractable from that tree; this tree yields 182 fragments.

Fragment Pruning

It is unrealistic to attempt to parse with a grammar of such magnitude. Pruning techniques can be used to reduce the example base size, thus enabling a more efficient implementation. There are several ways to do this; we may limit the number of open substitution sites per fragment, the number of lexical items per fragment, or the

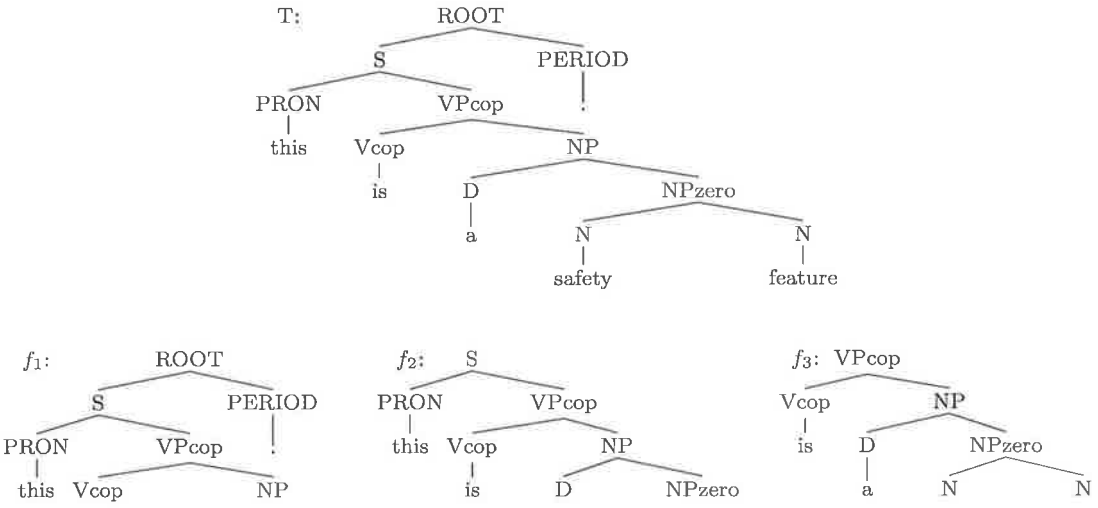


Figure 2.5: Illustration of some of the depth 3 fragments which could be generated from the tree T, by applying the *root* operation at nodes *ROOT*, *S* and *VPcop* respectively, selecting the empty *frontier* set in each case, and pruning at depth 3.

maximum fragment depth, among others. An example of the application of pruning by depth can be seen in Figure 2.5.

While some sort of pruning is required to control the size of the example base generated, discarding fragments results in reduced sensitivity to lexical and structural dependencies. Although some of the relationships discarded will be statistically very weak, other more significant relationships can potentially be lost.

The DOP Hypothesis

The DOP Hypothesis states that as we increase the size of the fragment set extracted from the training data, and include larger fragments, parse accuracy should also increase. However, this has been shown not to hold true in parsing experiments for both English and French by Hearne (2005), where decreases in f-scores were noted most frequently between fragments of depth 2 and 3, but also between fragments of depth 3 and 4. Furthermore, the incorporation of fragments of greater depths was shown to be unproductive by Bod (2001).

Given that the DOP hypothesis has been shown not to hold true always, and the

fact that an "all fragments" approach is prohibitive to an efficient implementation, we do not follow Bod (1992)'s original implementation method, or Sima'an (1995a)'s subsequent, more efficient revision. We utilise an approach which guarantees sensitivity to lexical and structural dependencies and can be also efficiently implemented: the Goodman reduction model.

2.2.2 Goodman Reduction Implementation

The Goodman reduction model (Goodman, 2003) produces a grammar which is linear in size relative to the training data. This *O(n)* grammar substantially reduces parsing time when compared to parsing a standard DOP grammar, which is exponential relative to the training data; while a DOP grammar creates all possible fragments, the Goodman model creates a Probabilistic Context Free Grammar (PCFG) which comprises (at most) eight rules per node in the training data. These PCFG rules generate the same strings with the same probabilities, and generate the same parse trees with the same probabilities. However, we must sum over several PCFG derivations for each DOP derivation.

Every node in the training grammar is assigned a unique address, e.g. A@k is the node labelled A at address k. A new non-terminal node is created for every tree in the treebank, e.g. A_k is the new non-terminal node created to correspond to node A@k. The original nodes are called "exterior" nodes, while the new non-terminals are called "interior". The number of subtrees with root node A@k is a_k. The number of subtrees with root node A is a. For any node A@k with a set of children CH, (where CH = {B@l...C@m}), we calculate a_k, the number of subtrees which have A@k as their root node, based on the number of fragments each child node yields: $a_k = \prod_{X@n \in CH}(x_n + 1)$. The relative frequency estimator is used to calculate the probabilities of these rules.

(2.5)

For a group of nodes, such as those shown in (2.5), we extract the eight PCFG rules

shown in (2.6). Their associated probabilities are shown in brackets. Goodman (2003) proves by induction that these rules and their probabilities are equivalent to a DOP grammar. These rules correspond to the fragment contexts the node group in (2.5) may appear in; in rules (1) - (4), A is an interior (non-root) node, while rules (5) - (8) show A as an exterior (root) node. In rules (1) and (5), B and C are external nodes (substitution sites in the fragment). In rules (2) and (6) B is internal (i.e. not a substitution site) to the fragment while C is an external node. Rules (3) and (7) show the reverse case, where B is the external and C is the internal node. Finally, in rules (4) and (8) B and C are both internal to the fragment.

$$\begin{array}{llllllll} (1) & A_j & \longrightarrow & BC & (\frac{1}{a_j}) & (5) & A & \longrightarrow & BC & (\frac{1}{a}) \\ (2) & A_j & \longrightarrow & B_kC & (\frac{b_k}{a_j}) & (6) & A & \longrightarrow & B_kC & (\frac{b_k}{a}) \\ (3) & A_j & \longrightarrow & BC_l & (\frac{c_l}{a_j}) & (7) & A & \longrightarrow & BC_l & (\frac{c_l}{a}) \\ (4) & A_j & \longrightarrow & B_kC_l & (\frac{b_kc_l}{a_j}) & (8) & A & \longrightarrow & B_kC_l & (\frac{b_kc_l}{a}) \end{array} \tag{2.6}$$

As previously stated, a Goodman reduction projects at most eight rules per node; this maximum number of rules is generated for each node which is internal to a tree and has two internal children. For each node which is the root of a tree in the treebank, only four rules are projected. These root nodes can never be internal to a fragment, so rules of the type (1) - (4) in (2.6) are not produced. For any node which dominates a single terminal, only two rules are produced, as terminal symbols cannot be substitution sites. Handling of rules which have more than three children is discussed in section 2.2.3.

Goodman (2003) states that a PCFG subderivation is homomorphic to a DOP tree "if the subderivation begins with an external non-terminal, uses internal non-terminals for intermediate steps, and ends with external non-terminals", as illustrated in Figure 2.6 (from (Goodman, 2003)).

2.2.3 Chart Parsing Algorithm: CKY

Standard chart parsing techniques, such as the CKY (Cocke-Kasami-Younger) algorithm (Younger, 1967), are employed to create the derivation space. However, in

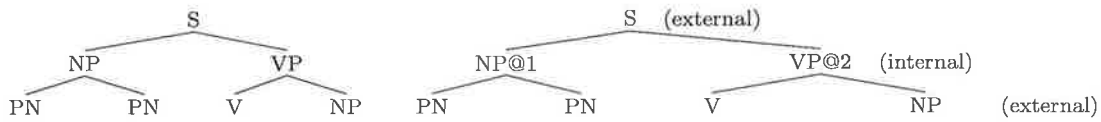


Figure 2.6: On the left, a DOP tree which is homomorphic to the PCFG subderivation on the right.

order to use the CKY algorithm, all trees must be in Chomsky Normal Form (CNF), as shown in the example in Figure 2.7. CNF trees are of one of the forms described below:

- `X` $\rightarrow$ `Y Z` (one non-terminal rewrites as two non-terminals)
- `X` $\rightarrow$ `x` (one non-terminal rewrites as one terminal)

where `X`, `Y` and `Z` are non-terminals, and `x` is a terminal. By inserting additional nodes we convert all trees to CNF, as in Figure 2.7, in advance of processing the input string.

Original Rule: `NodeA` $\rightarrow$ `NodeB NodeC NodeD`

$$\left.\begin{array}{lcl} \texttt{NodeA} & \rightarrow & \texttt{NodeB NodeB}_{\texttt{x}} \\ \texttt{NodeB}_{\texttt{x}} & \rightarrow & \texttt{NodeC NodeD} \end{array}\right|$$

Figure 2.7: Conversion of a non-CNF format fragment to CNF format.

The nodes inserted during conversion are given unique names, meaning that the subtree `NodeA` $\rightarrow$ `NodeB` $\texttt{NodeB}_x$ can only ever combine with the subtree $\texttt{NodeB}_x$ $\rightarrow$ `NodeC NodeD`. Subtrees with these artificial nodes as their root are assigned probability 1, as there is only one occurrence of this exact subtree in the treebank, and no other subtree has the same root value as this subtree; the first subtree generated by the CNF conversion in Figure 2.7, `NodeA` $\rightarrow$ `NodeB` $\texttt{NodeB}_x$, will receive the probability which was assigned to the original tree `NodeA` $\rightarrow$ `NodeB NodeC NodeD`. When these subtrees are recombined to produce the original tree, the probability of the

combined subtrees will be exactly equal to the probability of the original tree. The original DOP tree is easily retrieved by reversing the CNF conversion process and removing the inserted artificial nodes. This reversal recovers the internal structure of the original tree.

From the binarised treebank, Goodman reduction rules are extracted. These rules are in CNF which is required in order to use the CKY algorithm.

The CKY Chart Parsing Algorithm Having binarised the treebank and extracted the Goodman PCFG reduction, we are ready to apply the grammar rules to new input. We initialise the parse chart by counting the number of words in the input string. For an input string of length n words, we initialise a chart, size n by n. After this initialisation, the CKY algorithm comprises a base case, and a recursive case. The base case fills the bottom row of the chart (denoted `[i][1]`, where `i` is the column number and `1` is the row number) with unary productions; that is, rules of the form `X` $\rightarrow$ `x`, where `X` is the syntactic category corresponding to `x`, the lexical word. Only unary productions are considered at this stage. Binary productions (rules which span exactly two constituents) will be considered during the recursive case. The recursive case fills the chart bottom-up, left-to-right (column `1` to n), from the second row on (row `2` to n). At any chart position `[i][j]` (where `[i]` is the column number and `[k]` is the row number), a rule `X` $\rightarrow$ `Element1 Element2` may be inserted into the chart if there is already a rule with left hand side `Element1` at position `[i][k]` and a rule with left hand side `Element2` at position `[i+k][j-k]`, for `i` $\leq$ `k` $\leq$ `j`. A note of the combining chart positions is kept with each rule.

For example, given the grammar in Figure 2.8, and the input string *printing documents*, we can see that the rule `NPadj` $\rightarrow$ $\texttt{A}_{\texttt{[0][1]}}$ $\texttt{N}_{\texttt{[1][1]}}$ will combine with some rule with left hand side `A` at chart position `[0][1]`, and some rule with left hand side `N` at chart position `[1][1]`; there may be more than one rule in a particular chart position with the same left hand side value. Pseudocode for chart parsing an input string, of length `n` words, with this algorithm is given in Figure 2.9. Having

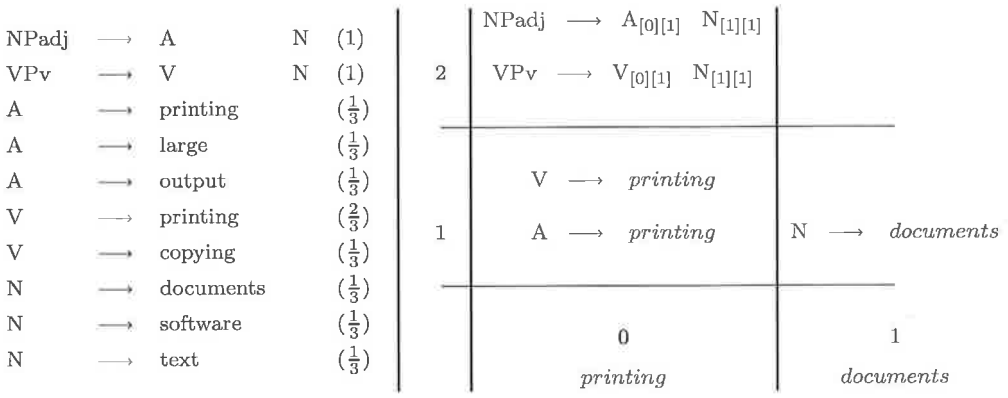


Figure 2.8: The parse space for the input string *printing documents* according to the PCFG grammar on the left.

completed the parse chart, we must now decide which of the derivations is the best parse.

The Viterbi Algorithm The Viterbi algorithm is a dynamic programming algorithm which calculates the most likely sequence of states. In the context of a parse chart, the Viterbi algorithm finds the most probable rule at each stage, and retains that rule only. In this way, we efficiently calculate the n-most likely derivations.

2.3 Disambiguation Strategies

As more than one possible analysis is generally assigned to an input string, we must have a method of selecting the best analysis. There are several methods to consider.

2.3.1 MPP: Most Probable Parse

The DOP model specifies that to select the best analysis for an input string, we must calculate the MPP. The probability of a parse is calculated by summing over the probabilities of all derivations which yield that exact parse, as in equation (2.3). An example of this in practice can be seen in Figure 2.1 (E). However, exact calculation of the MPP has been shown to be an NP-hard problem by Sima'an (1995b). Instead

```
(A) The Base Case :
for (i = 0 up to n)
   for (each unary production rule in the grammar)
      if (current rule's RHS[1] == word[i])
            insert rule into chart position [i][1]
(B) The Recursive Case :
for (j = 2 up to n)
   for (i = 1 up to n)
      for (k = 1 up to j − 1)
            Element1 = [i][k]
            Element2 = [i + k][j − k]
            for (each binary production rule X → Element1 Element2 in the grammar)
               if (current rule's RHS[1] == Element1) and (current rule's RHS[2] == Element2)
                  insert this rule into chart position [i][j]
```

Figure 2.9: Pseudocode for implementing the CKY chart parsing algorithm, where LHS refers to the Left Hand Side of a rule, RHS refers to the Right Hand Side of a rule, RHS[1] is the first element of the RHS of the rule, RHS[2] is the second element in the RHS of the rule, and n is the length of the input string.

we can calculate an approximation of the MPP using Monte Carlo disambiguation techniques (Chappelier and Rajman, 2003).

2.3.2 MPD: Most Probable Derivation

The probability of a derivation is the product of the probabilities of the fragments used to build that derivation, as in equation (2.2). Using the left-most composition operation ensures that each DOP derivation is unique. However, it is possible to derive the same tree in more than one way: Figure 2.1 (C) shows how identical derivations might be produced by different combinations of fragments. We sum over the probabilities of identical derivations to calculate the most probable parse. Computing the MPD is a more viable option than calculating the MPP either exactly, or by random sampling, but Bod (2003b) has demonstrated a 16% drop in parse accuracy when maximizing derivation probability, rather than parse probability. For this reason, we concentrate on approximating the MPP.

2.3.3 Approximating the MPP

In order to approximate the MPP effectively, a relationship must be established between the sampling frequency of a parse and its DOP probability. These sampling frequencies are used to rank parses, with the most frequently sampled parse corresponding to the MPP.

The Sampling Algorithm To begin sampling a derivation, a fragment is selected at random from the parse chart. Further random fragments, which are selected from the chart in a top-down left-to-right fashion, are composed with this derivation until there are no remaining substitution sites. The fragments are selected at random such that if the sampling probability of f_a is n times that of f_b, f_a is n times more likely to be randomly selected than f_b. When sampling DOP derivations, the distribution of parse trees in the sampled set must correspond to their DOP probability distribution. To do this, exact sampling may be employed.

Exact Sampling Chappelier and Rajman (2003) have shown that the exact sampling method guarantees the sampling probability of a derivation is equal to its DOP probability conditioned on the input string. The parse with the highest sampling probability is the most frequent parse, and the most likely MPP candidate.

The sampling probability of a fragment in chart position `[i][j]` is the DOP probability (calculated according to equation (2.1)) of that parse tree multiplied by the total sampling probability mass at each of that fragment's substitution sites, divided by the total sampling probability mass of the fragments at chart position `[i][j]` which have the same root node as that fragment, as in equation (2.7). This ensures we consider the total sampling probability mass available at each substitution site in the given fragment.

$$\textit{Sampling probability of } (f_{ij}) \quad \frac{P\,DOP\,(f_{ij}) \;*\; TSPM(f_{ij})\textit{'s substitution sites}}{TSPM[i][j] \textit{ with root } = f_{ij} \textit{ root}} \tag{2.7}$$

where $\mathtt{f}_{ij}$ is a fragment in chart position `[i][j]`, `P DOP` is the DOP probability of the fragment and `TSPM` is the total sampling probability mass of the appropriate fragment or fragments.

Controlling Sample Size As stated in section 2.3.1, calculating the MPP directly is an NP-hard task, so we carry out exact sampling to approximate the MPP. However, we must control the size of the sample set to ensure that we do not sample too many or too few derivations. One approach to controlling the size of the sample set is Bechhofer-Kiefer-Sobel method (BKS) sampling, (Chappelier and Rajman, 2003). The BKS method is a sequential sampling method, which combines two known properties:

- "For any multinomial random variable with K modalities such that $p_1 \geq \theta$ p_2 with $\theta > 1$, the probability for the most frequent modality in a sample to effectively be the most probable one is always bigger than the probability P_{min} of selecting the most probable modality in the case where all the modalities but the most probable one have equal probabilities. This lower bound P_{min} can be *a priori* computed as a function of K, θ and n, the number of samples."

- "For such a multinomial random variable, the probability for the most frequent modality in a sample to be the most probable one is always bigger than $\frac{1}{1+Z}$ where

$$Z = \sum_{i=2}^{k} (\tfrac{1}{\theta})^{(f_{[1]} - f_{[i]})}$$

and $f_{[i]}$ is the relative frequency in the samples set of the i-th most frequent parse tree in that set." (Chappelier and Rajman, 2003)

Adopting the BKS method involves sampling derivations at random, until we reach a stopping condition. This stopping condition is defined in advance, but recalculated every time we select another sample. Having already computed the parse chart with all possible derivations ($< p_{[1]} ... p_{[k]} >$) for the input string, we compute K, the

number of different parse trees for the input string. Next we define *a priori* values for $\theta = \frac{p_{[1]}}{p_{[i]}}$, and the error probability P_{err}.

Using the exact sampling method, we extract a derivation from the parse chart. This derivation is stored in an ordered list[2]. The frequencies[3] of all the different sampled parse trees are updated and the list is reranked based on the updated values. Z is recomputed. If the stopping condition has been achieved, sampling halts and the most frequent parse in the sampled set (the ordered list) is selected as the MPP. If the stopping condition has not been achieved, that is $Z > \frac{P_{err}}{1-P_{err}}$, we continue sampling further derivations.

2.3.4 Viterbi n-best Derivations

Although the Goodman reduction implementation guarantees sensitivity to lexical and structural dependencies, and enables an efficient implementation of parse space creation, there is a trade-off; for any reasonable size data set, even allowing for the efficiency of the Goodman PCFG reduction, a complete derivation chart cannot be computed for a reasonably long sentence. As we can no longer efficiently sample the parse chart, we calculate the n most probable derivations by means of the Viterbi n-best optimisation. From these n derivations, we determine p, the number of distinct parses seen. We sum over derivation probabilities to approximate the most probable parse.

2.3.5 Simplicity-DOP, Likelihood-DOP and Combinations

Bod (2000) describes two further methods of disambiguation: Simplicity-DOP, which selects the parse with the shortest derivation in terms of the fewest fragments, and Likelihood-DOP, which selects the parse constructed from the most probable fragments. He reports that although Simplicity-DOP is outperformed by Likelihood-

[2]All further derivations extracted will be stored in decreasing order of occurrence, i.e. the derivation which is sampled the most often will be at the top of the list, with derivations seen less often following in their appropriate order.

[3]That is the frequency in the ordered list, or how many times this exact parse has been sampled so far.

DOP (f-scores of 87.049 and 89.39 respectively for fragments of depth 14 or less), its results are outstanding for such a simple model. In addition, he notes that the best parse trees selected by Simplicity-DOP are very different to the best parse trees determined by Likelihood-DOP. With this in mind, he proposes two further hybrid models: Simplicity-Likelihood-DOP (SL-DOP) and Likelihood-Simplicity-DOP (LS-DOP), (Bod, 2003a).

SL-DOP vs LS-DOP The SL-DOP Model selects the simplest tree from the n most probable trees. The LS-DOP model selects the most likely tree from the n shortest derivations. If n is equal to 1, SL-DOP is equal to LS-DOP, as there is only one most likely derivation and only one most simple derivation to choose from. As n increases, SL-DOP converges to Simplicity DOP. Likewise, LS-DOP converges to Likelihood-DOP. Empirical investigation by Bod (2003a) showed an increase in accuracy for both models as the value of n increases from 1 to 12. From *n=14* on, the accuracy of SL-DOP decreases, converging to Simplicity-DOP. However, the accuracy of LS-DOP continues to improve, and also converges to Likelihood-DOP.

2.4 Bias in the Probability Model

The probability of a fragment is taken to be its frequency in the fragment set divided by the total number of fragments in the fragment set which have the same root node as that fragment. This method of estimating fragment probabilities has been shown to be both biased and inconsistent by (Johnson, 2002); this relative frequency estimator introduces a bias in favour of larger trees.

Figure 2.10 shows a practical example of this bias. The treebank in Figure 2.10 (A) contains three trees; trees T_1 and T_2 are identical. Their relative frequency in the treebank is $\frac{2}{3}$. The relative frequency of tree T_3 in the treebank is $\frac{1}{3}$. All fragments which can be extracted from this treebank are shown in Figure 2.10 (B). All derivations for the input string *printer paper* are shown in Figure 2.10 (C). There

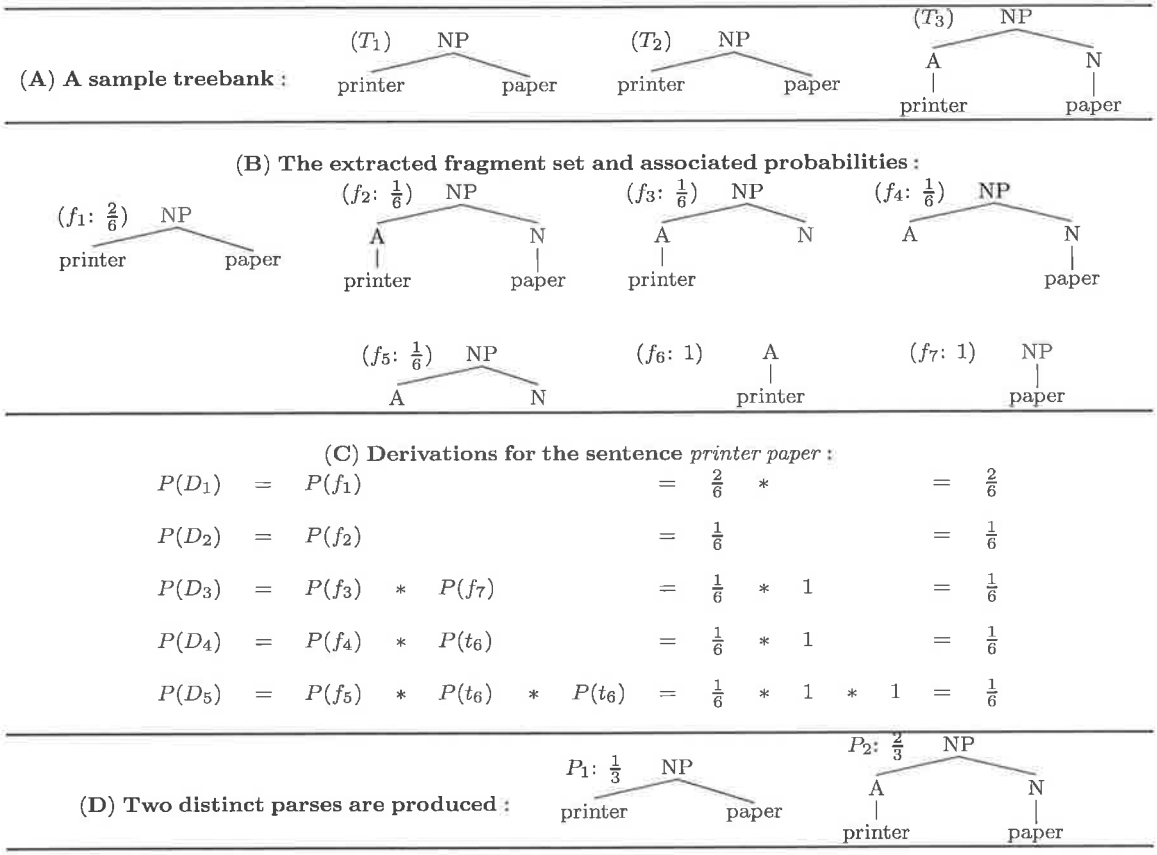


Figure 2.10: Illustration of bias in the probability model. Although the probability model has selected P_2 ($\frac{2}{3}$) to be the more probable than P_1 ($\frac{1}{3}$), evidence in the treebank shows P_1 ($\frac{2}{3}$) to be more probable than P_2 ($\frac{1}{3}$).

are 5 derivations for this string, yielding 2 parses. Summing over the probabilities of identical parses, we establish the probability of each of the two parses: parse P_1 is yielded by derivation D_1 only, its probability is $\frac{1}{3}$. Parse P_2 is yielded by derivations D_2, D_3, D_4 and D_5. Its probability is $\frac{2}{3}$. The DOP probability model therefore selects P_2 to be the most probable parse for the input string *printer paper*.

The relative frequency estimator has selected a large tree, with a probability which was summed over several derivations, as the most probable analysis, showing a bias for larger trees. Selection of parse P_2 as the most probable parse for *printer paper* given the treebank in Figure 2.10 (A) is in direct conflict with the evidence in the treebank. The probability distribution in this example does not reflect the actual distribution of the treebank; thus it is inconsistent. Notwithstanding this shortcom-

ing, our DOP experiments, using the relative frequency estimator, demonstrate very high parse quality.

2.5 Summary

In this chapter we have given a detailed description of the DOP model, defining the four main elements with emphasis on the Tree-DOP application. We presented some of the approaches taken in implementing the DOP model and motivated our choice of method. We outlined a variety of disambiguation techniques before identifying some shortcomings in the probability model.

Chapter 3

LFG-DOP: Lexical Functional Grammar Data-Oriented Parsing

In this chapter we introduce the Lexical Functional Grammar (LFG) formalism, describe how it is integrated into the Tree-DOP model to form a powerful hybrid model of language processing known as LFG-DOP, and consider some of the difficulties surrounding the satisfactory implementation of this model.

3.1 What is LFG?

LFG is a constraint-based theory of language which was developed by Joan Bresnan and Ron Kaplan, (Kaplan and Bresnan, 1982). They advocated the view that there is more to syntax than can be expressed using only phrase-structure trees. Two syntactic levels of representation are assumed by the original LFG model: constituent-structure (or c-structure), which encodes phrasal dominance and precedence relations, and functional-structure (or f-structure) which encodes syntactic predicate-argument relations. The c-structure is represented by a context-free, phrase-structure tree, while the f-structure is represented as an attribute-value matrix. LFG c-structures are annotated with equations, as illustrated in Figure 3.1, which are resolved to produce the attribute-value pairs which form the corresponding

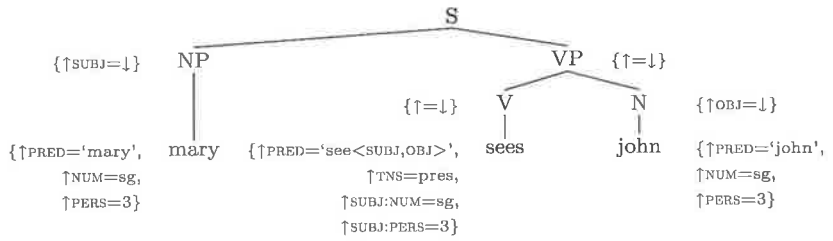


Figure 3.1: Illustration of a c-structure annotated with equations which are resolved to form the corresponding f-structure.

f-structure. We examine the equations present in Figure 3.1 for a moment; in each rule, the ↑ refers to the mother of the current, annotated c-structure node, while the ↓ refers to the annotated non-terminal node itself. The equation {↑SUBJ=↓} on the node NP is interpreted as this NP's mother node's SUBJ is equal to this NP. The equation {↑=↓} on the VP node indicates that this node is the head of its mother node, S. Likewise, {↑=↓} on the V node indicates that this node is the head of its mother node, VP; that is, all equations and information relating to the V node percolate up to its mother node, VP, and from there on to VP's mother node, S.

In this example, V dominates the terminal node *sees*. *Sees* is annotated with four equations: {↑PRED='see<SUBJ,OBJ>', TNS=pres, ↑SUBJ:PERS=3, ↑SUBJ:NUM=sg}. These equations translate into the following information and constraints; the predicate of this verb is *see*, this verb must take an obligatory SUBJ and OBJ and this verb's form is in the present tense. Furthermore, the SUBJ required by this verb must be in third person singular form.

From the equation {↑SUBJ=↓} on the node NP, we know that this NP is functioning as S's SUBJ. This NP dominates the terminal node *mary* whose equations tell us the surface form is *mary*, and that *mary* is a third person singular form.

To this point, we have resolved that *sees* requires a third person singular form SUBJ and some OBJ. In addition, there are equations which tell us *mary* is functioning as the SUBJ of *sees* and is a third person singular form. *Sees* has not placed any particular constraints on the OBJ required; the c-structure shows us there is an OBJ present (*john*). These equations will be successfully resolved to produce an

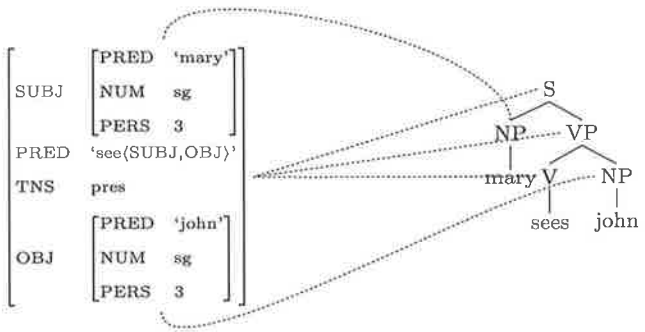


Figure 3.2: ϕ-linked c-structure and f-structure representations for the sentence *mary sees john.*

f-structure, as seen in Figure 3.2. The relationship between the c- and f-structures is illustrated using ϕ-links which connect corresponding c-structure nodes and f-structure units.

The successful resolution of c-structure equations results in an f-structure containing detailed information regarding grammatical features and functions. This information is encoded as sets of attribute-value pairs. Attributes may have one of four value types; the value may be an atomic symbol, as in [PERS 3] where PERS is the attribute and 3 the atomic value. The value may be a semantic form, as in [PRED 'mary'] where PRED is the attribute, and *mary* is the semantic form value. The value may be an f-structure, as in Figure 3.2, where SUBJ is an attribute, and its value is the f-structure unit containing attribute-value pairs defining the SUBJ. Finally, the value may be a set of f-structure units, an example of which can be seen later, in Figure 3.9.

The f-structure shown in Figure 3.2 was generated as the result of the successful resolution of the equations on the c-structure shown in Figure 3.1. All equations were successfully resolved; all obligatory arguments were present and unification was achieved. The result is a well-formed f-structure: that is, an f-structure which conformed to each of the three well-formedness conditions, discussed in section 3.1.1.

3.1.1 Well-formedness conditions

There are three well-formedness conditions which every f-structure must adhere to, as defined in (Butt et al., 1999), page 6:

> **Uniqueness:** In a given f-structure, a particular attribute may have at most one value.
>
> **Completeness:** An f-structure is *locally complete* if and only if it contains all the governable grammatical functions that its predicate governs. An f-structure is *complete* if and only if it and all its subsidiary f-structures are locally complete.
>
> **Coherence:** An f-structure is *locally coherent* if and only if all the governable grammatical functions it contains are governed by a local predicate. An f-structure is *coherent* if and only if it and all its subsidiary f-structures are locally coherent.

We can see that the example in Figure 3.2 adheres to these three conditions: no attribute has more than one value, all governable grammatical functions required to satisfy subcategorisation requirements are present, and all grammatical functions present are indeed governed by some predicate. An f-structure which violates any of these three conditions is invalid. The examples in Figure 3.3 each violate one of these constraints.

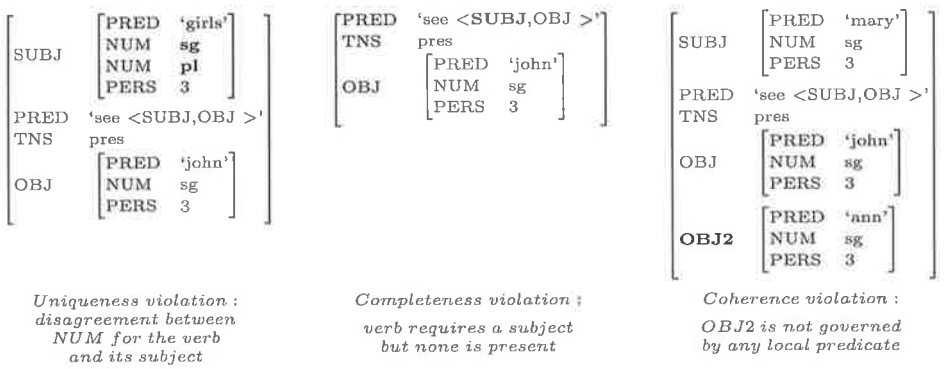


Figure 3.3: F-structures which each violate a well-formedness condition.

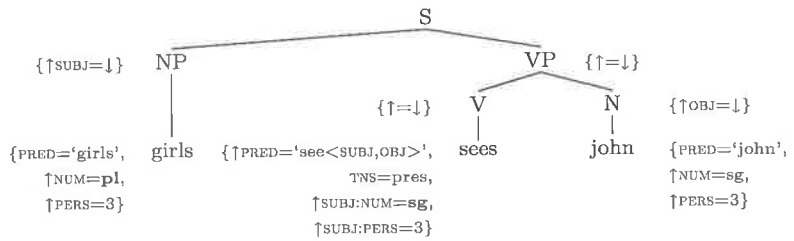


Figure 3.4: Illustration of a c-structure annotated with equations which cannot be fully resolved to produce a well-formed f-structure due clashing NUM values.

In the case where c-structure equations do not unify, as those illustrated in Figure 3.4, an ill-formed f-structure will be generated. Figure 3.4 shows a clash of equation values which must unify to satisfy the first well-formedness condition; the verb *sees* requires a third person singular form SUBJ. However, *girls*, which is functioning as SUBJ in this example, is third person plural. The ill-formed f-structure which would result can be seen in the leftmost f-structure presented in Figure 3.3.

3.2 What is LFG-DOP?

Although the Tree-DOP model achieves excellent parse accuracy (Hearne, 2005), the power of the model is limited by the corpus representations it assumes. Phrase-structure trees reflect only surface-level syntactic phenomena, and do not accurately describe many other aspects of language. However, the incorporation of a linguistic formalism such as LFG, which is known to be beyond context-free, brings the potential to capture many previously unhandled issues, such as number, person or gender agreement violations, and output a far more informative parse, for example showing grammatical functions and re-entrancies.

The LFG-DOP Model, proposed by Bod and Kaplan (1998), is a robust, constraint-based approach to parsing. The tree representations of DOP are augmented with the functional representations of LFG. As for the Tree-DOP Model, we formalize the four elements which define the model.

3.2.1 Representations

The example-bank consists of syntactically labelled context-free phrase-structure trees (c-structures), which are ϕ-linked to their corresponding f-structures, such as the sample LFG-DOP representation given in Figure 3.5 (A). There may be more than one occurrence of any given LFG-DOP representation in an example-base.

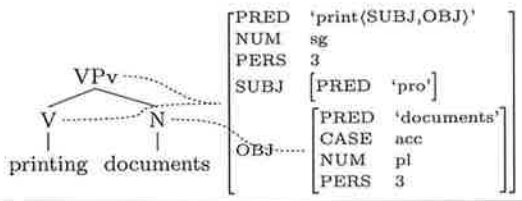
3.2.2 Fragmentation

Fragments are extracted using the *root* and *frontier* operations and must satisfy the same conditions for validity as Tree-DOP fragments, as given in section 2.1.2. However, these fragmentation operations are extended for LFG-DOP in order to decompose f-structures appropriately. These extensions, as defined in Bod and Kaplan (1998), are as follows:

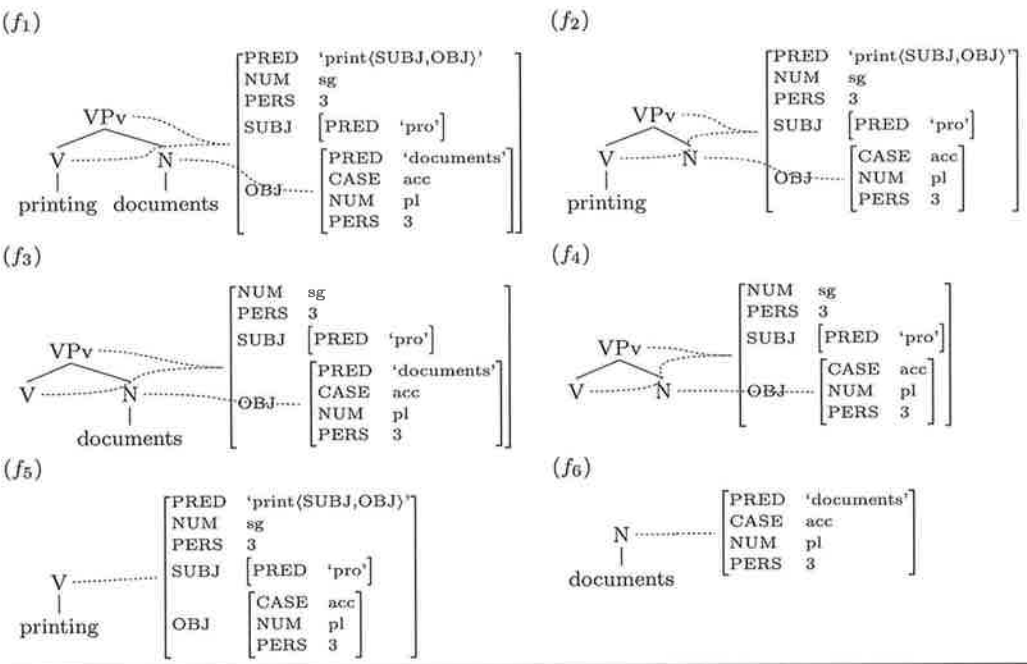
> "When a node is selected by the *Root* operation, all nodes outside of that node's subtree are erased, just as in Tree-DOP. Further, for LFG-DOP, all ϕ links leaving the erased nodes are removed and all f-structure units that are not ϕ-accessible from the remaining nodes are erased.
>
> In addition, the *Root* operation deletes from the remaining f-structure all semantic forms that are local to f-structures that correspond to erased c-structure nodes, and it thereby also maintains the fundamental two-way connection between words and meanings.
>
> As with Tree-DOP, the *Frontier* operation then selects a set of frontier nodes and deletes all subtrees they dominate. Like *Root*, it also removes the ϕ links of the deleted nodes and erases any semantic form that corresponds to any of those nodes."

The concept of ϕ-accessibility is defined such that an f-structure unit is retained if it is ϕ-linked to a node in the c-structure, or if it is contained within an f-structure which is ϕ-linked to a node in the c-structure.

(A) An LFG representation :



(B) The fragment set extracted from the representation in (A) using the *root* and *frontier* operations :



(C) LFG-DOP fragments extracted from fragment f_6 using the *discard* operation :

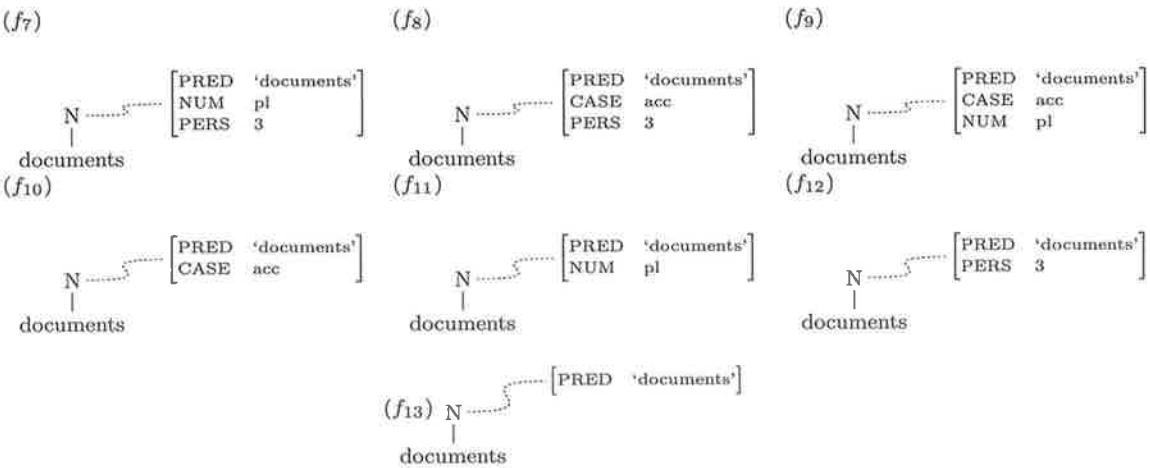


Figure 3.5: Illustration of (A) an LFG-DOP representation, (B) fragments which can be extracted using the *root* and *frontier* operations, and (C) the *discard* operation.

LFG-DOP fragments which can be extracted from the representation in Figure 3.5 (A) using the *root* and *frontier* operations are presented in Figure 3.5 (B). Let us examine how fragment f_6 would be extracted: node N is selected by the *Root* operation, and all nodes which are not dominated by this node are deleted. All ϕ-links leaving the erased nodes are also removed. Any f-structure unit which is not ϕ-accessible from the remaining nodes is removed. In the remaining f-structure, any semantic forms which are local to f-structures corresponding to deleted c-structure nodes are also deleted. This completes the *Root* operation. The *Frontier* operation selects a set of frontier nodes, in this case the node labelled *documents*, and all subtrees dominated by the frontier set are deleted. Any ϕ-links or semantic forms corresponding to the deleted nodes are also removed from the remaining f-structure. The result of these operations is the fragment labelled f_6.

As can be seen in Figure 3.5 (B), LFG-DOP fragments provide a lot of contextual detail. Although this information helps us to construct accurate, grammatical parses, it also reduces the number of candidates suitable for any composition. Indeed there may be no candidate fragments. We are constrained by the level of detail provided. If we could relax some of these constraints, we could propose more fragments as candidates for composition.

Further fragments may be extracted using the *discard* operation: attribute-value pairs which are not ϕ-linked to nodes in the remaining c-structure, and are not PRED values which correspond to remaining c-structure terminals, may be deleted. No changes are made to the c-structure or the ϕ-links, only f-structure attribute-value pairs are affected. An example of additional fragments which can be extracted from fragment f_6 (Figure 3.5 (B)) are illustrated in Figure 3.5 (C): fragments f_7, f_8 and f_9 each have one constraint relaxed. Fragments f_{10}, f_{11} and f_{12} have two constraints discarded, while fragment f_{13} has all attribute-value pairs deleted, with the exception of the PRED feature.

3.2.3 Composition

As for Tree-DOP, composition takes place at the left-most open substitution site of the fragment c-structure. However, unification must also take place in the f-structures, ensuring that the three well-formedness conditions are met by the composed fragments. An example of composition resulting in a well-formed fragment is given in Figure 3.6. Uniqueness, coherence and completeness are maintained in the composed fragment.

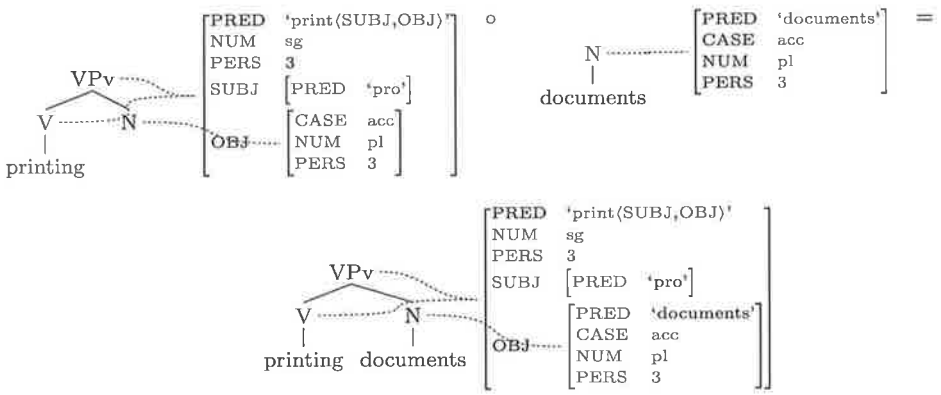


Figure 3.6: Composition of these two fragments results in a well-formed LFG-DOP fragment which satisfies each of the well-formedness conditions.

An example of composition resulting in an ill-formed fragment is given in Figure 3.7. Uniqueness is not maintained in the resulting composed fragment, as the CASE attribute cannot have both the value 'acc' and the value 'nom'. Any attribute can have at most one value.

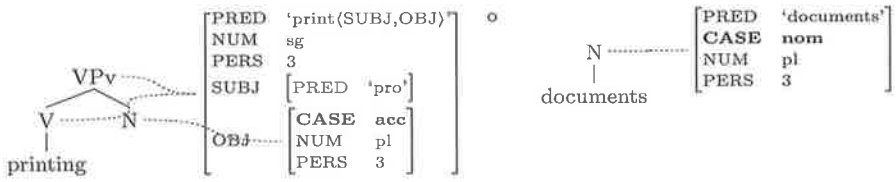


Figure 3.7: Composition of these two fragments results in an ill-formed LFG-DOP fragment, due to a uniqueness violation with regard to the value of the attribute CASE.

We can increase the robustness of the LFG-DOP model via *discard*-generated

fragments. Where no parse is possible using *root-* and *frontier*-generated fragments only, due to a violation of the unification condition, we can compose *discard*-generated fragments, thereby avoiding this violation, as in Figure 3.8.

Strings whose parses rely on discard-generated fragments are considered ungrammatical with respect to the corpus. These strings may be ill-formed, or may be well-formed but unaccounted for in the training data.

3.2.4 The Probability Model

In constructing a Tree-DOP derivation, a fragment is chosen at random to begin the derivation, and successive fragments are substituted into the left-most open substitution site, assuming their node labels match. The probability of each valid parse is calculated from the associated probabilities. The probabilities of all valid parses which can be constructed sum to 1.

The probability of an LFG-DOP derivation is calculated in the same way as that of a DOP derivation: the probability of a fragment is its relative frequency in the example-base, as in equation (2.1). The probability of a derivation is the product of the probabilities of the fragments which were used to construct that derivation, as in equation (2.2). The probability of a parse is the sum of the probabilities of all derivations which yield exactly that parse, as in equation (2.3).

However, as demonstrated in Figure 3.7, not all constructed derivations yield valid parses. All parses must fulfil the category matching requirements of the c-structure composition, but may fail to adhere to the well-formedness requirements, resulting in an invalid parse with respect to the corpus. If our LFG-DOP probability model works in the same way as the Tree-DOP probability model, in that the probability distribution is defined according to fragment root nodes, the probabilities of all LFG-DOP derivations should also sum to 1. Given that not all LFG-DOP derivations are valid parses according to the model, the probabilities of valid parses no longer sum to 1; that is, there is no longer an accurate probability distribution as the model 'leaks' probability mass, Abney (1997). It is possible to normalise fragment

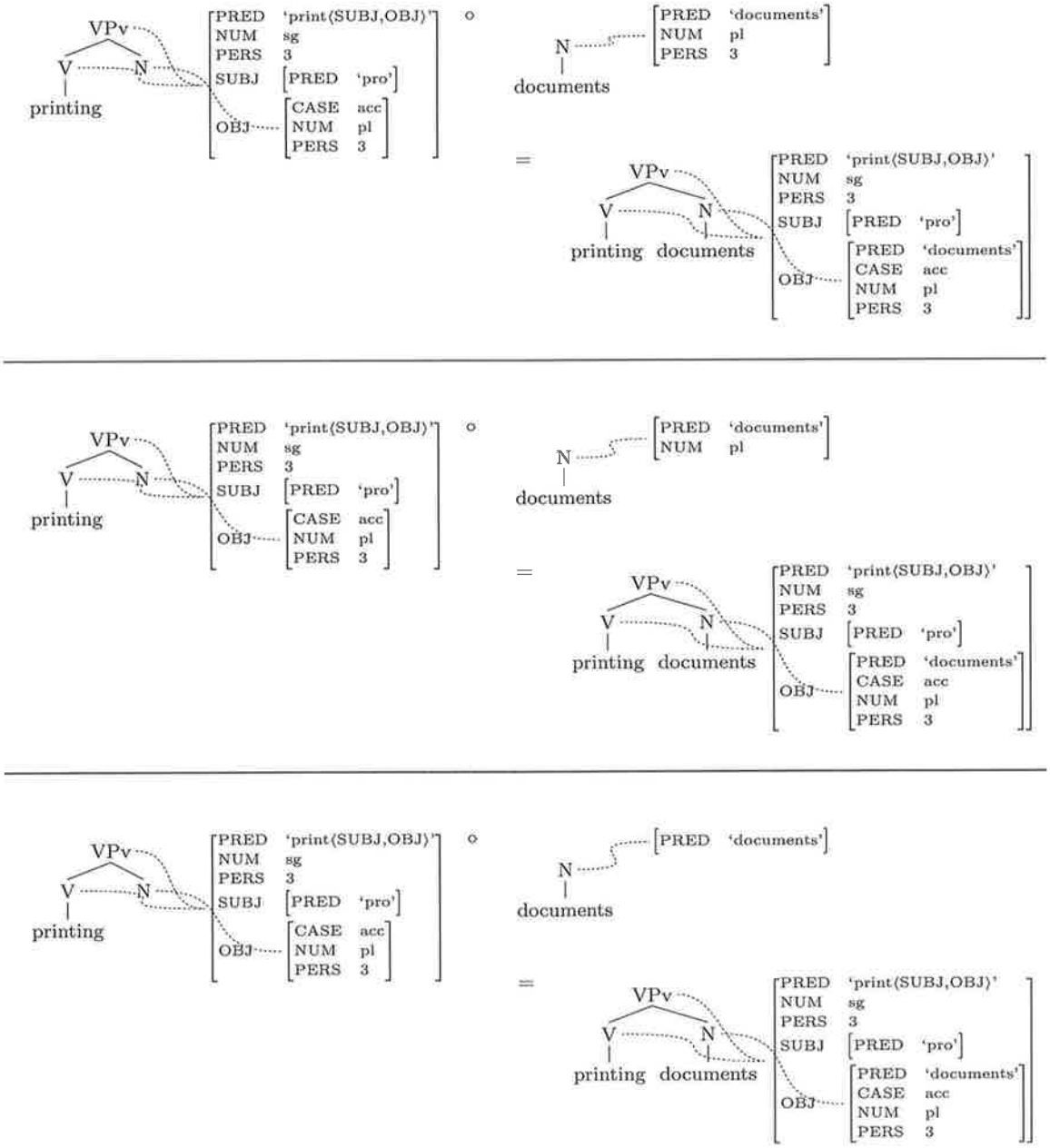


Figure 3.8: Composition with discard-generated fragments which produce parses. In the first composition sequence, a single constraint, CASE, has been relaxed in the substituting fragment, enabling composition without violation of any well-formedness conditions. In the second composition sequence, two constraints, CASE and PERS, have been relaxed. In the final sequence, three constraints have been relaxed, only the PRED feature remains.

probabilities over the probabilities of valid parses, however Abney (1997) observes that this normalisation merely masks the fact that using relative frequency estimation to establish grammar probabilities where context-sensitive dependencies are encoded does not yield good weights. There is currently no thoroughly satisfactory solution to this problem.

3.3 Challenges for LFG-DOP

Although it is clear that the LFG-DOP model is a robust, accurate model of language, the implementation of a system based on this model is not a straightforward task. We are faced with many, as yet, unresolved issues relating to a satisfactory LFG-DOP implementation.

3.3.1 Theoretical Issues

Hearne (2005) observes that although recursive and re-entrant structures occur frequently in language, the *root* and *frontier* operations as defined by Bod and Kaplan (2003) do not sufficiently describe how to handle these structures. Additionally, information which is unrelated to the corresponding c-structure may remain in the f-structure after the *root* and *frontier* operations are applied.

Consider the example in Figure 3.9 (from (Hearne, 2005)): the SUBJ of the sentence is the f-structure unit labelled f_2, with PRED 'LED', which corresponds to the noun *LED* in the c-structure. The f-structure defines *yellow* to be the ADJUNCT of *LED*. In addition, *LED* is defined to be the SUBJ of *yellow*, denoted by co-indexation. The value of the SUBJ in this instance is the outer f-structure unit labelled f_2.

When extracting fragments from the c-structure in this LFG-representation, the fragment $A \rightarrow$ *yellow* is encountered. By applying the *root* operation to the c-structure, all except the subtree dominated by A is deleted. All ϕ-links which

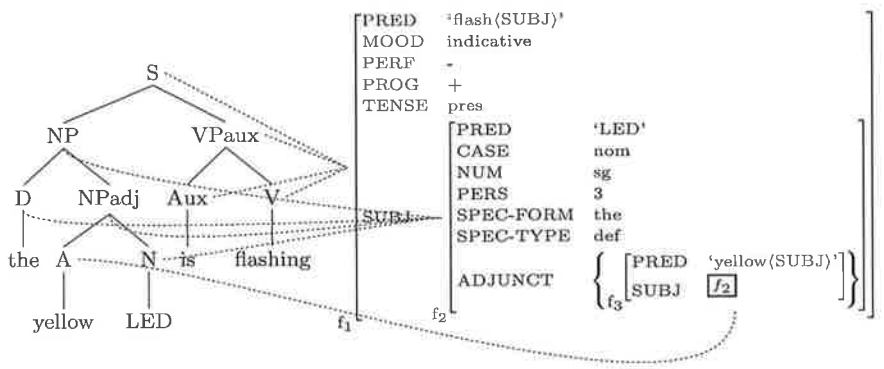


Figure 3.9: An example LFG representation for the string "*the yellow LED is flashing*".

connect deleted nodes to the f-structure are subsequently deleted. All f-structure units which are no longer ϕ-accessible from the c-structure are deleted. Finally, any semantic forms remaining in the f-structure, which are local to f-structure units corresponding to deleted c-structure terminals are deleted. In this example, the empty set is selected as the *frontier* set.

By following the definition given, we cannot arrive at a conclusive f-structure corresponding to the fragment $A \rightarrow$ *yellow*. The definition of ϕ-accessibility states that an f-structure unit is retained if it is ϕ-linked to a node in the c-structure, or if it is contained within an f-structure which is ϕ-linked to a node in the c-structure. From examining the original f-structure, we can see that the f-structure unit which would appear to correspond to the fragment in question, labelled f_3, does not contain the f-structure unit labelled f_2. However, f_2 is the value of an attribute within f_3. In this case, it is unclear how we should ascertain the f-structure which corresponds to the fragment $A \rightarrow$ *yellow*. A similar problem arises when re-entrant structures are encountered.

As mentioned previously, the current fragmentation method, based on the definition of ϕ-accessibility, leads to the preservation of f-structure attribute-value pairs which may not be merited given the evidence in the c-structure. In an f-structure which satisfies each of the well-formedness conditions, every f-structure unit is ϕ-accessible from the outermost f-structure unit. Each inner f-structure unit is present

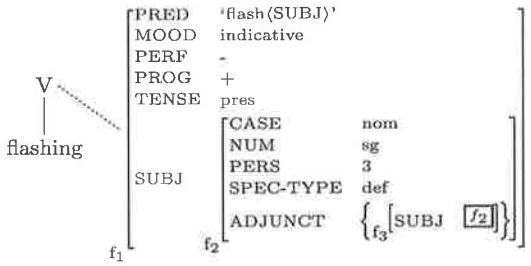


Figure 3.10: An LFG-DOP fragment extracted from the LFG representation given in Figure 3.9 where the constraints present in the f-structure are determined using the ϕ-accessibility criterion.

because it is a requirement of some subcategorisation frame. Therefore, it must be linked to some outer f-structure unit, a link which percolates back to the outermost f-structure unit. As a result, any c-structure fragment which contains at least one node which is ϕ-linked to the outermost f-structure unit also has the ability to ϕ-access every inner f-structure unit.

Consider the example in Figure 3.10, (from (Hearne, 2005)), representing the fragment $V \rightarrow$ *flashing*. The f-structure specifies that the SUBJ of the verb *flashing* must have an ADJUNCT. Given the little evidence presented by the c-structure fragment, this is perhaps an over-constraint, for English at least. It is possible that this constraint would be justified for a similar fragment in another language.

An alternative fragmentation process is proposed by Hearne (2005), whose method deals with this constraint overspecification (while maintaining the language independency of the model). In this modified fragmentation process, f-structure units which are supported by evidence in the c-structure fragment are retained. Initially, the c-structure fragment is extracted using *root* and *frontier*, as for Tree-DOP, while retaining the entire f-structure given in the original representation. All f-structure units and their associated attributes which are not ϕ-linked from some c-structure node are deleted, unless that f-structure unit is the value of an attribute which is subcategorised for by a PRED value whose corresponding terminal is dominated by the current fragment root node in the original representation. Where there are floating

f-structure units, retain the unit containing both floating f-structures with their attributes. Any semantic forms which are not associated with a remaining c-structure terminal are deleted, as in the revised fragment in Figure 3.11. The attribute-value pair ADJUNCT {[SUBJ f_2]} was not supported by evidence in the c-structure, and has been deleted from this revised fragment.

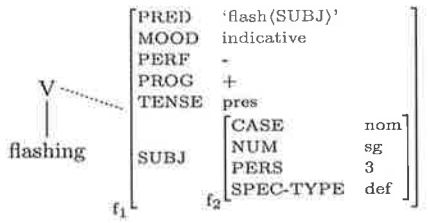


Figure 3.11: An LFG-DOP fragment extracted from the LFG representation given in Figure 3.9 where the constraints present in the f-structure are determined using the support criterion.

While this approach avoids some overspecification, we are still left with the problem of how to distinguish between constraining and informative features. Upon extracting the fragment on the right of Figure 3.12 from the LFG-DOP representation on the left, we learn that the verb *see* must have a singular SUBJ, and also a singular OBJ. Admittedly this particular example is a language-dependent issue, but, nonetheless, an unresolved problem which will occur in other languages.

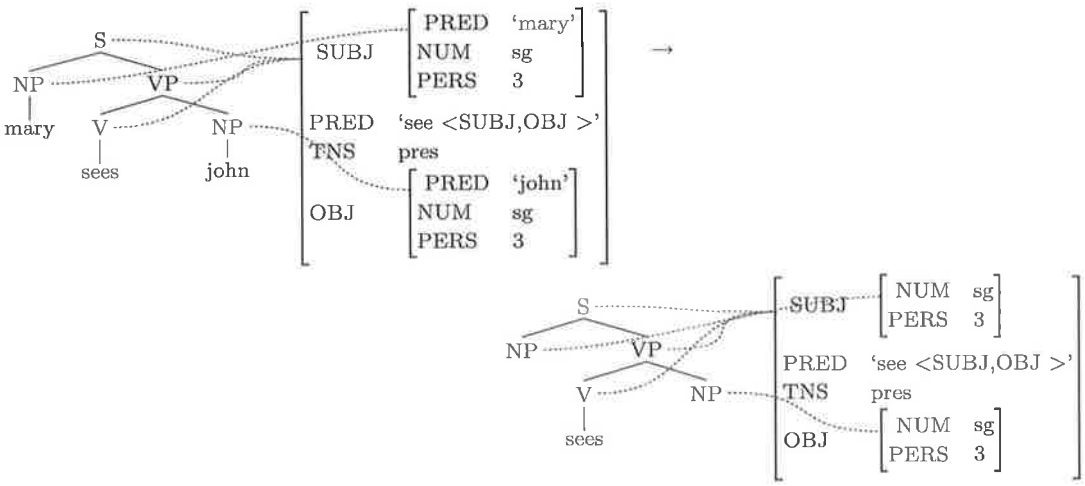


Figure 3.12: Illustration of a fragment extracted, based on the supporting evidence in the c-structure, which is still overconstrained.

In section 3.4.3 we will propose an alternative model which makes use of the information present in the f-structure, while managing to avoid the difficulties presented here.

3.3.2 Practical Issues

While composition is a simple matter in Tree-DOP, based on local substitution, LFG-DOP composition must also deal with unification which is a global operation. In any LFG-DOP derivation, unification must be successful for every substitution. As in the example given in Figure 3.13, we see that a derivation may fail because of a single unification violation.

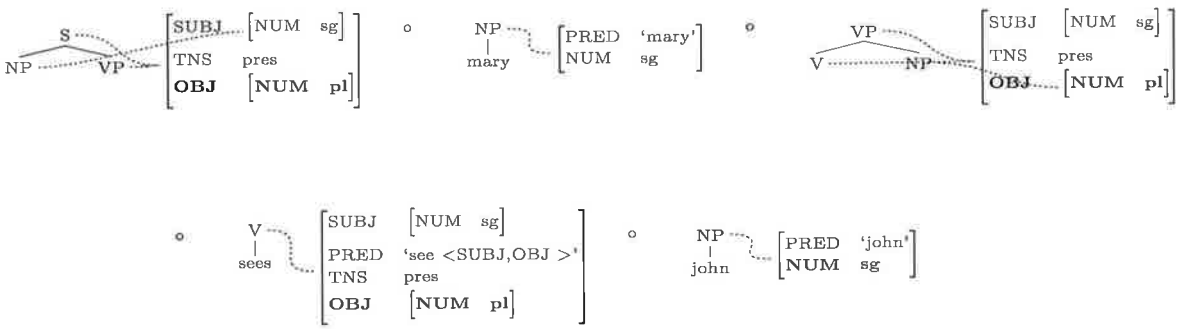


Figure 3.13: An LFG-DOP derivation which fails due to a unification violation during the final substitution; the previously composed fragments expect a plural form OBJ in the final substitution. Instead a singular form fragment is proposed. This composition violates the uniqueness condition.

Category-matching is not the only well-formedness condition which must be satisfied during composition; in LFG-DOP it is not the case that any fragment whose root node corresponds to the left-most open substitution site is eligible for composition at this point. As a result, the probability distribution for each competition set is not the same as the distribution of the fragments in the fragment set.

A second difficulty arises from the fact that a partial derivation cannot be checked for completeness; it can only be appraised when there are no further substitution

sites, and no more compositions are possible.

> "The stochastic branching process by which derivations are constructed does not necessarily yield valid representations even when the other well-formedness conditions have been verified during fragment selection" (Hearne, 2005).

This is the reason for normalisation of the probability mass which is assigned to valid derivations.

Although a derivation must be complete to judge for completeness, uniqueness and coherence can be executed during or after the sampling process. Three different probability models, M_1, M_2 and M_3, are proposed by Bod and Kaplan (1998).

Model M_1 This model is an extension of the Tree-DOP model, where only the category matching condition is enforced during sampling. Immediately after a complete derivation has been sampled, that sampled derivation is appraised for uniqueness, completeness and coherence to determine whether it is valid or not. We calculate the competition sets for Model M_1 according to equation (3.1), where `CS` is the competition set, `LSS` is the leftmost substitution site, and `f:root(f)` is the set of fragments whose root nodes match leftmost substitution site of derivation $\mathtt{D_{i-1}}$.

$$\mathtt{CS_{M_1}} = \{\mathtt{f{:}root(f)} \;=\; \mathtt{LSS(D_{i-1})}\} \tag{3.1}$$

Model M_2 The sampled (partial) derivations are checked for uniqueness at each sampling step in this model. Fragments must satisfy the category-matching and uniqueness conditions in order to be considered eligible for the competition set. After sampling, completed derivations are appraised for completeness and coherence. Having completed derivation step D_{i-1}, we calculate the competition sets for Model M_2 according to equation (3.2).

$$\mathtt{CS_{M_2}} = \{\mathtt{f{:}root(f)} \;=\; \mathtt{LSS(D_{i-1})} \wedge \mathtt{unique(D_{i-1} \circ f)}\} \tag{3.2}$$

Model M_3 The competition sets in this model are checked for uniqueness and coherence at each sampling step. Fragments must satisfy the category-matching, uniqueness and coherence conditions in order to be considered eligible for the competition set. Again, completeness can only be checked after completing a derivation. Having completed derivation step D_{i-1}, we calculate the competition sets for Model M_3 according to equation (3.3).

$$\mathtt{CS}_{M_3} = \{\mathtt{f{:}root(f)} = \mathtt{LSS(D_{i-1})} \wedge \mathtt{unique(D_{i-1} \circ f)} \wedge \mathtt{coherent(D_{i-1} \circ f)}\} \quad (3.3)$$

Even using these probability models does not correct the problem of 'leaked' probability mass (Abney, 1997); the only way to avoid sampling invalid derivations is to check uniqueness, completeness and coherence at each step. However, this fourth model does not exist as completeness can only be checked once an entire parse has been obtained. Until this issue is resolved, we will be unable to sample in an accurate, efficient way.

Exact sampling for the Tree-DOP model, as described in section 2.3.3, is a relatively straightforward task. The exact probability of sampling fragment f at chart position `[i][j]`, with root node *VP*, shown in Figure 3.14, is calculated by equation (2.7). We compute the DOP probability of fragment f, which is then multiplied by the sampling probability mass available at each of f's substitution sites, `[i][k]`,V and `[i+k][j-k]`,NP. This is then divided by the total sampling probability mass available at chart position `[i][j]` for fragments with root node *VP*.

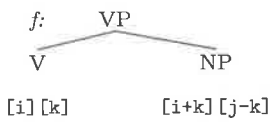


Figure 3.14: Tree-DOP fragment, f, occurring at position `[i][j]` on the parse chart.

Unfortunately, this is not such a simple task when applied to LFG-DOP frag-

ments. Let us try to associate the fragment shown in Figure 3.14 with the appropriate ϕ-linked f-structure representation. Calculating the exact probability of sampling fragment f at chart position `[i] [j]`, with root node *VP*, we attempt to follow the same procedure described above. However, we cannot know the sampling probability mass at substitution site `[i+k] [j-k]`, NP until `[i+k] [j-k]`, NP is the leftmost substitution site.

In exact sampling, we stop sampling when enough samples have been seen. To proceed with this, we must first establish how many valid parses there are in total. Until all constraints in all parses have been resolved, we cannot count the valid parses.

3.4 Comparing Tree-DOP and LFG-DOP

3.4.1 Advantages of Tree-DOP

The primary advantage of the Tree-DOP model is that it is possible to develop an implementation which adheres to the model in practice. No difficulties arise in extracting fragments from the treebank. Dealing with recursive and re-entrant structures are not a problem when fragmenting c-structures only. The probability distribution for each competition set is the same as the distribution of the fragments in the fragment set; that is, the probability model can be accurately implemented. It is unnecessary to consider the generation of additional fragments using the *discard* operation, as category-matching is the only condition which is enforced during composition.

One weakness in the Tree-DOP model is the limited form the representations assume. This model produces parses which are grammatical with respect to a linguistically simple treebank. As illustrated in Figure 3.15, a parse may be generated which is grammatical, given the grammar extracted from the treebank, but is otherwise an ungrammatical example of language. This issue is dealt with by the LFG-DOP model through the unification of features and functional information

which are present in the f-structures.

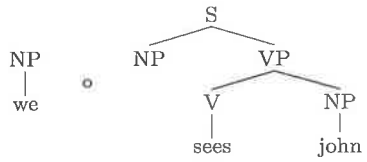


Figure 3.15: This sequence of composed fragments would be considered to yield a valid parse according to the Tree-DOP model.

3.4.2 Advantages of LFG-DOP

The LFG-DOP model improves on the Tree-DOP model. This strength comes from the representations it assumes; the additional information available in the f-structures ensures that the output produced is grammatical with respect to the corpus, and also to real world examples of language. The unification violation, shown in Figure 3.16, would be considered invalid according to the LFG-DOP model (but could still be handled).

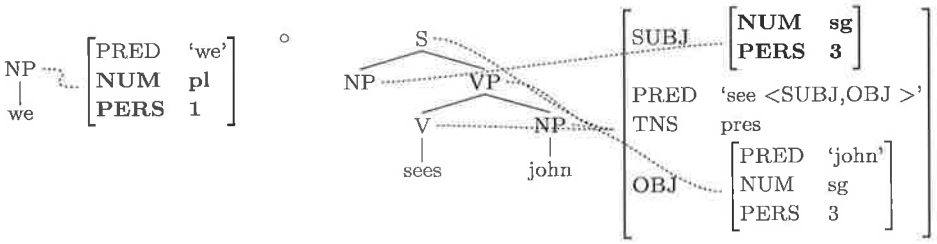


Figure 3.16: This sequence of composed fragments would be considered to yield an invalid parse according to the LFG-DOP model, due to violations of the uniqueness condition.

Although the LFG-DOP model is clearly the preferred model in terms of its ability to produce linguistically accurate output, there remain several obstacles to be overcome before a satisfactory implementation is in place. With this in mind, we propose an alternative model which has the ability to make use of linguistic functions and features, as in LFG-DOP, but avoids the implementational difficulties inherent in this model.

3.4.3 Considering an alternative model

The model we propose is the GF-DOP model; Grammatical Feature Data-Oriented Parsing. This model appends features extracted from f-structures to c-structure category labels. We then apply the Tree-DOP model to the transformed treebank. As summarised in Table 3.1, this model generates linguistically informed output, while maintaining the integrity of the probability model. We present this model in detail in Chapter 4.

	Tree-DOP	GF-DOP	LFG-DOP
Output corresponds to the probability model (i.e. "no leaking probability mass)	√	√	X
Can identify and make use of grammatical features	X	√	√

Table 3.1: Summary of the GF-DOP model as compared to the Tree-DOP and LFG-DOP models

3.5 Summary

In this chapter we introduced the constraint-based theory of language known as LFG, and described how it can be used to augment the Tree-DOP model, resulting in a robust, linguistically informed model of parsing; LFG-DOP. We presented the LFG-DOP model formally, and discussed some of the theoretical and practical issues which impede a dependable implementation. Finally, in considering the pros and cons of the Tree-DOP model compared to the LFG-DOP model, we propose a third model which combines the strengths of both models, while avoiding the difficulties inherent in the accurate implementation of the LFG-DOP model.

Chapter 4

GF-DOP: Grammatical Feature Data-Oriented Parsing

Given the theoretical and practical difficulties inherent in the LFG-DOP model, as outlined in Chapter 3, we propose the GF-DOP model as an alternative. This chapter describes the GF-DOP model in detail. It discusses the different annotation approaches considered, and classifies the features identified. We describe where this model fits into the DOP spectrum, as strengths and weaknesses of the models are compared and we develop our hypothesis, which will form the basis for experiments presented in later chapters. Finally, we consider future extensions of the GF-DOP model.

4.1 What is GF-DOP?

The GF-DOP model can be seen as an extension of the Tree-DOP model, and an approximation towards LFG-DOP. It combines the robustness of the DOP model with some of the linguistic competence of LFG. This model exploits a corpus of annotated c-structures: features are extracted from f-structures and appended to the c-structure category labels. As this model extends the Tree-DOP model, category-matching is the only restriction imposed on fragments which are candidates for

composition. No restrictions are placed on the label form, so labels which incorporate features incur no extra computational cost; no changes to the model are required to handle the adapted labels. The Tree-DOP model is applied to the transformed treebank. This model can be as accurately and efficiently implemented as the Tree-DOP model, and produces linguistically detailed output, based on identification and incorporation of grammatical functions and features.

4.2 Feature Classification

LFG f-structures contain informative features (for example LAYOUT-TYPE may specify that a sentence is a header, a listitem or is unspecified) and functional information (such as SUBJ and OBJ which describe the grammatical features of the constituents in question). For the treebank used in our experiments (the Xerox Parc HomeCentre corpus, described in section 5.1), 77 features were identified in the English data set, and 80 were identified in the French data set. These features were grouped into five classes:

- grammatical function features, e.g. SUBJ, XCOMP
- atomic features, e.g. NUM=sg, PERS=3
- lexical features, e.g. PRON-FORM=this, SPEC-FORM=a
- non-grammatical function features which have an f-structure containing a group of features as their values, e.g. TNS-ASP[MOOD=imperative, PERF=-, PROG=-], NTYPE[GRAIN=count]
- predicates, e.g. PRED 'be<[XCOMP]>'[pro].

The classification of these features can be seen in Table 4.1 (PRED has been excluded).

Grammatical Functions	Atomic Features		Lexical Features	Non-Grammatical Functions
ADJUNCT	ABBREV	NEG-FORM	AUX-SELECT	ARG-EXT
APP	ACONSTR	NUM	COMP-FORM	ARGS-INT
APSEC	ADEG-DIM	NUMBER-TYPE	CONJ-FORM	ASPECT
ASPEC	ADEGREE	PASSIVE	FORM	CONJ-FORM-COMP
COMP	ADJUNCT-LAYOUT	PCASE-TYPE	DEP	FIN
COMP-EX	ADJUNCT-TYPE	PERF	NEG-FORM	NON-DEP
COMPOUND	ADV-TYPE	PERS	PCASE	NTYPE
OBJ	ANIM	POL	PRON-FORM	PRECONJ-FORM
OBJ2	APOS	PREDET-TYPE	PREDET-FORM	TNS-ASP
OBL	ATYPE	PREVERB-OBJ	PREDET	
OBL-AGT	AUX-FORM	PROG	PRT-FORM	
OBL-COMP	CASE	PRON-TYPE	SPEC-FORM	
PRON-INT	CONJOINED	PROPER		
PRON-REL	CONJTYPE	PSEM		
SPEC	DEIXIS	PTYPE		
SUBJ	EMPH	REFL		
TOPIC-INT	EMPHASIS	SPEC-TYPE		
TOPIC-REL	FOO	STATUS		
XCOMP	GEND	STMT-TYPE		
	GERUND	STRESSED		
	GRAIN	TEMPORAL		
	INF	TENSE		
	INV	TIME		
	LAYOUT-TYPE	TYPE		
	MOOD	VCONSTR		
	NE	VFORM		
	NEG	VTYPE		

Table 4.1: Classification of all features identified in the data set (excluding PRED).

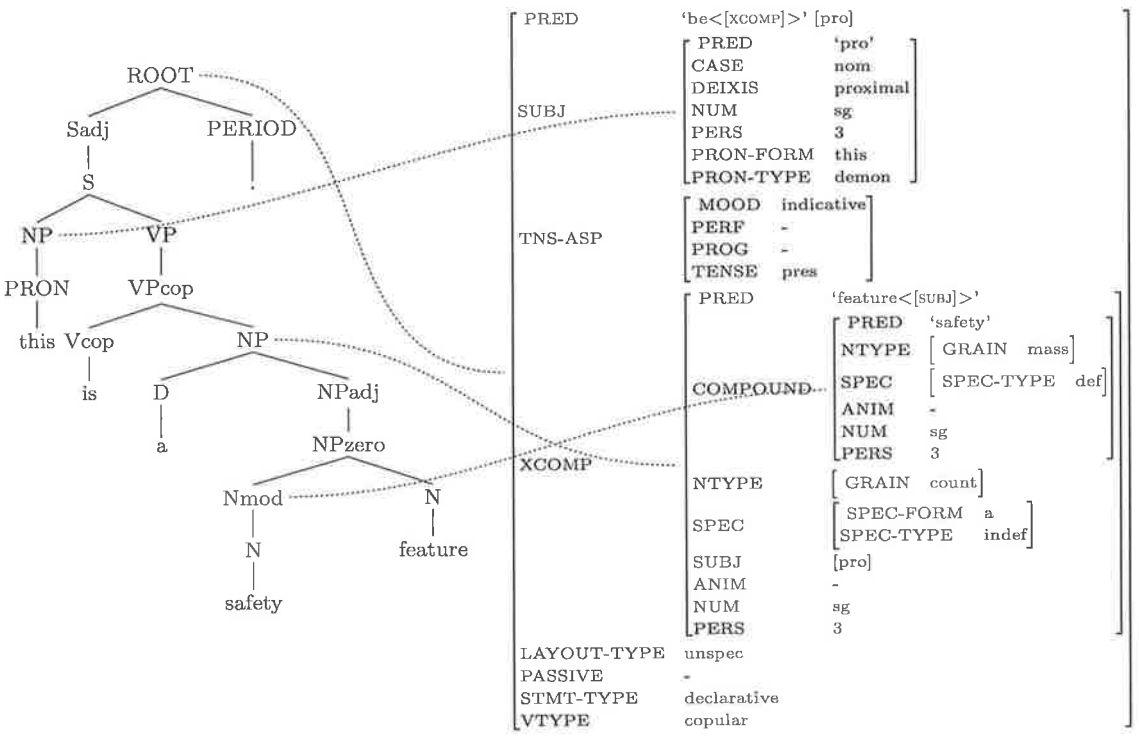


Figure 4.1: A c-structure with its corresponding ϕ-linked f-structure, from which we extract features.

4.3 Annotating Trees with Grammatical Features

We begin with a corpus of sentences, represented as c-structures. Each c-structure is ϕ-linked to its associated f-structure. An example of such a representation can be seen in Figure 4.1. We divide the discussion of feature annotation into two groups: annotation of functions and annotation of all other features. The annotation of all other features is further subdivided in two: annotation with root-based features and annotation with preterminal based features. Subsequently, we present an example of annotation for each feature class.

Let us begin by considering how we might annotate the tree with function features. By examining the f-structure, we can see how the constituents of the c-structure function. Beginning with the main verb in the sentence, described in the outermost f-structure unit as PRED 'be<[XCOMP]>'[pro], consider its subject, described by the f-structure unit labelled SUBJ. Its ϕ-link shows that the leftmost

constituent in the tree, NP→PRON→*this*, functions as the SUBJ of the main verb in this sentence. Now that we have identified the function of this constituent, we must decide how we will denote this.

In the first approach considered, we annotate each non-terminal node which forms part of constituent whose function is SUBJ. The result of this annotation approach can be seen in Figure 4.2. Both the NP and PRON nodes have labels indicating that they function as the SUBJ of the sentence. The function is appended to the category label using the logical "and" symbol (∧) which we reserve for this purpose only. This reserved symbol enables us to evaluate parses in several different ways and this is discussed in section 5.4.

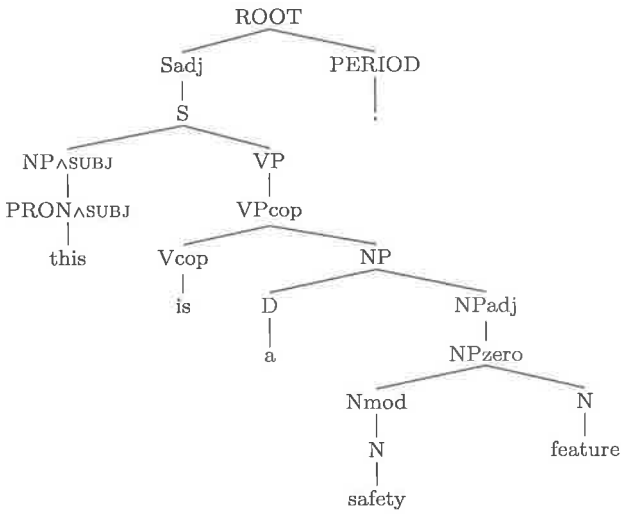


Figure 4.2: A c-structure annotated with SUBJ on all non-terminal nodes which form the SUBJ constituent.

Upon reviewing the annotated tree in Figure 4.2, we see that multiple annotations of this sort result in falsely inflated annotation frequencies; for example, in a given sentence, there may be three constituents which function as the SUBJ of various predicates. If each node which forms part of the SUBJ constituents is annotated with the SUBJ label, we will count more than three SUBJ annotations in the sentence. Annotating several nodes as SUBJ where there is actually only one subject function results in linguistically inaccurate representations. A side-effect of this is a difficulty in calculating the exact frequency of feature annotations. This issue has no effect on the implementation of the GF-DOP model *per se*, but the distribution

of features is pertinent in the analysis of parses produced. These figures assist us in the interpretation of scores achieved and can help us to draw correlations between parse quality, coverage and feature occurrences.

In order to ensure that we have the same number of functional annotations as functions, we make only one annotation per function. We place this annotation on the highest node in the constituent which corresponds to the function in question. All nodes dominated by this annotated node are part of the constituent which fulfils this function, but they are no longer individually annotated. For example, the annotated tree shown in Figure 4.2 will now have only one SUBJ label, placed on the NP node, as illustrated in Figure 4.3.

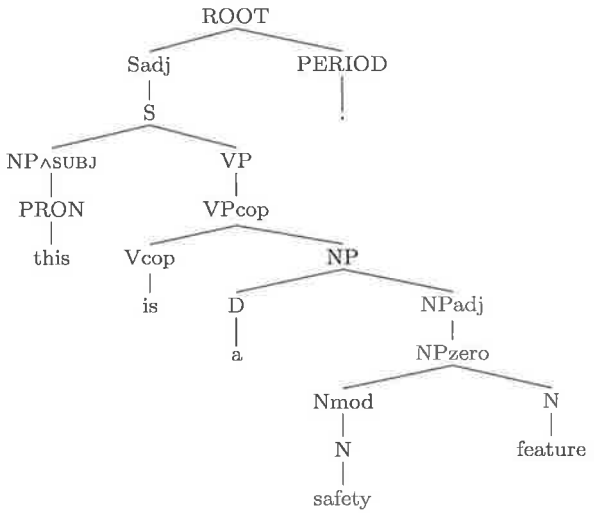


Figure 4.3: A c-structure annotated with SUBJ on the topmost node of the SUBJ constituent.

There may be more than one SUBJ (or other such function) in a sentence. All constituents which fulfil a function are annotated with the appropriate label. Further examination of the f-structure in Figure 4.1 shows that there are two constituents in the c-structure which function as SUBJ; the main verb of the sentence, *be*, requires a SUBJ (PRED 'be<[XCOMP]>'[pro]), as does *feature* which operates as an XCOMP in this sentence (PRED 'feature<[SUBJ]>'). In this example, the same nodes serve as the SUBJ of both *be* and *feature*. The highest node of the appropriate constituent receives an annotation for each function it serves, as can be seen in Figure 4.4.

As there may be more than one of a particular function in any given sentence, we

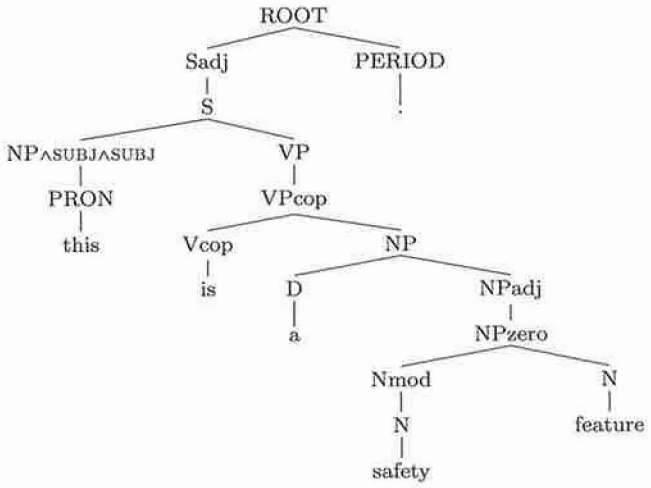


Figure 4.4: A c-structure annotated with SUBJ on the topmost node of each of the SUBJ constituents. In this case, one node receives two annotations as it is the topmost node in a constituent which serves as two functions.

elected to use more specific labels, indicating the exact relationship between nodes. The nodes are annotated with their function and the lemma of the predicate whose function they fulfil. For the example c-structure given in Figure 4.1, the annotations on the node NP become SUBJ_of_be and SUBJ_of_feature.[1] An illustration of this type of annotation can be seen in Figure 4.5.

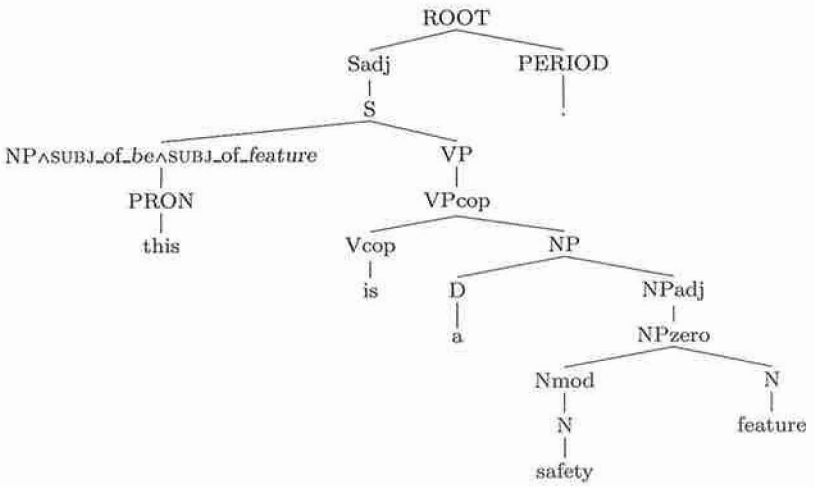


Figure 4.5: A c-structure annotated with specific labels which incorporate the function SUBJ and the related lemma.

[1]The implementation of the annotation approach requires us to maintain single token node labels. For this reason, we use underscores in place of spaces in node labels.

This approach is used to annotate all functions. However, not all features can be annotated in this way. All other features are divided into two groups:

- root-based annotations
- preterminal-based annotations.

Features in the first group provide information about one or more constituents, and are annotated in the same way as functions; they are placed on the uppermost node which dominates all appropriate constituents. For example, the features LAYOUT-TYPE or STMT-TYPE describe the structure of the sentence; these annotations will be placed on the root node of the constituents they dominate. The feature ANIM indicates whether or not some constituent describes an animate concept; this annotation will be placed such that it dominates all constituents with this animate property. Examination of the f-structure in our original example, Figure 4.1, shows that there are two ANIM features; we see that the XCOMP has an ANIM value of -, which will be annotated on the highest node in the XCOMP constituent, NP. The second ANIM feature occurs within the COMPOUND f-structure unit, indicating that *safety*, which forms a compound with *feature*, is also inanimate. This is annotated on the node Nmod. Figure 4.6 illustrates this type of feature annotation.

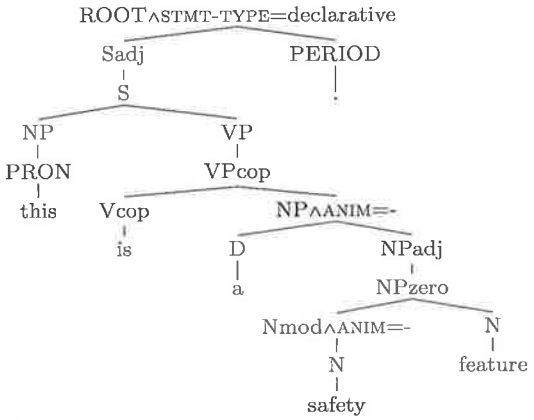


Figure 4.6: A c-structure with root-based feature annotations: STMT-TYPE on ROOT (root of the constituent whose STMT-TYPE is declarative), ANIM on NP, (root of the constituent whose ANIM value is negative). Nmod is annotated as it is the highest node in a constituent which is also ϕ-linked to an f-structure unit with ANIM.

The second group consists of features which are related to particular terminal nodes in the c-structure. These features are annotated on preterminal nodes. Annotating these features higher up the c-structure would enforce inappropriate restrictions on many more fragments than are necessary. Figure 4.7 illustrates the second type of feature annotation.

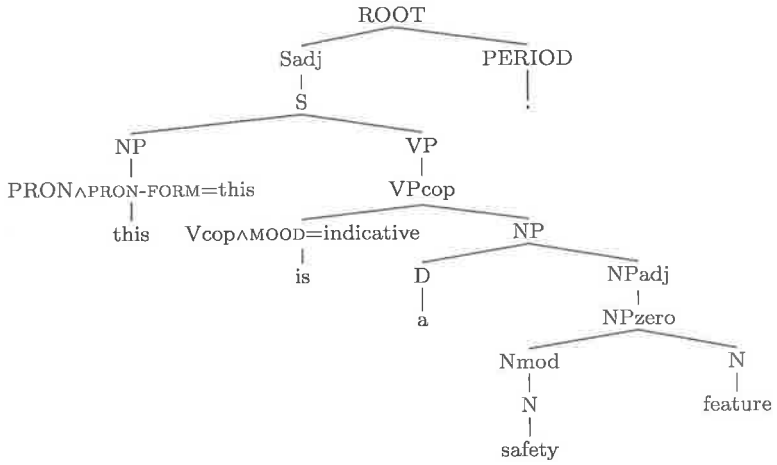


Figure 4.7: A c-structure with preterminal-based feature annotations: PRON-FORM on PRON - this annotation indicates that *this* is the most appropriate word to appear in this position, and MOOD on Vcop - this annotation signals that an indicative verb should appear in this position.

As can be seen in this example, the feature PRON-FORM is annotated on the node which directly precedes the pronoun. In a fragment where this node is an open substitution site, this annotation indicates which pronoun fragments are most appropriate for composition at this site. Likewise, MOOD is annotated on preterminals corresponding to only those nodes which have a modal aspect; annotating an entire constituent with this function would wrongly imply that, for example, a nominal object of a verb phrase also has a modal aspect. The sub-division of atomic and lexical features into features which are placed on the root of dominated nodes and preterminal nodes is shown in Tables 4.2 and 4.3.

Atomic			Lexical
ACONSTR	DEIXIS	PREVERB-OBJ	COMP-FORM
ADEG-DIM	FOO	PROG	COMP-FORM-ANAPH
ADEGREE	GEND	PSEM	CONJ-FORM
ADJUNCT-LAYOUT	GERUND	PTYPE	CONJ-FORM-COMP
ADJUNCT-TYPE	LAYOUT-TYPE	SPEC-TYPE	FORM
ADV-TYPE	NE	STATUS	PCASE
ANIM	NEG	STMT-TYPE	PREDET-FORM
APOS	PASSIVE	TEMPORAL	SPEC-FORM
ATYPE	PCASE-TYPE	TENSE	
CASE	PERF	TYPE	
CONJOINED	POL	VTYPE	
CONJTYPE	PREDET-TYPE	VCONSTR	

Table 4.2: ROOT: atomic and lexical features which are annotated on the root of the dominated constituents.

Atomic		Lexical
ABBREV	NUMBER-TYPE	AUX-SELECT
GRAIN	PERS	NEG-FORM
INF	PRON-TYPE	PRECONJ-FORM
INV	PROPER	PRON-FORM
MOOD	REFL	
NUM	VFORM	

Table 4.3: PRETERMINAL: atomic and lexical features which are annotated on preterminal nodes.

4.3.1 Grammatical Functions

Using ϕ-linked f-structure units, we identify functions of constituents within the c-structure. The leftmost NP in the c-structure representation in Figure 4.1 functions as the SUBJ of the main verb in the sentence, *be*. As there may be more than one occurrence of a particular function in any given sentence, we use specific labels, indicating the exact relationship between nodes; a node is annotated with its function, and also with the lemma of the predicate governing it. We construct an annotation which denotes that the node NP functions as the SUBJ of *be*: NP$\wedge$SUBJ_of_*be*. We place this annotation on the topmost node in the constituent which corresponds to the function in question. All nodes dominated by this annotated node form the

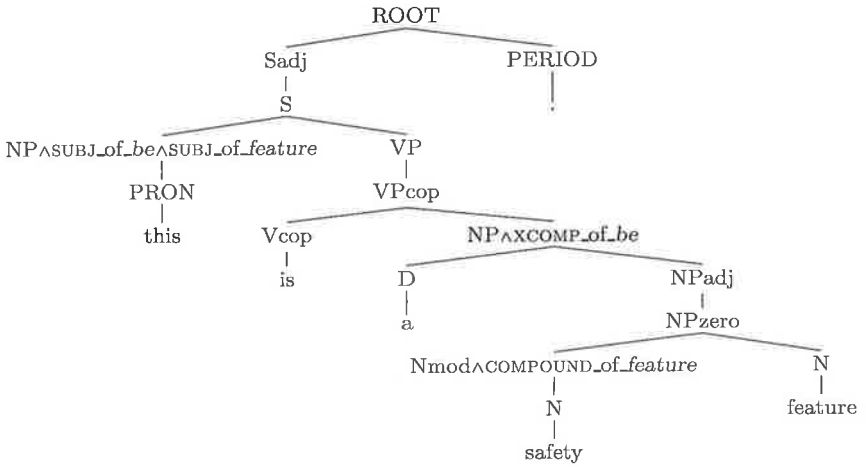


Figure 4.8: A c-structure annotated with some functional information on the topmost nodes of the appropriate constituents. It is possible for any node to fulfil more than one function; such a node receives an annotation for each function. An example of this can be seen in the leftmost NP; this node functions as the SUBJ of both *be* and *feature*, and so is annotated with both functions.

constituent which fulfils this function.

Where a constituent fulfils more than one function in the sentence, we append a label for each function to the topmost node in the constituent which fulfils that function. Upon further examination of the f-structure, we see that the NP node also functions as the SUBJ of *feature*; this label becomes NP∧SUBJ_of_*be*∧SUBJ_of_*feature*. A c-structure annotated with functions can be seen in Figure 4.8.

4.3.2 Atomic Features

The second class is atomic features. These features have a small set of closed class items as possible values; for example the feature NUM can only ever have the value 1, 2 or 3. A single atomic feature may apply to more than one node; in this case, each applicable node receives the atomic annotation.

As described in section 4.3, features are divided into two groups: features which are annotated on the root node of the constituent, and features which are annotated on the preterminal nodes dominating the terminals to which they specifically apply.

Figure 4.9 shows the sentence previously illustrated in Figure 4.1 with atomic annotations for the features ANIM and NUM; in this example we can see that the nodes NP and Nmod are annotated with the feature ANIM. In a fragment where this NP is an open substitution site, we know that the NP-rooted fragment which is substituted at this point must include an inanimate (ANIM=-) concept.

The feature NUM is applied to preterminals. As can be seen in Figure 4.9, the node dominating the *this* is annotated PRON∧NUM=3. The f-structure for this sentence, given in Figure 4.1, also shows that *safety* and *feature* have NUM of 3 each. The preterminals to these nodes are also annotated.

Looking at the ϕ-linked f-structure for the sentence in Figure 4.1, we see that the outermost f-structure is linked to the ROOT node, which dominates all other nodes. If we consider the features which lie within this f-structure unit, but outside other inner units, it might appear that the feature PASSIVE should be annotated on all preterminal nodes, even to those which, logically, we know to be unrelated; for instance, we know that determiners, such as the terminal *a*, do not have a PASSIVE quality. However, this does not occur in the GF-DOP model; nodes which correspond to inner f-structure units are ϕ-linked to their respective f-structure units, rather than the outermost unit which dominates them. In the c-structure shown in Figure 4.1, only the Vcop node receives this annotation, as illustrated in Figure 4.9.

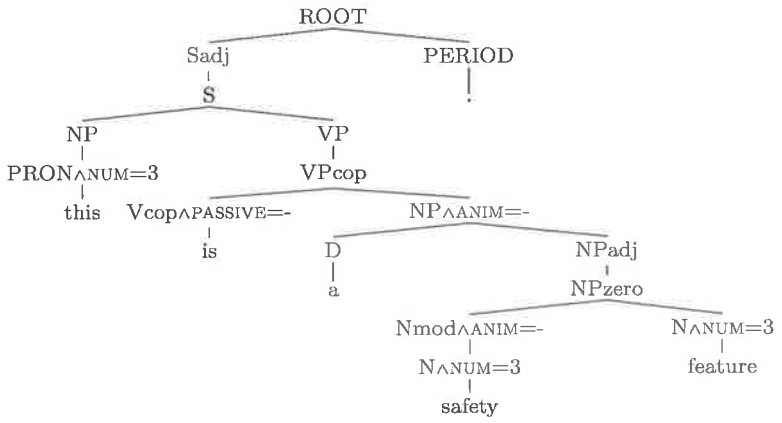


Figure 4.9: A c-structure with both root-based and preterminal-based atomic annotations. ANIM is placed on the root of the constituent which dominates all words in the animate concept. NUM is on preterminal nodes only.

Although the preterminal PERIOD is also dominated by this f-structure unit, and not ϕ-linked to any other unit, we do not annotate preterminals of punctuation.

4.3.3 Lexical Features

The third class of features mentioned is lexical features. These features have one of a small number of lemmas as their values; for example, CONJ-FORM can have one of *and*, *or*, *and-or*, *then*, *plus* or *null* as its value. As for atomic features, lexical annotations are divided into two groups: those which are placed on the constituent-root, and those which are placed on preterminals. Predicates are not included in this group; these will be considered separately in section 4.3.5. An example of a lexically annotated c-structure can be seen in Figure 4.10.The PRON-FORM is specified as *this*. The SPEC-FORM used with *feature* is specified as a, which is also indicated on this c-structure.

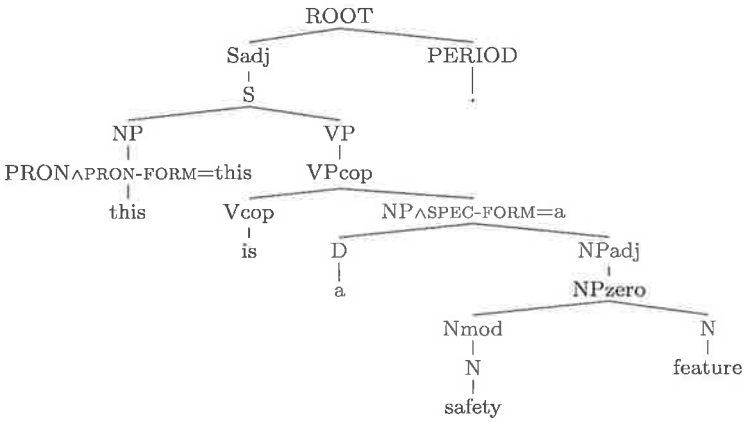


Figure 4.10: A c-structure with lexical annotations on preterminal nodes; PRON-FORM is specified as *this*. SPEC-FORM used with *feature* is specified as a. The singular form specifier a influences the form of the noun 'feature'.

As can be seen in this example, where a feature applies to one specific node, the annotation is placed on the preterminal dominating the terminal, (PRON-FORM=this). Where a feature is ϕ-linked to several nodes, as can be seen in the f-structure in Figure 4.1 (SPEC-FORM=a), it is placed on the highest node which dominates those constituents only .

4.3.4 Non-Grammatical Function Features

The fourth class of are non-grammatical function features. These features have a set of atomic features as their value. Intuitively, it is more useful to annotate the node with the contents of the feature's f-structure value: that is, rather than identifying that a node has, for example, tense and aspect, denoted by the feature TNS-ASP, (VCOP∧TNS-ASP), we annotate it with the features which define the tense and aspect: VCOP∧MOOD=indicative∧PERF=-∧PROG=-∧TENSE=pres. These features are added in the same manner as atomic features, as described in section 4.3.2. An example of these annotations can be seen in Figure 4.11.

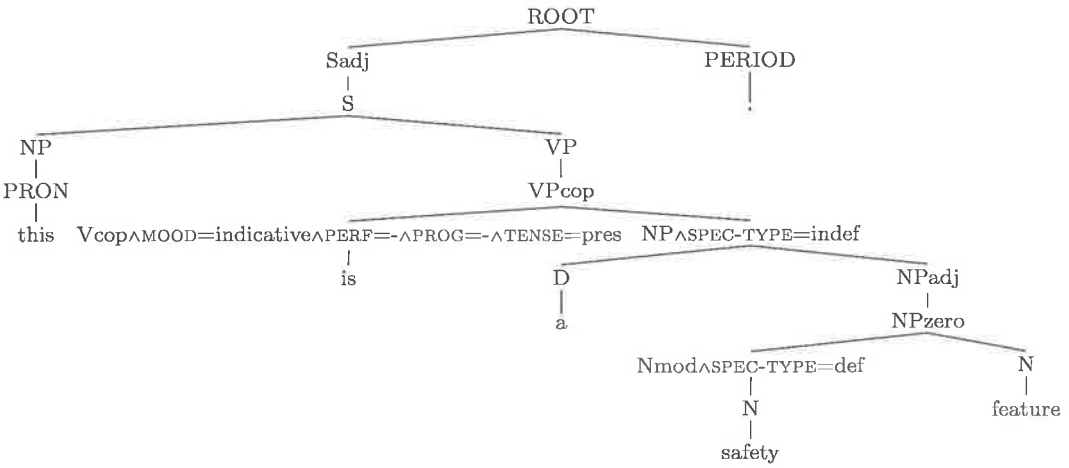


Figure 4.11: A c-structure with super-feature atomic annotations on the nodes preceding the relevant terminal word.

The features which describe the tense and aspect are annotated on the VCOP node, rather than on the VP or VPCOP nodes. If these features were added to the VP or VPCOP nodes, there would be the implication that the other nodes dominated by VP or VPCOP carry these precise tense and aspect features also.

4.3.5 Predicates

A final, single feature class contains the PRED feature. This feature has a lemma as its value, but it differs from the lexical features described in section 4.3.3 because lexical features can have only a small number of lemmas, essentially a closed class set, as their values, while PRED can have any word as its value. The PRED feature

may also have subcategorisation arguments; this is a list of arguments which are required by a verb or other predicate. For example, one possible subcategorisation frame for the verb *to eat* as a PRED feature in an f-structure might be written PRED 'eat<SUBJ, OBJ>'; in this context, the predicate *eat* requires a SUBJ and an OBJ in order to satisfy its subcategorisation requirements, and fulfill the LFG well-formedness conditions.

Let us consider the annotation possibilities for this feature. From the PRED we can establish the lexical word and the list of obligatory arguments. There is perhaps no great advantage in extracting the lexical word from the value, as this word features in the c-structure as a terminal anyway. However, the subcategorisation frame might be used to specify the context in which this word can appear. For example, if we encounter a sentence with the word *eat*, we might use the subcategorisation requirements to check that the sentence also has some node which is labelled SUBJ of *eat* and a node labelled OBJ of *eat*. However, using this type of annotation would require us to implement unification. The philosophy behind GF-DOP is to transform the treebank, rather than the parser. With this in mind, we propose another way to incorporate subcategorisation information.

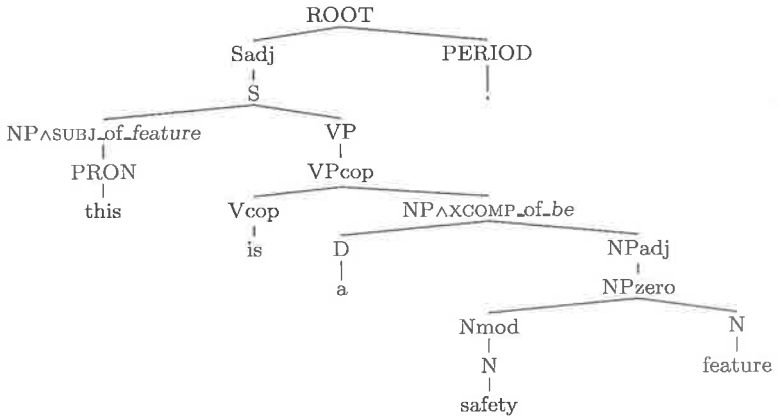


Figure 4.12: A c-structure annotated with subcategorised functions only.

From the subcategorisation frames, we can determine which functions are obligatory arguments of predicates; from this we can create a treebank annotated with these functions only. An example of this type of annotation can be seen in Figure

4.12. From the set of functions which appear in the f-structure, we aim to identify specific functions which are important in the given context.

4.4 Preserving Robustness

Data sparseness is a prominent issue in implementing Tree-DOP and is further exacerbated by the detailed node labels in GF-DOP. The GF-DOP model's use of additional feature information may mean it does not generate some parses which would be proposed by the Tree-DOP model. This reduction in coverage might be seen as a weakness in the GF-DOP model. To preserve robustness in the model, we incorporate a 'backing-off' technique in the GF-DOP model; after extracting the GF-DOP grammar from the annotated corpus, we extract a second grammar from a copy of the treebank with all annotations removed. In effect, we extract a Tree-DOP grammar. We assign the majority of the probability mass (W_1) to the GF-DOP grammar, and allocate a small amount of probability mass (W_2) to the Tree-DOP grammar, such that $W_1 + W_2 = 1$. These two grammars are merged and their probabilities smoothed. By merging the GF-DOP and Tree-DOP grammars, we ensure some valid parse is obtained for sentences which might not be parsed by the GF-DOP grammar alone. We maintain at least the same level of coverage and robustness as the Tree-DOP model.

We present here a practical illustration of the impact of feature annotations and back-off. As discussed in section 3.4.1, the Tree-DOP model is limited by the representations it assumes. The parses generated are grammatical with respect to the given corpus. Figure 4.13 (C) shows how fragments (Figure 4.13 (B)), derived from the treebank given in Figure 4.13 (A), may be combined to produce a parse for a sentence which is grammatical given the grammar, but would otherwise be considered ungrammatical; Tree-DOP has no way of modeling such ungrammaticalities. There are several possible derivations for this sentence, as can be seen in Figure 4.13 (C).

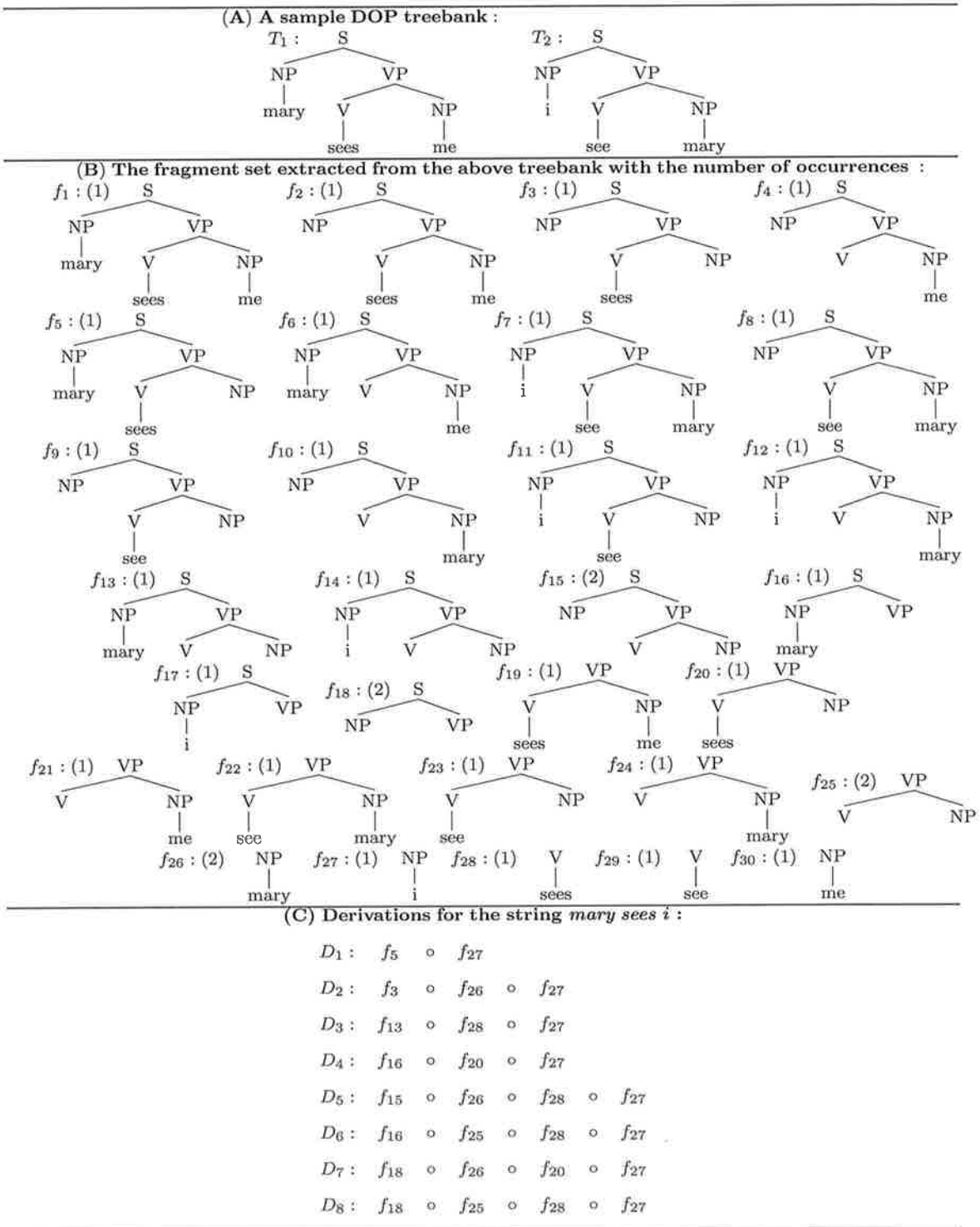


Figure 4.13: Illustration of the Tree-DOP model parse for the sentence *mary sees i*.

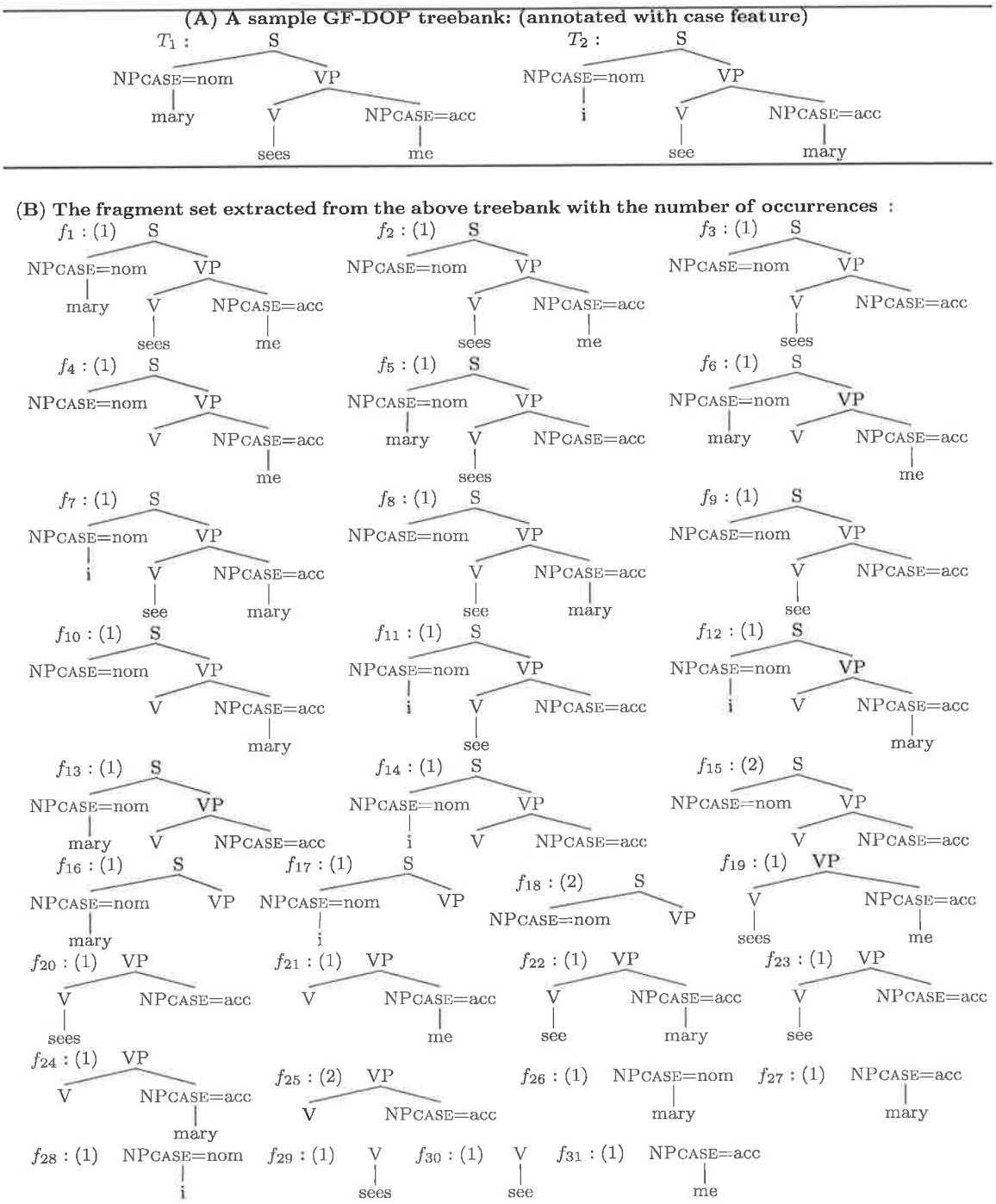


Figure 4.14: Illustration of the GF-DOP model parse for the sentence *mary sees i*. There are no valid derivations for this sentence given the GF-DOP grammar extracted from the corpus. *i* is never annotated CASE=acc, and can never appear in OBJ position in this example.

Through probabilistic weighting and annotation of the corpus with features, GF-DOP is less likely to produce ungrammatical parses than Tree-DOP. Figure 4.14 illustrates how GF-DOP excludes the possibility of generating a parse for the sentence *mary sees i* given the treebank in Figure 4.14 (A). The fragment set which can be extracted is shown in Figure 4.14 (B). Before annotation (Figure 4.13 (B)), we can see that there are two fragments of the form $NP \rightarrow mary$; after annotation (Figure 4.14 (B)), we have two distinct fragments: in the first instance, fragment f_{26}, the *NP* is annotated CASE=nom. In the second instance, fragment f_{27}, the *NP* is annotated CASE=acc. Thus $NP \rightarrow mary$ can compose with a leftmost open substitution site with label *NP*, appearing in a nominative or accusative role (by selecting the appropriately annotated fragment). However, there is only one fragment of the form $NP \rightarrow i$, fragment f_{28}; this fragment is annotated CASE=nom. In a parse generated from the treebank given in this example, *i* can only ever appear in a nominative role. Therefore, no valid parses are possible for the sentence *mary sees i* given the extracted grammar.

In this example, the GF-DOP model exhibits reduced robustness. However, backing-off is a fundamental element of the model and so the fragment set available to the GF-DOP model is in fact the union of the fragments in Figure 4.13 (B) and Figure 4.14 (B). As a result, some parse will be generated for the input sentence *mary sees i*, although it is most probable that fragments involved would have originally been extracted from the Tree-DOP fragment set, shown in Figure 4.13 (B).

4.5 How does GF-DOP improve on Tree-DOP?

The prevailing advantage of GF-DOP over Tree-DOP is that GF-DOP has the capacity to generate more informative parses than Tree-DOP alone. In particular, the use of functional annotations in GF-DOP provides a considerable amount of detail regarding the relationships between constituents. Let us compare the parses which

would be generated for the sentence *i see me*[2] given the treebanks in Figure 4.13 (A) and Figure 4.14 (A), as illustrated in Figure 4.15; parse (A) has been constructed from Tree-DOP fragments, extracted from the treebank in Figure 4.13 (A), while parse (B) has been constructed from GF-DOP fragments, extracted from the treebank in Figure 4.14 (A). Although both parses have the same internal structure, parse (B) provides us with additional grammatical information. We can identify the CASE of *i* and *me*; NPCASE=nom $\longrightarrow$ *i* identifies *i* as a nominative NP. This can only be composed with an open substitution site NP in a nominative position. NPCASE=acc $\longrightarrow$ *me* identifies *me* as an accusative NP. This can only be composed with an open substitution site NP in an accusative position.

Parse (B) is generated from GF-DOP fragments. As this model is trained on data with a greater degree of linguistic detail than the Tree-DOP model, the parses generated contain more detailed information than those generated by the Tree-DOP model, such as parse (A) in this example. This level of annotation may provide us information as to which constituents are likely to function as the subject or object of this sentence. Tree-DOP provides none of this detail.

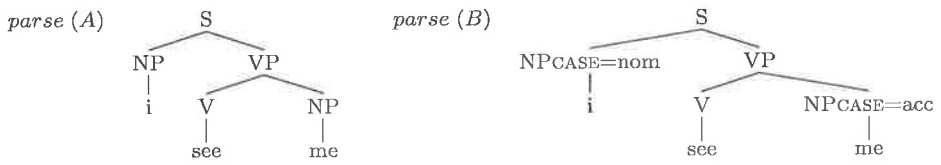


Figure 4.15: Parse (A) shows a parse for the sentence *i see me*, constructed from the Tree-DOP fragments in Figure 4.13 (B). Parse (B) shows a parse for the sentence *i see me*, constructed from the GF-DOP fragments in Figure 4.14 (B).

[2]Although this sentence is ungrammatical, it may be considered grammatical with respect to the given treebank; the GF-DOP parser must still be able to deal with such examples of language. We have chosen this sentence to emphasise the point that the GF-DOP model can correctly place nominative forms in subject position and accusative forms in object position.

4.6 How does GF-DOP improve on LFG-DOP?

LFG-DOP's strength comes from the representations assumed by its fragments. The unification of features ensures well-formed grammatical parses are generated. However, not all LFG-DOP derivations unify globally, or they may fail to meet (one or more of) the three well-formedness conditions, defined in section 3.1.1, which are required to produce a valid parse. Because we exclude these ill-formed derivations as they are encountered, we lose probability mass; the probability distribution of derivations does not correspond to the probability model. As a result of this, there is currently no satisfactory realisation of the LFG-DOP model.

The GF-DOP model extracts information from LFG f-structure representations, and appends this information to the c-structure category labels. Category-matching is the only constraint which is required to be enforced during composition; as a result, only valid derivations are constructed. In this way, we make use of available feature and functional information, while avoiding the probabilistic difficulties which arise due to the generation of invalid parses.

Parallels may be drawn between the 'backing-off' technique employed by the GF-DOP model and the 'discounted relative frequency' technique in LFG-DOP (Bod and Kaplan, 2003). The number of discard-generated fragments generated in LFG-DOP is exponential compared to the number of root- and frontier-generated fragments. In order to ensure that the probability model exhibits a preference for the more specific representations (root- and frontier-generated fragments), the 'discounted relative frequency' technique is applied. This approach separates root- and frontier-generated fragments and discard-generated fragments into two separate bags. The root- and frontier-generated bag is treated as a bag of "seen" events, while the discard-generated bag is treated as a bag of "unseen" events. The total probability mass (W) is divided; a small amount of probability mass (W_1) is assigned to the "unseen" events, the remainder (W_2) is assigned to the "seen" events, such that $W_1 + W_2 = W = 1$. The probability of each fragment is calculated as its relative

frequency in the bag multiplied by the probability mass assigned to that bag.

Similarly, the GF-DOP model assigns the majority of the total probability mass (W_3) to annotated fragments, and a small amount (W_4) to back-off-generated fragments, such that $W_3 + W_4 = 1$. By assigning only a small amount of probability mass to Tree-DOP and discard-generated fragments ("unseen" events), we promote the use of GF-DOP and root- and frontier-generated fragments as our "first choice" fragments. Probabilistic weighting encourages the use of back-off- or discard-generated fragments only where no other fragments are possible.

However, the inclusion of discard-generated fragments in LFG-DOP increases the size of the treebank exponentially compared to that of the Tree-DOP model; all possible combinations of attribute-value pair deletions are applied. The result is computationally very expensive. As the GF-DOP model deletes all feature annotations at once, only one backed-off fragment is created per GF-DOP fragment. The backed-off GF-DOP model generates at most double the number of fragments of the Tree-DOP model. The resulting model is less powerful than the LFG-DOP model, but computationally much more manageable.

The GF-DOP model combines the robustness of the DOP model with some of the linguistic competence of LFG and can be seen as an approximation towards LFG-DOP. Through use of the discard operator, LFG-DOP can generate a parse for input, whether it is well- or ill-formed with respect to the corpus, incorporating considerable real-world linguistic detail. In GF-DOP, we aim to model as much of this linguistic detail as possible, with the objective of approximating LFG-DOP, without adversely affecting the coverage of the grammar extracted from the annotated treebank. Any loss in coverage would equate to losing some of the robustness which is characteristic of the Tree-DOP model.

4.7 The GF-DOP Hypothesis

Having described the GF-DOP model, and highlighted its principal strengths, we consider what the implications of this new model might be. We hypothesise that through the incorporation of grammatical annotations, the GF-DOP model can accurately learn grammatical features, and apply this acquired knowledge to better model language, producing more accurate, and more informative, phrase-structure trees than the Tree-DOP model.

4.8 Summary

In this chapter we have presented in detail the new model we propose: the GF-DOP model. We discuss several approaches considered for annotating the treebank which is exploited by the model. We classified features found in the treebank, and present examples of annotations from each class. After presenting how we maintain robustness in the new GF-DOP model, we related GF-DOP to the Tree-DOP and LFG-DOP models, comparing the strengths and weaknesses of each. We hypothesize what we hope the GF-DOP model will achieve experimentally and, finally, suggest some future avenues for expansion of the model.

Chapter 5

Experimental Set Up

The GF-DOP hypothesis states that through the incorporation of functions and feature annotations, the GF-DOP model can produce more accurate phrase-structure trees than the Tree-DOP model. The GF-DOP model should also enable us to learn grammatical features accurately. In this chapter, we outline the experimental set up which will be used to investigate the actual performance of the new model. We present the data set and give a breakdown of features found in the data, and used in the experiments. We describe our treebank preparation before giving an overview of the parser used. Finally, we describe the evaluation metrics employed. Examination of the new model with respect to English and French, both individually and contrastively, is reserved for discussion in Chapters 6, 7 and 8. Figure 5.1 illustrates the relationship between these four chapters, with the current chapter highlighted.

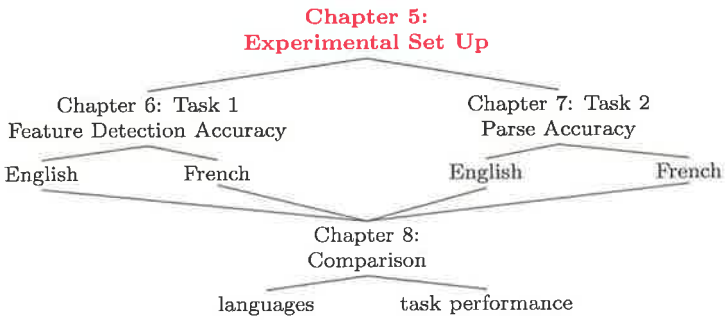


Figure 5.1: Illustration of relationships between Chapters 5, 6, 7 and 8.

5.1 The Data Set

The corpus used in the experiments presented in this chapter is the Xerox HomeCentre corpus. This corpus comprises 980 sentences in English and their translation into 930 sentences in French (several n-to-m translations occur). There are an average of 8.54 words per sentence in the English side of the corpus, and an average of 9.87 words per sentence for the French side. Each sentence is annotated with a c-structure representation and its corresponding ϕ-linked f-structure. The corpus was created from an instruction manual for a home printer. Each sentence was automatically parsed at Xerox Parc using their XLE grammars,[1] and the 'best' annotation was manually selected.

Functions	Atomic Features		Lexical Features	Non-Grammatical Functions
ADJUNCT	ABBREV	NUM	COMP-FORM	ARG-EXT
APP	ACONSTR	NUMBER-TYPE	CONJ-FORM	ARGS-INT
COMP	ADEG-DIM	PASSIVE	CONJ-FORM-COMP	ASPECT
COMP-EX	ADEGREE	PERF	PCASE	DEP
COMPOUND	ADJUNCT-TYPE	PERS	PRECONJ-FORM	NON-DEP
OBJ	ADV-TYPE	POL	PREDET-FORM	NTYPE
OBJ2	ANIM	PREDET-TYPE	PRON-FORM	PREDET
OBL	ATYPE	PROG	PRT-FORM	SPEC
OBL-AGT	AUX-FORM	PRON-TYPE	SPEC-FORM	TNS-ASP
OBL-COMP	CASE	PROPER		
PRON-INT	DEIXIS	PSEM		
PRON-REL	EMPH	PTYPE		
SPEC	EMPHASIS	SPEC-TYPE		
SUBJ	FIN	STMT-TYPE		
TOPIC-INT	GEND	TEMPORAL		
TOPIC-REL	GERUND	TENSE		
XCOMP	GRAIN	TIME		
	INF	TYPE		
	LAYOUT-TYPE	VFORM		
	MOOD	VTYPE		
	NEG-FORM			

Table 5.1: ENGLISH: Classification of 76 features identified in the English section of the data set.

[1] http://www2.parc.com/isl/groups/nltt/xle/

5.1.1 English Features

In the English side of this data set, 76 features (excluding PRED) were identified. These features were divided into four classes, as described in section 4.2. The classification of these features can be seen in Table 5.1. As stated in section 4.3.4, we do not annotate trees with non-grammatical function features directly, but rather use the features listed within their f-structure values. These features are grouped under the atomic and lexical feature categories. In addition to non-grammatical function features, there are five other features we do not use:

- AUX-FORM: although this feature is a form like most of the lexical features, only one value is possible: *contracted*. This feature is used to indicate that an auxiliary form is contracted, for example *here's* rather than *here is*, or *you're* instead of *you are*. This feature occurs only 11 times in the data set. We manually 'cleaned up' the corpus by removing all contracted forms from the c-structures, so this f-structure feature is no longer relevant. In addition, this step helps slightly counteract the effect of data sparseness.

- NEG-FORM: like AUX-FORM, NEG-FORM has *contracted* as its only value. This feature works in the same way as AUX-FORM: it indicates that a negative form has been contracted, for example *doesn't* rather than *does not*, or *don't* in place of *do not*. This feature occurs only 14 times in the data set. We removed occurrences of contracted negative forms from the c-structures, making this f-structure feature redundant, and again modestly reducing the effects of data sparseness.

- VFORM: despite this feature being called a form, it appears to behave more like an atomic feature in that it has a small set of non-lexical values: *presp*, *base*, *passp* and *perfp*. Upon examination of the corpus, we found that this feature was contained in f-structure units which were neither linked to the main f-structure unit, nor to any c-structure nodes. As this feature is not connected to c-structure nodes either directly, via ϕ-links, or indirectly, through another

f-structure unit which is ϕ-linked to some c-structure node, we do not generate a treebank annotated with this feature. Any such treebank would essentially be the same as the baseline (original, unannotated) treebank.

- FIN: this atomic feature occurs in f-structure units which are not linked to the main f-structure, and are not linked to any c-structure nodes. Thus we do not generate a treebank annotated with this feature.

- INF: this atomic feature occurs in the same situations as FIN: in f-structure units which are not linked to the main f-structure, or linked to c-structure units. We do not generate a treebank annotated with this feature.

We clarify how a feature might be present in an f-structure but not linked to the main f-structure unit with the illustration in Figure 5.2. For each pair of linked c- and f-structures used in the experiments carried out, we have textual representations. Figure 5.2 shows a section of the textual representation of an f-structure. The unit shown is referred to as "%40". It contains three features, ARG-EXT, DEP and FIN. The value of the attribute ARG-EXT is the f-structure unit labelled "%41" and the value of DEP is the f-structure unit labelled "%42". These units (%41 and %42, not shown here) are nested one level deeper than unit %40. Any attribute which has %40 as its value will have the f-structure unit shown as its value. However in our data set, the f-structure units containing FIN (and also those containing INF) are never the value of any attribute. Neither are these units ϕ-linked to any node in the corresponding c-structure. As a result, we say that these features are not linked to the main f-structure, or to any c-structure nodes.

```
( %40   ARG-EXT )  =  %41
( %40     DEP   )  =  %42
( %40     FIN   )  =   +
```

Figure 5.2: Illustration of the textual representation of an f-structure.

Upon generating our initial treebanks, we excluded four further features: ACONSTR, EMPH, EMPHASIS and PRECONJ-FORM. We made this decision as each of these features resulted in only one or two annotations in the entire treebank, resulting in very little information for the parser to use to learn appropriate feature environments.

Thus the number of features we use when generating treebanks is reduced to 58; these features are listed and classified in Table 5.2. A treebank is generated for each of the features listed, a single feature annotated on each treebank. In addition to these singly-annotated treebanks, we generate several treebanks annotated with combinations of features. We generate eight multi-feature treebanks, using the most frequently occurring features:

- a treebank annotated with all grammatical functions (as listed in the leftmost column in Table 5.2)
- a treebank annotated with the five most frequently occurring grammatical functions in the data (ADJUNCT, OBJ, SUBJ, COMPOUND and XCOMP)
- a treebank annotated with the functions SUBJ and OBJ only.
- a treebank annotated with the atomic features NUM and PERS
- a treebank annotated with the atomic features PERF, PROG and TENSE
- a treebank annotated with the atomic features PERF, PROG, TENSE, PASSIVE and MOOD
- a treebank annotated with the combination of atomic and lexical features PREDET-TYPE and PREDET-FORM
- a treebank annotated with the combination of atomic and lexical features SPEC-TYPE and SPEC-FORM.

Functions	Atomic Features		Lexical Features
ADJUNCT	ABBREV	PERF	COMP-FORM
APP	ADEG-DIM	PERS	CONJ-FORM
COMP	ADEGREE	POL	CONJ-FORM-COMP
COMP-EX	ADJUNCT-TYPE	PREDET-TYPE	PCASE
COMPOUND	ADV-TYPE	PROG	PREDET-FORM
OBJ	ANIM	PRON-TYPE	PRON-FORM
OBJ2	ATYPE	PROPER	PRT-FORM
OBL	CASE	PSEM	SPEC-FORM
OBL-AGT	DEIXIS	PTYPE	
OBL-COMP	GEND	SPEC-TYPE	
PRON-INT	GERUND	STMT-TYPE	
PRON-REL	GRAIN	TEMPORAL	
SPEC	LAYOUT-TYPE	TENSE	
SUBJ	MOOD	TIME	
TOPIC-INT	NUM	TYPE	
TOPIC-REL	NUMBER-TYPE	VTYPE	
XCOMP	PASSIVE		

Table 5.2: ENGLISH: List and classification of the 58 English features for which we generated singly-annotated corpora.

5.1.2 French Features

In the French side of the data set 79 features were identified (excluding PRED). These features were divided into four classes, as described in section 4.2. The classification of these features can be seen in Table 5.3. As for English, we do not annotate with all 79 of these features; we exclude all non-grammatical function features. We initially classified APSEC as a non-grammatical function; however manual inspection of the data set shows that this feature occurs in one sentence only. Furthermore, it appears where we would normally have seen the feature ASPEC, so we conclude that this feature is an error. In addition to the named excluded features, we do not annotate with the following:

- VFORM: as for English, VFORM has more in common with atomic features than lexical features. As this feature is only present in f-structure units which are neither ϕ-linked to any other f-structure units nor any c-structure nodes, we do not generate a treebank for this feature. Any such treebank would essentially be the same as the baseline (original, unannotated) treebank.

Functions	Atomic Features		Lexical Features	Non-Grammatical Functions
ADJUNCT	ADEG-DIM	PASSIVE	AUX-SELECT	ACONSTR
COMP	ADEGREE	PCASE-TYPE	COMP-FORM	APSEC
COMPOUND	ADJUNCT-LAYOUT	PERF	COMP-FORM-ANAPH	ARG-EXT
OBJ	ADJUNCT-TYPE	PERS	CONJ-FORM	ARGS-INT
OBJ2	ADV-TYPE	PREDET-TYPE	CONJ-FORM-COMP	ASPEC
OBL	APOS	PREVERB-OBJ	FORM	DEP
OBL-AGT	ATYPE	PRON-TYPE	NEG-FORM	FIN
OBL-COMP	CASE	PROPER	PCASE	INF
PRON-REL	CONJOINED	PSEM	PRECONJ-FORM	NON-DEP
SPEC	CONJTYPE	PTYPE	PREDET-FORM	NTYPE
SUBJ	DEIXIS	REFL	PRON-FORM	PREDET
TOPIC-REL	FOO	SPEC-TYPE	SPEC-FORM	TNS-ASP
XCOMP	GRAIN	STATUS		VCONSTR
	GEND	STMT-TYPE		
	INV	STRESSED		
	LAYOUT-TYPE	TENSE		
	MOOD	TIME		
	NE	TYPE		
	NEG	VFORM		
	NUM	VTYPE		
	NUMBER-TYPE			

Table 5.3: FRENCH: Classification of 79 features identified in the French section of the data set.

- FIN: this atomic feature also occurs in f-structure units which are not linked to the main f-structure, and are not linked to any c-structure nodes. We do not generate a treebank annotated with this feature.

- INF: this atomic feature occurs in the same situations as FIN: in f-structure units which are not linked to the main f-structure, or linked to c-structure units. We do not generate a treebank annotated with this feature.

The number of features we actually use for annotation is reduced, in this case to 65; the features used can be seen in Table 5.4. We generate a treebank annotated with each of these 65 single features, and also several treebanks annotated with combinations of features:

- a treebank annotated with all grammatical functions (as listed in the leftmost column in Table 5.4)

- a treebank annotated with the five most frequently occurring grammatical functions in the French data (ADJUNCT, COMPOUND, OBJ, OBL and SUBJ)
- a treebank annotated with the five most frequently occurring grammatical functions in the English data (ADJUNCT, COMPOUND, OBJ, SUBJ and XCOMP)
- a treebank annotated with the functions SUBJ and OBJ only
- a treebank annotated with the atomic features NUM and PERS
- a treebank annotated with the atomic features NUM, PERS and GEND
- a treebank annotated with the atomic features PERF and TENSE
- a treebank annotated with the atomic features PERF, TENSE, PASSIVE and MOOD
- a treebank annotated with the combination of atomic and lexical features PREDET-TYPE and PREDET-FORM
- a treebank annotated with the combination of atomic and lexical features SPEC-TYPE and SPEC-FORM.

We have used some slightly different combinations of features in generating French treebanks. The five most frequently occurring features in the French data set are not the same as the five most commonly occurring features in the English data set. To facilitate comparison of performance later, we have generated two TOP5 treebanks: FRE5 which is annotated with ADJUNCT, COMPOUND, OBJ, OBL and SUBJ, and ENG5 which is annotated with ADJUNCT, COMPOUND, OBJ, SUBJ and XCOMP.

For both English and French we generated treebanks annotated with NUM and PERS. However, as GEND is particularly salient for French, and a word's surface form is frequently related to number, person and gender, we have also generated a treebank NUM_PERS_GEND.

As the feature PROG was not present in the French data set, we have generated treebanks PERF_TENSE and PERF_TENSE_PASSIVE_MOOD to correspond to the

Functions	Atomic Features		Lexical Features
ADJUNCT	ADEG-DIM	NUMBER-TYPE	AUX-SELECT
COMP	ADEGREE	PASSIVE	COMP-FORM
COMPOUND	ADJUNCT-LAYOUT	PCASE-TYPE	COMP-FORM-ANAPH
OBJ	ADJUNCT-TYPE	PERF	CONJ-FORM
OBJ2	ADV-TYPE	PERS	CONJ-FORM-COMP
OBL	APOS	PREDET-TYPE	FORM
OBL-AGT	ATYPE	PREVERB-OBJ	NEG-FORM
OBL-COMP	CASE	PRON-TYPE	PCASE
PRON-REL	CONJOINED	PROPER	PRECONJ-FORM
SPEC	CONJTYPE	PSEM	PREDET-FORM
SUBJ	DEIXIS	PTYPE	PRON-FORM
TOPIC-REL	FOO	REFL	SPEC-FORM
XCOMP	GRAIN	SPEC-TYPE	
	GEND	STATUS	
	INV	STMT-TYPE	
	LAYOUT-TYPE	STRESSED	
	MOOD	TENSE	
	NE	TIME	
	NEG	TYPE	
	NUM	VTYPE	

Table 5.4: FRENCH: List and classification of the 65 French features for which we generated singly-annotated corpora.

English treebanks PERF_PROG_TENSE and PERF_PROG_TENSE_PASSIVE_MOOD respectively.

5.2 Experimental Set-Up

As stated in sections 5.1.1 and 5.1.2, we identified 75 features in the English data set and 79 in the French. However, for reasons also outlined in sections 5.1.1 and 5.1.2, the number of features we actually annotated the treebanks with is reduced to 58 for English and 65 for French. For each feature identified we generated a copy of the original treebank; examination of f-structure units enables us to determine the presence or absence of the feature in question in each sentence, and we annotate the copy of the treebank as described in section 4.3. By tracking the number of annotations per treebank, we identify the most frequently and infrequently occurring features. Where a feature annotation occurs only once or twice in the entire

treebank, the data is too sparse for us to draw any informative conclusions. We cannot identify useful patterns, for example is this feature useful in improving the quality of the phrase-structure trees generated? Is there a correlation between the impact this feature has on the parser's performance and the feature's frequency of occurrence? How often is this feature accurately identified? In a treebank with multiple feature annotations, if this feature is not accurately identified, what alternative feature is proposed by the parser? For several English features, such as ACONSTR, EMPH, EMPHASIS and PRECONJ-FORM there were very low feature occurrences, and so we do not include these treebanks in our experiments.

Having determined which features will provide sufficient data to attempt to avoid the issue of feature-sparseness, we pause for a moment to consider other sources which might introduce this problem. Although the approach outlined in section 4.3 is our ideal function annotation method for the GF-DOP model, the experiments carried out here make use of a very limited data set. In ensuring a very fine-grained description of functional relationships in the treebanks, we may reintroduce feature sparseness; that is, the baseline parser coverage is not affected, but a high number of infrequently occurring features will hinder the parser's ability to 'learn' about these features and most likely result in low feature detection accuracy scores. To verify if this is indeed the case, we generate two further treebanks per function.

We will refer to the original treebank, generated according to the approach outlined in section 4.3 as a "Type 1 Lexicalised Duplicate Functions" treebank. The two additional treebanks will be generated in almost the same way as the Type 1 Lexicalised Duplicate Functions treebank, the only difference being in the annotation generation. When generating the Type 1 Lexicalised Duplicate Functions treebank, any node which fulfils a function (or number of functions) is annotated with the function and the lemma of the predicate whose function it serves. An example of such an annotation can be seen in (5.1).

$$NP \wedge \text{SUBJ}\_of\_be \wedge \text{SUBJ}\_of\_feature \tag{5.1}$$

A "Type 2 Duplicate Functions" treebank has some of the fine-grainedness of these annotations removed. The annotations in the Type 2 Duplicate Functions treebank will only reflect the functions fulfilled, but will not specify the related predicate. The adaptation of the Type 1 Lexicalised Duplicate Functions annotation in (5.1) to a Type 2 Duplicate Functions annotation can be seen in (5.2). This serves to boost the number of occurrences of identical functions in the treebank, thus assisting in reducing feature sparseness.

$$NP \wedge \text{SUBJ} \wedge \text{SUBJ} \tag{5.2}$$

A "Type 3 Minimal Functions" treebank has further simplified labels: no duplicate labels are permitted. Where a node fulfils the same function for more than one predicate, only one annotation is appended to the syntactic category label. This reduces the number of distinct node labels, increases feature occurrence counts and further reduces feature sparseness. The adaptation of the Type 1 Lexicalised Duplicate Functions annotation in (5.1) to a Type 3 Minimal Functions annotation can be seen in (5.3).

$$NP \wedge \text{SUBJ} \tag{5.3}$$

In a multi-feature annotated treebank, a node which fulfils two or more different functions is annotated with a single instance of each of the functions. An example of such an annotation can be seen in (5.4).

$$NP \wedge \text{SUBJ} \wedge \text{OBJ} \tag{5.4}$$

In the final preparatory step, we remove all unary branching structures from the treebanks. This is necessary because of our chosen chart parsing algorithm, the CKY algorithm described in section 2.2.3. In truncating the fragments, we have two options: first, we could concatenate the labels from the truncated nodes to form precise, long labels, which maintain the level of detail provided by the corpus.

Secondly, we could keep only one node label and discard the rest. We chose to take the second option as this will avoid contributing to the data sparseness issue any further. We elected to keep the bottom-most label, as this node is most closely related to the dominated subtree.

From each of the training treebanks we generated eight training sets and eight corresponding test and reference sets. These splits were generated at random such that every word in the test set occurs in the corresponding training set, thus avoiding the issue of unknown words. For the English experiments, each training set contained 890 training sentences, and 90 test sentences along with their gold standard reference trees. For the French experiments, each training set contained 840 training sentences, and 90 test and reference sentences.

For each of the features presented in Tables 5.2 and 5.4, and each of the multi-feature treebanks described in sections 5.1.1 and 5.1.2, an annotated treebank is created. The eight pre-established splits are applied. For each split, the training set is used to train the parser. The parser is tested on the test set and evaluated on the corresponding reference set. Scores are calculated for each split and averaged over the eight splits for each annotated treebank. The scores presented in Chapters 6 and 7 are thus averages of the scores for each of the eight splits.

5.3 Parser Details

5.3.1 Training

During training, the parser extracts two PCFG grammars from the annotated treebank: a GF-DOP (annotated) grammar and a Tree-DOP (unannotated) grammar. These grammars are weighted; 99% of the probability mass is assigned to the GF-DOP grammar, with 1% assigned to the Tree-DOP grammar. The relative frequency estimator, given in equation (2.1), is used to calculate all probabilities. The grammars are then merged and their probabilities smoothed. As we use the Goodman reduction implementation approach, we incorporate no pruning techniques. This

concludes the training process.

5.3.2 Parsing

The system creates a parse chart for each new input string. The parse chart is filled with Goodman reduction rules by means of the CKY algorithm, described in section 2.2.3. Upon completion of the chart, we calculate the n-most probable derivations, where n is equal to 2000, using the Viterbi algorithm, as described in section 2.2.3. From these n derivations, we determine the number of distinct parses and sum over their derivation probabilities to find the most probable parse.

5.4 Evaluation

Several different evaluation metrics were used to evaluate output parses. We count the constituents in the output parse, and also in the reference parse, where a constituent comprises a syntactic category label, a starting position, and a span. Their intersection indicates how many correct constituents are present in the output parse.

The first metric is precision, calculated according to equation (5.5). Of the total number of constituents in the output parse, how many are correct? That is, how many of these constituents occur in the reference parse?

$$precision = \frac{the\ number\ of\ \textbf{correct}\ constituents\ in\ the\ output\ parse}{the\ total\ number\ of\ constituents\ in\ the\ output\ parse} \tag{5.5}$$

The second metric is recall, calculated according to equation (5.6). Of the total number of constituents in the reference parse, how many occur in the output parse?

$$recall = \frac{the\ number\ of\ \textbf{correct}\ constituents\ in\ the\ output\ parse}{the\ total\ number\ of\ constituents\ in\ the\ reference\ parse} \tag{5.6}$$

Subsequently, we calculate the harmonic mean of these two scores, or the f-score, the formula for which is given in equation (5.7).

$$f-score = \frac{precision\ *\ recall\ *\ 2}{precision\ +\ recall} \tag{5.7}$$

The output from each experiment is transformed in three different ways and precision, recall and f-score are calculated. The purpose of these transformations is to determine how effectively the GF-DOP model performs on three different tasks.

The first treebank transformation replaces all non-terminal node labels with the generic '.' label. An example of this transformation can be seen in Figure 5.3. This transformation enables us to calculate the unlabelled precision, recall and f-score for the output parses. Unlabelled scores show how well the parser chunks the sentence, or how well it determines the constituent hierarchy (ignoring constituent labels).

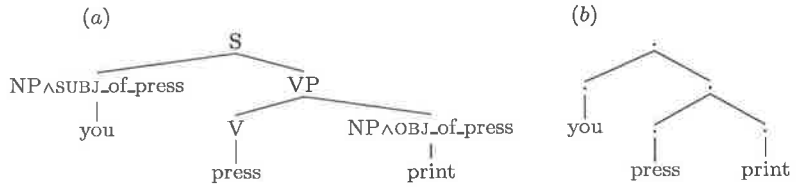


Figure 5.3: Transformation One: tree (a) shows the output from the parser, tree (b) shows the transformed output, with all non-terminal node labels replaced with the generic '.' label.

The second transformation strips all node labels of feature annotations, with only syntactic category labels remaining. An example of this transformation can be seen in Figure 5.4. From this transformation, we calculate labelled precision, recall and f-score for output parses. These scores illustrate the parser's phrase-structure tree accuracy.

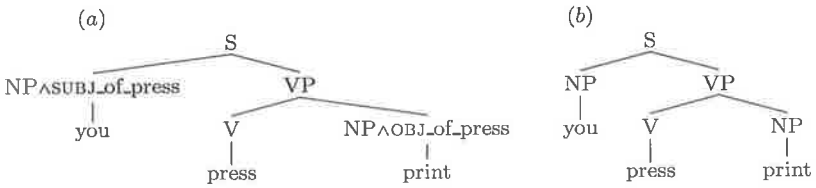


Figure 5.4: Transformation Two: tree (a) shows the output from the parser, tree (b) shows the transformed output, with all features removed from node labels.

The third transformation strips all syntactic node labels, with only annotated features remaining. An example of this transformation can be seen in Figure 5.5.

This transformation allows us to evaluate feature annotation accuracy.

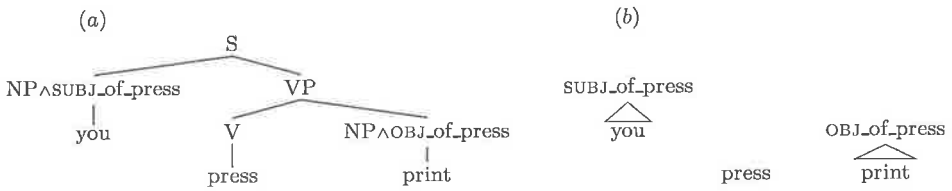


Figure 5.5: Transformation Three: tree (a) shows the output from the parser, tree (b) shows the transformed output, with all syntactic category labels removed, only feature labels remain.

5.5 Summary

In this chapter, we have given a detailed account of the experimental set up we intend to use to verify the GF-DOP hypothesis. We began by presenting the bilingual corpus used, classifying the features present in the data set and giving an account of the features to be included or excluded in our experimental investigation. Subsequently, we presented the parser used and evaluation metrics which we will apply.

Having put forward our experimental set up, the results of our experiments with the GF-DOP model are presented in the next two chapters. Given that the GF-DOP hypothesis comprises two assertions, we divide the results of our experiments into two sections: we examine feature detection accuracy (for both English and French) in Chapter 6, then present parse accuracy (again for both English and French) in Chapter 7. We consider the performance of the GF-DOP model in both languages before comparing our approach to related work on feature detection and parse accuracy in Chapter 8.

Chapter 6

Task 1 Results and Discussion: Feature Detection Accuracy

This chapter examines the GF-DOP model's performance at feature detection accuracy. As illustrated in Figure 6.1, we present results first for the English experiments carried out, and then for the French.

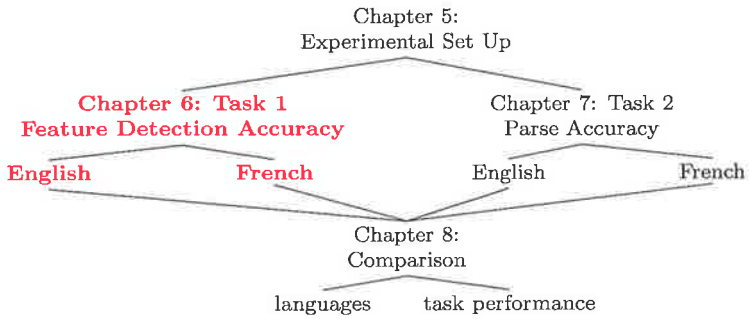


Figure 6.1: Illustration of relationships between Chapters 5, 6, 7 and 8.

Further to the language division shown in Figure 6.1, and the feature classification described in section 4.2, we group our experiments as illustrated in Figure 6.2. The three main feature categories are functions, atomics and lexicals. Each of these comprises several smaller subdivisions; function features are annotated in three ways, as described in section 5.2, resulting in Type 1 Lexicalised Duplicate Functions treebanks, Type 2 Duplicate Functions treebanks and Type 3 Minimal

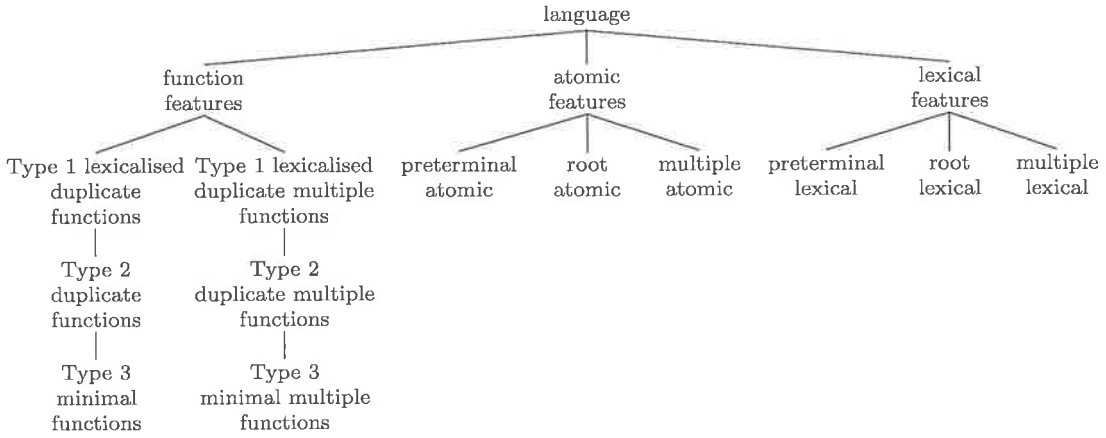


Figure 6.2: Illustration of feature subdivisions.

Functions treebanks. For each of these three treebank types, we also generated treebanks annotated with combinations of features: Type 1 Lexicalised Duplicate Multiple Functions, Type 2 Duplicate Multiple Functions and Type 3 Minimal Multiple Functions. As described in section 4.3, atomic and lexical features are subdivided into those features which are annotated at the root of the dominated constituents, and those which are annotated on preterminal nodes. Consequently, the atomic and lexical categories comprise preterminal-annotated features, root annotated features and their combinations. The result is twelve subdivisions of features, as illustrated in Figure 6.2; the same subdivisions are seen in both English and French.

For each table of results presented in this chapter, the first column names the feature-annotated treebanks being investigated. The second column, marked features, give the results for feature detection accuracy. Precision, recall and f-scores are given; however we concentrate our analysis on f-scores. The third column (occ) indicates the number of feature annotations present in the reference set (the total number of feature annotations we aim to identify across the 8 test splits). No BASELINE scores are presented as measuring feature detection accuracy for this treebank is inappropriate; no features are present in the test data, and no features are found, so the score would be 100%.

Complete score charts for each of these experiments are given in the appendix

(sections A and B) for completeness, but here we focus on those subsets of the results which illustrate the most important points; we show here only those features which have one or more feature occurrences in the testing data.

We begin with English treebanks annotated with a single function, before proceeding to multi-function annotated treebanks, treebanks with atomic annotations, and finally treebanks with lexical annotations. We then examine the same groups of features for French.

One factor which we expect to influence the feature detection accuracy scores is the frequency of occurrences of features in the training data. Where we can train the parser on a treebank with a very high number of features, we expect the parser to learn these features well; that is, the parser should identify appropriate environments for certain features, and be able to apply this pattern accurately. Where there are very few features present in the training data, or a very high number of distinct features, we expect feature sparseness to impair the parser's pattern detection and ultimately to impinge upon the parser's performance. The distribution of features in the reference sets corresponds roughly to the distribution of features in the treebank; from treebanks with few features we generate reference sets with few features, and the opposite is true for treebanks with high feature occurrences.

6.1 English: Feature Detection Accuracy

6.1.1 Functional Annotations

Type 1 Lexicalised Duplicate Function Annotations

A selection of the scores for Type 1 single function experiments are shown in Table 6.1. We show here only those features which have one or more features in the test set. Scores which are not displayed here, and which had no features to identify in their test sets, are APP, COMP-EX, OBJ2 and OBL-AGT. These features occurred very rarely in the original annotated treebanks (4, 3, 3 and 2 occurrences respectively).

	features			occ
	precision	recall	f-score	#
ADJUNCT	58.1465	46.6427	51.7631	834
COMP	77.2727	77.2727	77.2727	22
COMPOUND	78.6822	66.3399	71.9858	306
OBJ	63.4615	59.5361	61.4362	776
OBL	75.0000	56.2500	64.2857	16
OBL-COMP	100.0000	100.0000	100.0000	3
PRON-INT	100.0000	91.6667	95.6522	24
PRON-REL	100.0000	75.0000	85.7143	4
SPEC	85.5072	83.0986	84.2857	71
SUBJ	55.8480	45.6938	50.2632	418
TOPIC-INT	100.0000	100.0000	100.0000	25
TOPIC-REL	100.0000	66.6667	80.0000	3
XCOMP	75.0000	58.5366	65.7534	123

Table 6.1: Selected feature detection scores for Type 1 Lexicalised Duplicate Function annotated treebanks.

F-scores for Type 1 feature annotations range from 50.2632 (SUBJ) to 100 (OBL-COMP, TOPIC-INT). Scores of 100 were achieved only by features which had a very low occurrence count (OBL-COMP 3, TOPIC-INT 25); where there are not many features to be correctly identified, it is easier to get them all right. If we narrow our focus to only those features which have an occurrence count of 100 or more, this range is narrowed from 50.2632 to 71.9858. Only the 5 most frequently occurring features fall into this range; ADJUNCT, COMPOUND, OBJ, SUBJ and XCOMP. Of these, the highest feature accuracy score is achieved by COMPOUND, and the lowest by SUBJ. An interesting correlation may be noted between the f-scores for these 5 features and their occurrence counts. The features with the highest number of occurrences seem to achieve lower feature accuracy scores. We list here these features and their occurrence counts in decreasing order according to their f-scores:

- COMPOUND (306) f-score = 71.9895
- XCOMP (123) f-score = 65.7534
- OBJ (776) f-score = 61.4362
- ADJUNCT (834) f-score = 51.7631

- SUBJ (418) f-score = 50.2632.

However, if we consider how many features were actually correctly identified, we see an altered picture. Now we list these features and the number of correctly identified features in decreasing order according to the number of correctly identified features:

- OBJ features correctly identified = 477
- ADJUNCT features correctly identified = 432
- COMPOUND features correctly identified = 220
- SUBJ features correctly identified = 210
- XCOMP features correctly identified = 81.

Although OBJ and ADJUNCT have comparatively low f-scores, we see that they manage to correctly identify over 430 features each, approximately twice the number of features identified by the best performing feature COMPOUND. At this stage, we point out that Type 1 feature annotated treebanks tend to comprise a high number of different node labels with very low frequencies; we have a high number of distinct types relative to the number of tokens present, leading to feature sparseness. A reduction in feature sparseness may paint a considerably different picture.

Type 2 Duplicate Function Annotations

A selection of the scores for Type 2 single function experiments are shown in Table 6.2. As for Type 1 features, we show here only those features which have one or more features in the test set. F-scores for Type 2 feature annotations range from 64.2857 (OBL) to 100 (OBL-COMP, TOPIC-INT). Scores of 100 were achieved by the same features as for Type 1. SUBJ was previously the lowest scoring feature, but it has seen a large increase for Type 2 features and OBL is now the lowest scoring feature. This feature has a very low number of occurrences (16) and showed no improvement between Types 1 and 2.

	features			occ
	precision	recall	f-score	#
ADJUNCT	87.8173	82.9736	85.3268	834
COMP	77.2727	77.2727	77.2727	22
COMPOUND	87.2852	81.6720	84.3854	311
OBJ	88.4058	86.4691	87.4267	776
OBL	75.0000	56.2500	64.2857	16
OBL-COMP	100.0000	100.0000	100.0000	3
PRON-INT	100.0000	91.6667	95.6522	24
PRON-REL	100.0000	75.0000	85.7143	4
SPEC	100.0000	97.1831	98.5714	71
SUBJ	85.1175	77.2512	80.9938	422
TOPIC-INT	100.0000	100.0000	100.0000	25
TOPIC-REL	100.0000	66.6667	80.0000	3
XCOMP	77.5510	61.7886	68.7783	123

Table 6.2: Selected feature detection scores for Type 2 Duplicate Function annotated treebanks.

Narrowing our focus to features which have an occurrence count above 100, the range of f-scores is reduced to 68.7783 to 87.4267. Again, only the 5 most frequently occurring features fall into this range. This time the highest feature accuracy score is achieved by OBJ, and the lowest by XCOMP. We see that the reduction in feature sparseness has greatly boosted scores for some frequent features; ADJUNCT achieved an actual increase of over 33.5%, SUBJ has seen an actual increase of over 30% and OBJ shows an actual increase of almost 26%. However, there has been a much smaller impact on features which are less frequent; compound achieved an actual increase of 12.39% while XCOMP has seen an actual increase of only 3%. We see that the decrease in feature sparseness has had a hugely positive impact on the parser's ability to correctly identify features, particularly features with a high number of occurrences. The parser has clearly been better able to learn these features.

Type 3 Minimal Function Annotations

A selection of the scores for Type 3 single function experiments are shown in Table 6.3. As for Type 1 features, we show here only features which have one or more features in the test set. F-scores for Type 3 feature annotations range from 64.2857 (OBL) to 100 (OBL-COMP, TOPIC-INT). Scores of 100 were achieved by the same

features as for Type 1. Again, OBL is the lowest scoring feature, having shown no improvement over the previous two annotation types. Focusing again on features which have an occurrence count above 100, the range of f-scores is reduced to 68.7783 to 87.4267, exactly the same range as for Type 2 and the same 5 most frequently occurring features. We note that the exact same range is due to the fact that the highest and lowest scoring features have the same number of features for both Types 2 and 3. Changes between Type 2 and 3 scores are usually the result of a small reduction in the feature occurrences.

	features			occ
	precision	recall	f-score	#
ADJUNCT	88.5714	85.0374	86.7684	802
COMP	77.2727	77.2727	77.2727	22
COMPOUND	90.3475	85.4015	87.8049	274
OBJ	88.4058	86.4691	87.4267	776
OBL	75.0000	56.2500	64.2857	16
OBL-COMP	100.0000	100.0000	100.0000	3
PRON-INT	100.0000	91.6667	95.6522	24
PRON-REL	100.0000	75.0000	85.7143	4
SPEC	100.0000	97.1831	98.5714	71
SUBJ	87.1711	80.0604	83.4646	331
TOPIC-INT	100.0000	100.0000	100.0000	25
TOPIC-REL	100.0000	66.6667	80.0000	3
XCOMP	77.5510	61.7886	68.7783	123

Table 6.3: Selected feature detection scores for Type 3 Minimal Function annotated treebanks.

The largest Type 3 increase in any f-score over any Type 2 figure was achieved by COMPOUND, with an increase of 3.4195%, surpassing OBJ as the highest scoring function. Only 2 other Type 3 functions show any improvement over their Type 2 scores. Those features are SUBJ with an actual increase of 2.4708% and ADJUNCT with an actual increase of 1.4416%. These increases can probably be attributed to the reduction in feature occurrences over the reference set; there were slightly fewer features to identify, so proportionally they identified more correct functions.

Type 1 Lexicalised Duplicate Multiple Function Annotations

Scores for Type 1 multi-function experiments are shown in Table 6.4. These treebanks are annotated with all functions, the top five most frequently occurring functions, and SUBJ and OBJ; as a result each treebank has a high number of features present. F-scores for these experiments range from 56.4711 to 57.711, with SUBJ_OBJ achieving the best feature identification score. As was the case for Type 1 single functions, a high number of infrequently occurring types makes it difficult to learn features.

	features			occ
	precision	recall	f-score	#
ALL	63.4943	51.7761	57.0396	2590
TOP5	62.6428	51.4064	56.4711	2453
SUBJ_OBJ	60.8574	54.8739	57.7110	1190

Table 6.4: Feature detection scores for Type 1 Lexicalised Duplicate Multiple Function annotated treebanks.

Type 2 Duplicate Multiple Functions Annotations

Scores for Type 2 multi-function experiments are shown in Table 6.5. F-scores for these experiments range from 84.4528 to 85.1899, with SUBJ_OBJ again achieving the best feature identification score. We note that each of these treebanks has shown an f-score increase of approximately 28%, a result of the reduction in the number of distinct node labels. It appears that SUBJ_OBJ is the best performing combination of functions, and proportionally it is, correctly identifying 1021 of 1198 features. However, when we consider how many features each experiment aims to correctly

	features			occ
	precision	recall	f-score	#
ALL	87.7928	82.0592	84.8292	2603
TOP5	87.2460	81.8329	84.4528	2466
SUBJ_OBJ	87.1616	83.3055	85.1899	1198

Table 6.5: Feature detection scores for Type 2 Duplicate Multiple Function annotated treebanks.

identify, we see that ALL and TOP5 each identify a very high number; ALL identifies 2209 features, while TOP5 identifies 2083.

Type 3 Minimal Multiple Functions Annotations

Scores for Type 3 multi-function experiments are shown in Table 6.6. F-scores for these experiments range from 85.5295 to 86.2402, a slight increase on Type 2 f-scores due to a small reduction in the number of features present in the data. Each treebank shows an increase on its Type 2 scores; ALL increases by 1.1202%, TOP5 by 1.0767% and SUBJ_OBJ by 1.0503%. However SUBJ_OBJ again achieves the best feature identification score, showing consistency across all three types.

	features			occ
	precision	recall	f-score	#
ALL	88.7582	83.3129	85.9494	2445
TOP5	88.0220	83.1743	85.5295	2306
SUBJ_OBJ	87.8987	84.6432	86.2402	1107

Table 6.6: Feature detection scores for Type 3 Minimal Multiple Function annotated treebanks.

6.1.2 Atomic Feature Annotations

Atomic Preterminal Annotations

Scores for these experiments are shown in Table 6.7. F-scores range from 84.7458 (PROPER) to 100 (ABBREV, NUMBER-TYPE). Scores of 100 were achieved only by features which had relatively low occurrence counts (ABBREV 30, NUMBER-TYPE 95). We narrow our focus to only those features which have an occurrence count of 100 or more which reduces the range of f-scores to 88.0303 to 95.5182. This threshold discards all but the 5 most frequently occurring features: GRAIN, MOOD, NUM, PERS and PRON-TYPE. Of these features, the highest feature accuracy score is achieved by PERS, and the lowest by MOOD. We are satisfied that where there is a good atomic preterminal feature distribution, we accurately identify a very high proportion (on average 90.8289% of the time).

	features			occ
	precision	recall	f-score	#
ABBREV	100.0000	100.0000	100.0000	30
GRAIN	92.8678	91.4988	92.1782	2035
MOOD	90.4984	85.6932	88.0303	678
NUM	94.4909	92.8678	93.6723	2678
NUMBER-TYPE	100.0000	100.0000	100.0000	95
PERS	96.2170	94.8294	95.5182	2843
PRON-TYPE	95.2795	90.7692	92.9697	845
PROPER	100.0000	73.5294	84.7458	34

Table 6.7: Feature detection scores for Atomic Preterminal annotated treebanks.

Atomic Root Annotations

A selection of the scores for this experiment is shown in Table 6.8. We show here only features which have 100 or more features in the reference set. Features (and their occurrences) which are not displayed here are ADEG-DIM (11), DEIXIS (11), GEND (34), POL (3), PREDET-TYPE (4), TEMPORAL (1) and TIME (1). These features are omitted due to their low frequency of occurrence, which usually results in easily achieved high scores. It is interesting to note, however, that POL, TEMPORAL and TIME achieve f-scores of 0; the parser failed to correctly identify a single feature.

For features displayed in Table 6.8, f-scores range from 70.3614 (TENSE) to 92.283 (PSEM), while occurrences range from 108 (TYPE) up to 1115 (CASE). It is interesting to note that the least frequent feature TYPE achieves the second highest f-score of 90.4977, while the second least frequent feature TENSE scores the lowest of all (70.3614). In addition, the third least frequent feature scores 87.9079, one of the highest f-scores for this experiment. We conclude that for a reasonable distribution of features, we accurately identify them on average 85.07% of the time.

Multiple Atomic Annotations

The scores for this experiments are shown in Table 6.9. Although scores for each treebank are quite high, they are quite different to the average scores achieved by these features singly. The average for NUM and PERS individually would be 94.595;

	features			occ
	precision	recall	f-score	#
ADJUNCT-TYPE	90.1442	80.2998	84.9377	467
ADEGREE	86.2454	87.5472	86.8914	265
ADV-TYPE	85.0394	77.8846	81.3049	416
ANIM	84.7826	77.8271	81.1561	451
ATYPE	87.7395	88.0769	87.9079	260
CASE	87.5676	84.1558	85.8278	1155
GERUND	88.4956	92.5926	90.4977	108
LAYOUT-TYPE	91.8991	84.3902	87.9847	820
PASSIVE	84.9508	78.2383	81.4565	772
PERF	87.2456	81.5990	84.3279	788
PROG	85.9079	79.2500	82.4447	800
PSEM	92.7835	87.3786	90.0000	309
PTYPE	91.4013	93.1818	92.2830	308
SPEC-TYPE	90.2045	83.0525	86.4809	1009
STMT-TYPE	87.7256	82.4661	85.0146	884
TENSE	74.1117	66.9725	70.3614	218
TYPE	88.4956	92.5926	90.4977	108
VTYPE	85.3061	78.7688	81.9073	796

Table 6.8: Selected feature detection scores for Atomic Root annotated treebanks.

although these features appear to perform better when used separately, the parser has still managed to accurately identify 3707 features. The average f-score for PERF, PROG and TENSE is 79.044; an actual increase in score of 4.8986 for the combined features indicates that these features work well together. The average for PERF, PROG, TENSE, PASSIVE and MOOD is 81.3224; their combined treebank achieves an f-score increase of 1.8588%. This suggests that associated features do better when they are all present.

	features			occ
	precision	recall	f-score	#
NUM_PERS	90.3162	86.4839	88.3585	4195
PERF_PROG_TENSE	88.0822	80.1746	83.9426	802
PERF_PROG_TENSE_PASSIVE_MOOD	87.7579	79.0582	83.1812	807

Table 6.9: Feature detection scores for Multiple Atomic annotated treebanks.

6.1.3 Lexical Feature Annotations

Lexical Preterminal Annotations

The scores for this experiment are shown in Table 6.10. There is only one feature, PRON-FORM, in this category. It achieves a comparatively high feature identification f-score of 92.1833.

	features			occ
	precision	recall	f-score	#
PRON-FORM	96.6102	88.1443	92.1833	194

Table 6.10: Feature detection scores for Lexical Preterminal annotated treebanks.

Lexical Root Annotations

The scores for this experiment are shown in Table 6.11. A pattern we have observed to this point indicates that a very low number of feature occurrences results in easy feature identification and high f-scores. This holds for CONJ-FORM-COMP which has only 3 occurrences and achieves an f-score of 100. However, PREDET-FORM occurs only 4 times, and although a high f-score would be expected, 44.444 is the result achieved. From this figure, we can see how one or two misidentified features in a treebank with a low feature distribution can drastically alter scores.

	features			occ
	precision	recall	f-score	#
COMP-FORM	85.0467	73.3871	78.7879	124
CONJ-FORM	78.7879	50.3226	61.4173	155
CONJ-FORM-COMP	100.0000	100.0000	100.0000	3
PCASE	69.8171	69.6049	69.7108	329
PREDET-FORM	40.0000	50.0000	44.4444	4
PRT-FORM	55.5556	41.6667	47.6190	12
SPEC-FORM	87.7193	83.4725	85.5432	599

Table 6.11: Feature detection scores for Lexical Root annotated treebanks.

Again we focus our attention on features which have a high number of occurrences, over 100. The 4 remaining features, COMP-FORM, CONJ-FORM, PCASE and

SPEC-FORM, achieve f-scores in the range 61.4173 to 85.5432, with the highest score yielded by SPEC-FORM, the most frequently occurring feature.

Multiple Atomic Lexical Annotations

The scores for this experiment are shown in Table 6.12. As we observed for LEXICAL ROOT features, where a feature has a very low occurrence count, any mistake costs dearly, as can be seen by the very poor f-score achieved here by PREDET-FORM_PREDET-TYPE: 44.444. This score is consistent with the f-scores achieved by PREDET-FORM and PREDET-TYPE separately; they both score 44.444. Table 6.12 clearly shows the contrast between frequently and infrequently occurring features. Where we have a generous frequency distribution, the parser does well at learning features, and can achieve very high f-scores.

	features			occ
	precision	recall	f-score	#
PREDET-FORM_PREDET-TYPE	40.0000	50.0000	44.4444	4
SPEC-TYPE_SPEC-FORM	91.1514	83.6595	87.2449	1022

Table 6.12: Feature detection scores for Multiple Atomic Lexical annotated treebanks.

6.1.4 English: Discussion

From the results presented in section 6.1, we identify some interesting trends which show a strong correlation between the number of feature occurrences and the parser's ability to correctly identify features.

It is clear from the very high results achieved by low occurring features, such as OBL-COMP and TOPIC-INT which yield feature identification f-scores of 100%, that where there are very few features to correctly detect, high f-scores can generally be expected, but cannot always be guaranteed. When there are very few features to detect, even a single misidentification has a huge impact on feature accuracy scores; that is, getting 1 out of 3 features wrong will show a much bigger decrease in scores

than 1 out of 100.

Furthermore, in cases where the number of feature occurrences is very low, the parser does not really learn anything about these features. In the same way that we require a detailed analysis of a variety of experiments to establish how well the GF-DOP model performs, the parser cannot establish a performance pattern from one or two features distributed over an entire treebank. We examine the GF-DOP model's f-scores for groups of features to identify trends, for example which feature combinations perform best? In the same way, the GF-DOP model must be trained on a generous distribution of features to identify real patterns and trends; that is, which features have the greatest impact, either positive or negative, on parses generated?

One fact we ascertain from our experiment scores is that feature sparseness has an enormously negative impact on the parser's performance. For Type 1 Lexicalised Duplicate Functions, the parser manages to correctly identify over half the features present. Although this result is hardly trivial, we note that the reduction in feature sparseness between Type 1 and Type 2 Duplicate Functions boosts scores by 30% for the most frequently occurring functions. The same trend is seen for Type 1 and Type 2 multiple function annotated treebanks; over 56% of the time, Type 1 features are correctly identified. Upon reduction of the number of distinct annotations, that is using Type 2 annotations, we see increases of approximately 29%.

With regard to atomic features (which form the bulk of the information present in f-structures) we conclude that these features are accurately identified 85–90% of the time. We see that some combinations of atomic features work well together and may lead to improved performance when compared to the features in use individually. For example, the combined use of PERF, PROG and TENSE shows an increase of almost 5% over the average scores achieved by the individual features. Furthermore, the combined use of PERF, PROG, TENSE, PASSIVE and MOOD shows an increase of almost 2% over the average scores achieved by the features individually.

Generally lower scores were observed for lexical features when compared to

atomic features, although promisingly, for lexical features with a good number of occurrences, scores of at least 61% were achieved. Treebanks annotated with combined atomic and lexical features showed most clearly that for low frequency features even a single mis-identification costs dearly in f-scores, while high frequency features provide much more material to learn from, resulting in much higher scores.

6.2 French: Feature Detection Accuracy

6.2.1 Functional Annotations

Type 1 Lexicalised Duplicate Function Annotations

A selection of the scores for Type 1 single function experiments are shown in Table 6.13. We show here only those features which have one or more features in the test set. The only score not displayed here, and which had no features to identify in the test set, is OBL-COMP; this feature occurred very rarely in the original annotated treebank (only 6 occurrences).

	features			occ
	precision	recall	f-score	#
ADJUNCT	73.6348	68.5465	70.9996	1259
COMP	54.1667	65.0000	59.0909	20
COMPOUND	82.5243	69.6721	75.5556	122
OBJ	79.2672	75.4252	77.2985	1176
OBJ2	94.1176	64.0000	76.1905	25
OBL	89.2308	85.9259	87.5472	135
OBL-AGT	100.0000	100.0000	100.0000	1
PRON-REL	100.0000	100.0000	100.0000	15
SPEC	100.0000	100.0000	100.0000	1
SUBJ	66.8151	66.2252	66.5188	453
TOPIC-REL	100.0000	100.0000	100.0000	15
XCOMP	72.3404	55.7377	62.9630	61

Table 6.13: Selected feature detection scores for Type 1 Lexicalised Duplicate Function annotated treebanks.

F-scores for Type 1 feature annotations range from 59.0909 (COMP) to 100 (OBL-AGT, PRON-REL, SPEC, TOPIC-REL). Scores of 100 were achieved only by features

which had a very low occurrence count (OBL-AGT 1, PRON-REL 15, SPEC 1, TOPIC-REL 15); where there are not many features to be correctly identified, it is easier to get them all right. If we narrow our focus to only those features which have an occurrence count of 100 or more, this range is narrowed from 66.5188 to 87.5472. Only the 5 most frequently occurring features fall into this range; ADJUNCT, COMPOUND, OBJ, OBL and SUBJ. Of these, the highest feature accuracy score is achieved by OBL, and the lowest by SUBJ. An interesting correlation may be noted between the f-scores for these 5 features and their occurrence counts. As for English, the features with the highest number of occurrences seem to achieve lower feature accuracy scores. We list here these features and their occurrence counts in decreasing order according to their f-scores:

- OBL (135) f-score = 87.5472
- OBJ (1176) f-score = 77.2985
- COMPOUND (122) f-score = 75.5556
- ADJUNCT (1259) f-score = 70.9996
- SUBJ (453) f-score = 66.5188.

However, if we consider how many features were actually correctly identified, we see an altered picture. Now we list these features and the number of correctly identified features in decreasing order according to the number of correctly identified features:

- OBJ features correctly identified = 909
- ADJUNCT features correctly identified = 894
- SUBJ features correctly identified = 301
- OBL features correctly identified = 118
- COMPOUND features correctly identified = 92.

Although ADJUNCT and SUBJ have comparatively low f-scores, we see that they manage to correctly identify 894 and 301 features respectively, over 7.2 and 2.5 times the number of features identified by the best performing feature OBL. At this stage, we remind the reader that Type 1 feature annotated treebanks tend to comprise a high number of different node labels with very low frequencies, leading to feature sparseness. A reduction in feature sparseness paints a considerably different picture.

Type 2 Duplicate Function Annotations

A selection of the scores for Type 2 single function experiments are shown in Table 6.14. As for Type 1 features, we show here only those features which have one or more features in the test set. F-scores for Type 2 feature annotations range from 62.2222 (COMP) to 100 (OBL-AGT, PRON-REL, SPEC, TOPIC-REL). Scores of 100 were achieved by the same features as for Type 1. COMP is again the lowest scoring feature; this feature has a very low number of occurrences (20) and shows little improvement between Types 1 and 2 (actual increase of 3.1313%).

	features			occ
	precision	recall	f-score	#
ADJUNCT	91.0095	88.9746	89.9805	1297
COMP	56.0000	70.0000	62.2222	20
COMPOUND	86.5546	84.4262	85.4772	122
OBJ	88.2096	85.8844	87.0315	1176
OBJ2	85.0000	68.0000	75.5556	25
OBL	89.9225	85.9259	87.8788	135
OBL-AGT	100.0000	100.0000	100.0000	1
PRON-REL	100.0000	100.0000	100.0000	15
SPEC	100.0000	100.0000	100.0000	1
SUBJ	81.5678	84.9890	83.2432	453
TOPIC-REL	100.0000	100.0000	100.0000	15
XCOMP	78.7234	60.6557	68.5185	61

Table 6.14: Selected feature detection scores for Type 2 Duplicate Function annotated treebanks.

Narrowing our focus to features which have an occurrence count above 100, the range of f-scores is reduced to 83.2432 to 89.9805. Again, only the 5 most

frequently occurring features fall into this range. This time the highest feature accuracy score is achieved by ADJUNCT, and the lowest again by SUBJ. We see that the reduction in feature sparseness has greatly boosted scores for some frequent features; ADJUNCT achieved an actual increase of almost 19%, SUBJ has seen an actual increase of almost 17% while COMPOUND and OBJ show actual increases of almost 10% each. Normally, we see a much smaller impact on features which are less frequent; however in this case, COMPOUND (whose occurrence count is 1176) and OBJ (whose occurrence count is only 122) have seen almost exactly the same increase. In keeping with the expected trend, OBL has an actual increase of only 0.3316%. We see that the decrease in feature sparseness has had a positive impact on the parser's ability to correctly identify features, normally features with a high number of occurrences. The parser has been able to learn these features quite well.

Type 3 Minimal Function Annotations

A selection of the scores for Type 3 single function experiments are shown in Table 6.15. As for Type 1 features, we show here only features which have one or more features in the test set. F-scores for Type 3 feature annotations range from 62.2222 (COMP) to 100 (OBL-AGT, PRON-REL, SPEC, TOPIC-REL). Scores of 100 were achieved by the same features as for Types 1 and 2. Again, COMP is the lowest scoring feature, having shown no improvement over the previous annotation types, and very little improvement when compared to Type 1 (actual increase of 3.1313%). Focusing again on features which have an occurrence count above 100, the range of f-scores is reduced to 84.1699 to 89.9682, a slightly narrower range than for Type 2, but the same 5 most frequently occurring features. Changes between Type 2 and 3 scores are usually the result of a reduction in the feature occurrences.

Of the 5 most frequently occurring Type 3 features, 3 achieve small actual increases over their Type 2 scores: OBJ 0.4128%, COMPOUND 0.7153%, SUBJ 0.9267%. OBL yields exactly the same score, while ADJUNCT shows an actual decrease of -0.0123%. Where we see an increase in score, this may be attributed to a reduction

in the number of features we aim to identify; where there are fewer features to detect, the parser may get a larger proportion of them right. Correspondingly, where a decrease is observed, this may be attributed to the increased number of features to identify; features which the parser had previously accurately identified are no longer present, so it identifies proportionally less.

	features			occ
	precision	recall	f-score	#
ADJUNCT	91.0556	88.9064	89.9682	1271
COMP	56.0000	70.0000	62.2222	20
COMPOUND	88.0342	84.4262	86.1925	122
OBJ	88.7884	86.1404	87.4443	1140
OBJ2	85.0000	68.0000	75.5556	25
OBL	89.9225	85.9259	87.8788	135
OBL-AGT	100.0000	100.0000	100.0000	1
PRON-REL	100.0000	100.0000	100.0000	15
SPEC	100.0000	100.0000	100.0000	1
SUBJ	83.2061	85.1562	84.1699	384
TOPIC-REL	100.0000	100.0000	100.0000	15
XCOMP	78.7234	60.6557	68.5185	61

Table 6.15: Selected feature detection scores for Type 3 Minimal Function annotated treebanks.

Type 1 Lexicalised Duplicate Multiple Function Annotations

Scores for Type 1 multi-function experiments are shown in Table 6.16. These treebanks are annotated with all functions, the top five most frequently occurring functions (in both the English and French data sets), and SUBJ and OBJ; as a result each treebank has a high number of features present.

	features			occ
	precision	recall	f-score	#
ALL	77.0169	69.7789	73.2195	3256
ENG5	76.2230	69.0003	72.4321	3071
FRE5	77.0091	70.0795	73.3811	3145
SUBJ_OBJ	76.0618	72.5599	74.2696	1629

Table 6.16: Feature detection scores for Type 1 Lexicalised Duplicate Multiple Function annotated treebanks.

F-scores for these experiments range from 72.4321 to 74.2696, with SUBJ_OBJ

achieving the best feature identification score. We observed for Type 1 single functions, a high number of infrequently occurring features makes it difficult to learn features; however, these scores are non-trivial; between 1210 and 2384 Type 1 multi-function features are correctly identified in this experiment.

Type 2 Duplicate Multiple Function Annotations

Scores for Type 2 multi-function experiments are shown in Table 6.17. F-scores for these experiments range from 86.0602 to 87.5306, with ENG5 (previously the lowest scoring combination) achieving the best feature identification score. SUBJ_OBJ now scores the lowest of the four combinations. The reduction in feature sparseness has had greater impact on the treebanks with the highest numbers of occurrences; ENG5 shows an actual increase of over 15%, FRE5 and ALL show actual increases of almost 14%, while SUBJ_OBJ shows an actual increase of over 11.7%.

	features			occ
	precision	recall	f-score	#
ALL	88.5256	85.4888	86.9807	3294
ENG5	88.9701	86.1370	87.5306	3109
FRE5	88.7484	85.9881	87.3464	3183
SUBJ_OBJ	87.0603	85.0829	86.0602	1629

Table 6.17: Feature detection scores for Type 2 Duplicate Multiple Function annotated treebanks.

Type 3 Minimal Multiple Function Annotations

Scores for Type 3 multi-function experiments are shown in Table 6.18. F-scores for these experiments range from 86.5711 to 87.9343, a slight increase on Type 2 f-scores due to a small reduction in the number of features present in the data. Each treebank shows an increase on its Type 2 scores; ALL increases by 0.336%, ENG5 by 0.4037%, FRE5 by 0.2956% and SUBJ_OBJ by 0.5109%. As for Type 2, ENG5 achieves the best feature identification score, with SUBJ_OBJ again scoring lowest.

	features			occ
	precision	recall	f-score	#
ALL	89.0533	85.6465	87.3167	3163
ENG5	89.6684	86.2660	87.9343	2978
FRE5	89.3733	85.9764	87.6420	3052
SUBJ_OBJ	87.9485	85.2362	86.5711	1524

Table 6.18: Feature detection scores for Type 3 Minimal Multiple Function annotated treebanks.

6.2.2 Atomic Feature Annotations

Atomic Preterminal Annotations Scores for these experiments are shown in Table 6.19. F-scores range from 92.4370 (PROPER) to 100 (INV, NEG-FORM). Scores of 100 were achieved only by features which had relatively low occurrence counts (INV 3). We narrow our focus to only those features which have an occurrence count of 100 or more which reduces the range of f-scores to 92.437–96.4386. This threshold retains the 7 most frequently occurring features: GRAIN, MOOD, NUM, PERS, PRON-TYPE, PROPER and REFL. Of these features, the highest feature accuracy score is achieved by REFL (96.4286), and the lowest by PROPER (92.437). We are satisfied that where there is a good atomic preterminal feature distribution, we accurately identify a very high proportion (on average 94.7185).

	features			occ
	precision	recall	f-score	#
GRAIN	96.5622	94.7886	95.6672	2341
INV	100.0000	100.0000	100.0000	3
MOOD	97.5652	92.5743	95.0042	606
NUM	95.2782	92.4714	93.8538	3666
NUMBER-TYPE	100.0000	98.8095	99.4012	84
PERS	96.6993	94.9580	95.8207	3332
PRON-TYPE	95.2135	92.4623	93.8177	796
PROPER	94.8276	90.1639	92.4370	244
REFL	97.6845	95.2045	96.4286	709

Table 6.19: Feature detection scores for Atomic Preterminal annotated treebanks.

Atomic Root Annotations A selection of the scores for Type 1 single function experiments are shown in Table 6.20. We show here only features which have 100

or more features in the reference set. Features (and their occurrences) which are not displayed here are ADJUNCT-LAYOUT (55), CONJOINED (63), CONJTYPE (49), ADEG-DIM (5), DEIXIS (23), FOO (2), NE (67), NEG (4), PCASE-TYPE (20), PREDET-TYPE (0), PREVERB-OBJ (3), STRESSED (0), TIME (6) and TYPE (54). These features are omitted due to their low frequency of occurrence, which usually results in easily achieved high scores. It is interesting to note, however, that FOO and PREVERB-OBJ achieve f-scores of 0; these experiments failed to correctly identify a single feature.

	features			occ
	precision	recall	f-score	#
ADEGREE	93.7500	95.5882	94.6602	204
ADJUNCT-TYPE	94.2073	87.7841	90.8824	352
ADV-TYPE	88.7550	80.9524	84.6743	273
APOS	95.4286	93.2961	94.3503	179
ATYPE	94.9458	94.6043	94.7748	278
CASE	90.8560	83.0961	86.8030	1124
GEND	86.3399	82.0614	84.1463	2018
LAYOUT-TYPE	85.9023	79.7557	82.7149	573
PASSIVE	92.1127	86.0526	88.9796	760
PERF	91.9685	85.0073	88.3510	687
PSEM	95.5556	93.4783	94.5055	322
PTYPE	94.5736	93.0283	93.7946	918
SPEC-TYPE	90.0442	87.8575	88.9374	1853
STATUS	90.8676	85.5914	88.1506	1395
STMT-TYPE	93.0591	84.9765	88.8344	426
TENSE	89.4737	80.1887	84.5771	212
VTYPE	91.1807	85.5808	88.2920	749

Table 6.20: Selected feature detection scores for Atomic Root annotated treebanks.

For features displayed in Table 6.20, f-scores range from 82.7149 (LAYOUT-TYPE) to 94.7748 (ATYPE), while occurrences range from 179 (APOS) up to 2018 (GEND). If we examine the relationship between f-score and number of occurrences for the 17 features presented in Table 6.20, we see an interesting correlation; the four highest scoring features' (ATYPE, ADEGREE, PSEM, APOS) numbers of occurrences are some of the lowest ranks presented (13th, 16th, 12th and 17th respectively). The features with the highest number of occurrences (GEND, SPEC-TYPE, STATUS, CASE) show some of the lower scores in this group (ranking 16th, 8th, 12th and 13th respectively). As mentioned above, where there are very few features, it is generally easy to achieve

a high score. These trends suggest that we must find a balance between a generous distribution of features and the distinct number of features present; too few features present results in an easily achieved very high score, too many distinct features results in feature sparseness and low feature detection accuracy score. For treebanks in this group with a reasonable distribution of features, we are satisfied that we accurately identify them on average 89.26% of the time.

Multiple Atomic Annotations The scores for this experiments are shown in Table 6.21. Feature detection accuracy scores range from 87.832 to 88.9632. Although

	features			occ
	precision	recall	f-score	#
NUM_PERS	90.8413	86.1964	88.4579	4948
NUM_PERS_GEND	90.5897	85.0490	87.7320	5003
PERF_TENSE	92.0312	84.7482	88.2397	695
PERF_TENSE_PASSIVE_MOOD	92.7476	85.4756	88.9632	778

Table 6.21: Feature detection scores for Multiple Atomic annotated treebanks.

scores for each treebank are quite high, they are quite different to the average scores achieved by these features singly. The average for NUM and PERS individually would be 94.8372; although these features appear to perform better when used separately, the parser has still managed to accurately identify 4377 features. The average for NUM, PERS and GEND's individual scores is 91.2736, 3.5416% higher than their combined treebank's f-score. The average f-score for PERF and TENSE is 86.454; an actual increase in score of 1.786% for the combined features indicates that these features work well together. The average for PERF, TENSE, PASSIVE and MOOD is 89.2279; their combined treebank does not manage to achieve an f-score increase, instead yielding a decrease of -0.2647%. This suggests that associated features may not be as helpful in French as we had seen in our English experiments.

6.2.3 Lexical Feature Annotations

Lexical Preterminal Annotations

The scores for this experiment are shown in Table 6.22. Only the features with low feature occurrences obtain f-scores of 100. The 2 remaining features we consider to have a reasonable feature distribution (AUX-SELECT 1040, PRON-FORM 170); these features achieve very high scores of 96.0275 and 95.7055 respectively.

	features			occ
	precision	recall	f-score	#
AUX-SELECT	97.9980	94.1346	96.0275	1040
NEG-FORM	100.0000	100.0000	100.0000	30
PRECONJ-FORM	100.0000	100.0000	100.0000	14
PRON-FORM	100.0000	91.7647	95.7055	170

Table 6.22: Feature detection scores for Lexical Preterminal annotated treebanks.

Lexical Root Annotations

The scores for this experiment are shown in Table 6.23. We show here only features which had at least 1 occurrence in the reference set. A pattern we have observed to this point indicates that a very low number of feature occurrences results in easy feature identification and high f-scores; although this is frequently the case, we have here another example of a feature with a low number of occurrences, but for which the parser fails to identify even one (FORM).

	features			occ
	precision	recall	f-score	#
COMP-FORM	82.7160	74.0331	78.1341	181
CONJ-FORM	90.3226	87.5000	88.8889	160
FORM	0.0000	0.0000	0.0000	3
PCASE	88.3173	84.2536	86.2376	978
SPEC-FORM	88.6541	85.5150	87.0563	932

Table 6.23: Selected feature detection scores for Lexical Root annotated treebanks.

Now we focus our attention on features which have a high number of occurrences,

over 100. The 4 remaining features, COMP-FORM, CONJ-FORM, PCASE and SPEC-FORM, achieve f-scores in the range 78.1341 to 88.8889, with the highest score yielded by CONJ-FORM, the least frequently occurring feature.

Multiple Atomic Lexical Annotations

The scores for this experiment are shown in Table 6.24. We show again only results for treebanks with one or more features present in the reference set. Of the two features in this category, one combined treebank had no feature occurrences present (PREDET-FORM_PREDET-TYPE), consistent with its features's individual treebanks; the other (SPEC-TYPE_SPEC-FORM) had a very high number of occurrences, 1854. We see SPEC-TYPE_SPEC-FORM achieves a reasonably high feature detection accuracy score of 88.8525. Examination of our data set shows that SPEC-TYPE may have one of nine values, and SPEC-FORM one of fifteen; although there are many possible distinct combinations of values for SPEC-TYPE_SPEC-FORM, we can see that the high number of occurrences has enabled us to learn these features well.

	features			occ
	precision	recall	f-score	#
SPEC-TYPE_SPEC-FORM	90.0332	87.7023	88.8525	1854

Table 6.24: Feature detection scores for Multiple Atomic Lexical annotated treebanks.

6.2.4 French: Discussion

The results presented in section 6.2 display meaningful trends which support the GF-DOP hypothesis with regard to task 1, feature detection accuracy.

Some general trends (which can also be seen from the English results) concern the number of occurrences of features in a treebank. We have seen very infrequently occurring features, such as INV, NEG and NEG-FORM, yield feature identification scores of 100%; where there are very few features to correctly detect, we find that we normally score very highly, although this is not guaranteed. Misidentification

of features in a treebank which has very few features present has a very noticeable detrimental effect on feature detection accuracy scores.

Furthermore, where there are very few features present in the training data, the parser cannot establish any useful patterns; it does not learn how to apply sparse features well.

We ascertain from our experiments, particularly visible in the results for functional annotations, that for a good distribution of features, we correctly identify features a large proportion of the time. Although TYPE 1 LEXICALISED DUPLICATE FUNCTIONS are correctly identified at least two thirds of the time, the difference made by the reduction of feature sparseness is clearly evident when comparing the feature detection accuracy scores for TYPE 1 and TYPE 2 DUPLICATE FUNCTION annotations. For most of the frequently occurring features we see an actual increase of between 10 and 19%. Similarly, while TYPE 1 LEXICALISED DUPLICATE MULTIPLE FUNCTIONS are correctly identified at least 72% of the time, actual increases of approximately 12 to 15% are observed in the movement from TYPE 1 LEXICALISED DUPLICATE MULTIPLE FUNCTIONS to TYPE 2 DUPLICATE MULTIPLE FUNCTIONS annotations. These leaps in feature detection accuracy are due to the reduction in feature sparseness; that is, there are fewer distinct annotations present, and so more examples of each from which the parser establishes feature environments.

Focussing next on atomic features, we conclude that for a reasonable distribution of features, the parser accurately identifies this class 82.7–96.4% of the time. By far the most common group of features, this wide range is the result of a subset of 40 features of varying degrees of frequency. The range of feature detection accuracy scores for combined atomic features is considerably narrower; 87.7–88.9% over four groups of combined features. However, these groups of combined features did not manage to outperform their average individual scores, indicating that groups of associated features may not provide the boost we had hope for, for French at least.

A noticeably wider range of feature detection accuracy scores is noted for the

smaller group (12) of lexical features, 78–96% for 6 treebanks with a reasonable feature distribution. Only 1 feature distorts this range, as the majority of these features score between 86 and 96%. Recalling that lexical features showed the greatest overall increases and decreases, we conclude that the few lexical features present in the corpus have a considerable impact on the performance of the parser.

Given the generally high feature detection scores achieved by the parser, we conclude that the GF-DOP model has succeeded at its first task, feature detection accuracy. This evidence supports the GF-DOP hypothesis' first assertion: that the parser can learn grammatical features accurately.

6.3 Summary

Results for these experiments show that, for a training set with a reasonable distribution of features, the parser is able to establish useful patterns which it then reapplies to accurately identify a high proportion of features in test sentences. Where there are very few training features, one of two things may happen; for a very low number of features, the parser often correctly achieves a high score, but for a simplified task. In such an instance, even a single misidentification dramatically reduces the f-score achieved. The second possibility is that there is a handful of features present in the training data; there are not enough to learn the appropriate environments well, but this higher frequency of occurrences in the treebank means there will be more features to be correctly identified in the test set. Most likely, the parser will not have learned enough to score well at this task.

The results of these experiments are conclusive evidence to support the GF-DOP hypothesis' assertion that the parser can learn grammatical features accurately. In the next chapter, we will examine how well the parser uses this information to produce more accurate phrase-structure trees.

Chapter 7

Task 2 Results and Discussion: Parse Accuracy

This chapter examines the GF-DOP model's performance at parse accuracy. As illustrated in Figure 7.1, we present results first for the English experiments carried out, and then for the French. Further to the language division shown in Figure 7.1,

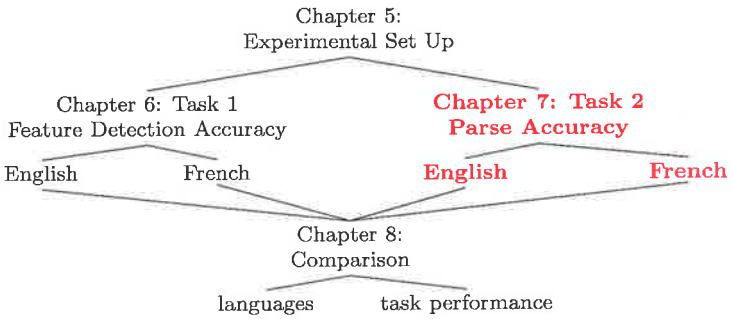


Figure 7.1: Illustration of relationships between Chapters 5, 6, 7 and 8.

and the feature classification described in section 4.2, features are again divided into three main categories (functions, atomics and lexicals), with further subdivisions as illustrated in Figure 6.2.

For each table of results presented in this section, the first column names the feature-annotated treebanks being investigated. The second and third columns, marked unlabelled and labelled, give the results for parse accuracy. The fourth

column (occ) indicates the number of feature annotations present in the reference set; that is the total number of feature annotations we aim to identify across the 8 test splits (720 test sentences in both English and French.) Where an occurrence count is zero, this indicates that while the feature was present in the training set (we only annotate with features which occur more than once in the treebank, as described in section 5.2), no annotation occurrences were found in the reference set.

For each table in sections 7.1 (English parse accuracy) and 7.2 (French parse accuracy), the first line of scores presented corresponds to the baseline: the experiment described carried out on a treebank with no grammatical feature annotations. The baseline scores in each table are identical and repeated for convenience only. Coverage for the English baseline is 93.89% and for French is 95.6944%. It remains constant for all experiments due to the GF-DOP model's backing-off capability. Any sentence which could not be fully parsed was assigned the most probable sequence of partial parses and grouped together under a dummy root node labelled 'TOP'.

Complete score charts for each of the experiments are given in the appendix (sections A and B) for completeness, but here we focus on those subsets of results which illustrate the most important points; we show here only those features which outperformed the BASELINE with either unlabelled or labelled f-score, as well as the features which result in the greatest decrease in score. In each table, the BASELINE scores are shown in blue, as are any scores which match the BASELINE exactly. Unlabelled and labelled scores which outperform the BASELINE are shown in red. The remaining unlabelled and labelled figures, in black, scored lower than the BASELINE. The highest and lowest overall scores are emphasized in bold.

As for task 1, we begin with English treebanks annotated with a single function, before proceeding to multi-function annotated treebanks, treebanks with atomic annotations, and finally treebanks with lexical annotations. We then examine the same groups of features for French.

We expect that the addition of functions will greatly improve the parser's accuracy. In particular, we believe Type 1 Lexicalised Duplicate Functions (both singly

and multiply annotated treebanks) will provide the most useful information. However, we concede that it is likely that Type 1 features' performance will be hindered by feature sparseness. This hypothesis will be tested by analysis of the performance of Type 2 Duplicate Functions and Type 3 Minimal Functions. No score increases between Types 1 and 2 will indicate either that functions are actually not as helpful as we had expected, or that feature sparseness has not been a problem. Large increases will suggest that delexicalised function annotation is worthwhile, but the parser has had insufficient data on which to train.

Furthermore, we believe that treebanks annotated with combinations of related features will outperform average scores for treebanks annotated with these features individually. As the GF-DOP model is an approximation of the LFG-DOP model, we endeavour to generate treebanks which comprehensively replicate the LFG-DOP model; the interaction of related features is one LFG behaviour we attempt to model.

The bulk of the features present in the f-structures from which we generated our annotations were features which we classed as "atomic". As these features form the majority, and provide much grammatical detail, we expect these features to greatly assist the parser's performance, particularly when we combine commonly co-occurring and co-dependent features.

The remaining features we classified as "lexical". Lexicalisation often shows improvements in PCFG parsing. However, the frequency of occurrence of lexical features in our data set is generally quite low. Given their overall lower frequency of occurrence, and the fact that they provide specific information which is most useful at a limited surface level, rather than internal structure, we do not expect these features to be as beneficial as atomic or functional features.

A final factor we expect to influence parse accuracy is the frequency of occurrences of features in the training data. We expect to see the greatest impact on treebanks with very high frequencies of annotation occurrences. Where we train the parser on a treebank with a very high number of features, we expect that the parser will learn these features well, thus boosting the overall parse accuracy.

7.1 English: Parse Accuracy

For all experiments in this section, the unlabelled BASELINE f-score was 96.245%. The labelled BASELINE f-score was 92.863%. Coverage for all experiments was 93.8889%.

7.1.1 Functional Annotations

As described in section 5.2, for each functional annotation we generate three treebanks which we call Type 1 Lexicalised Duplicate Functions, Type 2 Duplicate Functions and Type 3 Minimal Functions (samples illustrated in equations 5.1, 5.2 and 5.3 respectively). Here we examine the results from our preferred annotation type, Type 1, and compare its performance to our Type 2 and Type 3 treebanks.

Type 1 Lexicalised Duplicate Function Annotations

A selection of the scores for Type 1 experiments are shown in Table 7.1. Focusing initially on the unlabelled f-scores, of 17 functions, 6 outperform the baseline score (ADJUNCT, APP, OBJ, OBL, SPEC, XCOMP: average increase of 0.0.0489%); ADJUNCT and XCOMP give the highest improvements (of 0.0886% and 0.0728% respectively). Of the remaining functions, 8 maintain the BASELINE score (COMP, COMP-EX, OBL-AGT, OBL-COMP, PRON-INT, PRON-REL, TOPIC-INT, TOPIC-REL), while 3 yield a decrease (COMPOUND, OBJ2, SUBJ: average decrease of -0.026%).

	unlabelled			labelled			occ
	precision	recall	fscore	precision	recall	fscore	#
BASELINE	96.0913	96.3991	96.2450	92.7148	93.0117	92.8630	0
ADJUNCT	96.1427	96.5253	**96.3336**	92.7204	93.0894	92.9045	834
APP	96.1010	96.4088	96.2547	92.7148	93.0117	92.8630	0
OBJ	96.1141	96.5059	96.3096	92.8371	93.2156	**93.0260**	776
OBL	96.1107	96.4185	96.2644	92.7825	93.0797	92.9309	16
SPEC	96.1390	96.4282	96.2834	92.8005	93.0797	92.9399	71
SUBJ	96.0344	96.3700	**96.2019**	92.5041	92.8273	**92.6654**	418
XCOMP	96.1594	96.4768	96.3178	92.6768	92.9826	92.8295	123

Table 7.1: Selected parse accuracy scores for Type 1 Lexicalised Duplicate Function annotated treebanks.

The labelled f-scores show that 4 of 17 functions outperform the baseline (ADJUNCT, OBJ, OBL, SPEC: average increase of 0.0873%), with the greatest improvement achieved by OBJ, with an increase of 0.163%. Of the remainder, 8 maintain the BASELINE score (APP, COMP, COMP-EX, OBL-COMP, PRON-INT, PRON-REL, TOPIC-INT, TOPIC-REL) and 5 yield a decrease (COMPOUND, OBJ2, OBL-AGT, SUBJ, XCOMP: average decrease of -0.05334%). A single annotation, SUBJ, yields the biggest drop for both unlabelled and labelled scores, with decreases of -0.0431% and -0.1976% respectively.

Although unlabelled scores show a greater number of improvements over the BASELINE, labelled scores show the greatest average increase (unlabelled average increase of 0.0489% versus labelled average increase of 0.0873%); we see many small improvements in chunking performance, and fewer larger improvements in labelling accuracy. In addition, the absolute average increases for both evaluation types are greater than absolute average decreases (unlabelled -0.026%, labelled -0.05334%). Thus far, this annotation type shows promising improvement over the parser's BASELINE scores; this is particularly evident in the labelled f-scores.

One feature which yielded particularly interesting results is APP; although there were zero occurrences in the reference set (that is we were aiming to identify zero features in our output parses) we have seen an improvement over the unlabelled BASELINE score. Examination of the original annotated treebank shows very few occurrences of APP (4). However, these 4 annotations have clearly altered the probability mass assigned to fragments. It appears that this slightly altered probability distribution has lead the parser to select a different fragment set to parse some input strings; this has lead to an improvement in the structure assigned by it to some input strings, and as a result, a higher unlabelled f-score.

Type 2 Duplicate Function Annotations

A selection of the scores for Type 2 experiments are shown in Table 7.2. Unlabelled Type 2 f-scores show that of 17 functions, 4 outperformed the BASELINE figure

(ADJUNCT, APP, OBJ, XCOMP: average increase of 0.090875%), 10 maintained the score and 3 yielded a decrease (OBJ2, SPEC, SUBJ: average decrease of -0.0144%). This time OBJ and ADJUNCT give the greatest overall improvements (of 0.1938% and 0.1453% respectively). Here XCOMP achieves the third highest score, whereas for Type 1 the top three features (in decreasing order) were ADJUNCT, XCOMP and OBJ.

	unlabelled			labelled			occ
	precision	recall	fscore	precision	recall	fscore	#
BASELINE	96.0913	96.3991	96.2450	92.7148	93.0117	92.8630	0
ADJUNCT	96.2365	96.5447	96.3903	92.9373	93.2350	93.0859	834
APP	96.1010	96.4088	96.2547	92.7148	93.0117	92.8630	0
OBJ	96.2848	96.5932	**96.4388**	93.0244	93.3223	**93.1731**	776
SUBJ	96.0542	96.3991	**96.2263**	92.5532	92.8856	**92.7191**	422
XCOMP	96.1014	96.4185	96.2597	92.6478	92.9535	92.8004	123

Table 7.2: Selected parse accuracy scores for Type 2 Duplicate Function annotated treebanks.

Looking now at labelled scores, we see that only 3 outperform the BASELINE (ADJUNCT, OBJ, OBL, average increase of 0.0906%), 11 maintain the score and 5 features cause a decrease (COMPOUND, OBJ2, OBL-AGT, SUBJ, XCOMP, average decrease of -0.05784%). The largest improvements are achieved by OBJ (0.3101%) and ADJUNCT (0.2229%). For both unlabelled and labelled f-scores, SUBJ again causes the greatest decrease (-0.187% and -0.1439% respectively).

Again, unlabelled scores show a greater number of improvements over the BASELINE. However labelled scores show the greatest average increase (unlabelled average increase of 0.090875% versus labelled average increase of 0.2665%: labelled increase is almost three times the unlabelled increase). For Type 2 annotations, the absolute unlabelled average increase is over seven times greater than the absolute unlabelled average decrease (increase of 0.090875% versus decrease of -0.0144%). This is also the case for labelled averages, where the absolute average increase is almost five times the average decrease (increase of 0.2665% versus decrease of -0.05784%). Overall, Type 2 annotations appear to have a very positive impact on the parser's chunking

ability, illustrated by generally improved unlabelled scores, but perhaps less so on labelling accuracy, shown in the comparatively weaker labelled scores.

Type 3 Minimal Function Annotations

A selection of the scores for Type 3 experiments are shown in Table 7.3. The final function annotation type shows similar trends to the previous two: unlabelled f-scores show 5 functions outperforming the BASELINE figure (ADJUNCT, APP, COMPOUND, OBJ, XCOMP, average increase of 0.08146%), 9 maintaining the score and 3 causing a decrease (OBJ2, SPEC, SUBJ, average decrease of -0.0144%). Again, the greatest increases are yielded by OBJ (0.2035%) and ADJUNCT (0.16%).

	unlabelled			labelled			occ
	precision	recall	fscore	precision	recall	fscore	#
BASELINE	96.0913	96.3991	96.2450	92.7148	93.0117	92.8630	0
ADJUNCT	96.2465	96.5641	96.4050	93.0251	93.3320	**93.1783**	802
APP	96.1010	96.4088	96.2547	92.7148	93.0117	92.8630	0
COMPOUND	96.1107	96.4185	96.2644	92.6471	92.9438	92.7952	274
OBJ	96.2945	96.6029	**96.4485**	93.0244	93.3223	93.1731	776
SUBJ	96.0542	96.3991	**96.2263**	92.5629	92.8953	**92.7288**	331
XCOMP	96.1014	96.4185	96.2597	92.6478	92.9535	92.8004	123

Table 7.3: Selected parse accuracy scores for Type 3 Minimal Function annotated treebanks.

Type 3 labelled f-scores show similar patterns to those of Type 2: we see this time that only 3 outperform the BASELINE (ADJUNCT, OBJ, average increase of 0.3127%), 10 maintain the score and 5 features cause a decrease (COMPOUND, OBJ2, OBL-AGT, SUBJ, XCOMP, average decrease of -0.0559%). As for Type 2, the only increases are achieved by ADJUNCT and OBJ, but this time in reverse order. ADJUNCT has the greatest increase (of 0.3153%), followed closely by OBJ (0.3101%). As was noted for both Type 1 and Type 2 scores, the greatest decrease is yielded by SUBJ, for both unlabelled (-0.187%) and labelled f-score (-0.1342%).

Unexpectedly, results of Type 3 annotation have much in common with those of Type 1. Unlabelled scores show a greater number of improvements over the BASELINE, but again labelled scores show the greatest average increase (unlabelled

average increase of 0.08146% versus labelled average increase of 0.3127%). In addition, the absolute average increases for both evaluation types (unlabelled 0.08146%, labelled 0.3127%) are greater than absolute average decreases (unlabelled -0.0144%, labelled -0.0559%). Type 3 labelled f-scores show the greatest average increase of the three types (Type 1 0.0873%, Type 2 0.2665%, Type 3 0.3127%), while maintaining roughly the same labelled average decrease as Types 1 and 2 (Type 1 -0.05334%, Type 2 -0.05784%, Type 3 -0.0559%).

Figures 7.4, 7.5 and 7.6 show scores for treebanks annotated with multiple functions. Annotation type ALL refers to the treebank annotated with all 17 functions listed as single annotations. Annotation type TOP5 refers to the treebank annotated with the 5 most frequently occurring functions (ADJUNCT, OBJ, SUBJ, COMPOUND and XCOMP). Annotation type SUBJ_OBJ refers to the treebank annotated with those two functions only.

Type 1 Lexicalised Duplicate Multiple Function Annotations

All scores for Type 1 multi-function annotated experiments are shown in Table 7.4. From this table we can see that Type 1 multi-function annotations consistently outperform the BASELINE figures, for both unlabelled and labelled f-scores for each of the three treebanks. Improvements range from 0.1274% (ALL) to 0.2235% (TOP5) for unlabelled f-scores, and 0.1384% (ALL) to 0.249% (TOP5) for labelled f-scores. The average increase for unlabelled evaluation is 0.1818%, and for labelled evaluation 0.1762%.

	unlabelled			labelled			occ
	precision	recall	fscore	precision	recall	fscore	#
BASELINE	96.0913	96.3991	96.2450	92.7148	93.0117	92.8630	0
ALL	96.1814	96.5641	96.3724	92.8171	93.1865	93.0014	2590
TOP5	96.2959	96.6418	**96.4685**	92.9014	93.2350	**93.0679**	2453
SUBJ_OBJ	96.2669	96.6126	96.4395	92.8820	93.2156	93.0485	1190

Table 7.4: Parse accuracy scores for Type 1 Lexical Duplicate Multiple Function annotated treebanks.

Type 2 Duplicate Multiple Function Annotations

Scores for Type 2 multi-function annotated experiments are shown in Table 7.5. As for Type 1 multi-function annotations, Type 2 consistently outperform the BASELINE. Improvements over unlabelled f-scores range from 0.1794% (SUBJ_OBJ) to 0.2626% (TOP5), and for labelled f-scores range from 0.2959% (SUBJ_OBJ) to 0.5143% (ALL). The average increase in unlabelled f-score is 0.2314% and labelled f-score is 0.4354%. Although TOP5 was the overall highest scoring annotation for Type 1 multi-function annotations, for Type 2 it achieves the greatest improvement over the BASELINE figures for unlabelled f-score only; ALL yields the highest increase in labelled f-score.

	unlabelled			labelled			occ
	precision	recall	fscore	precision	recall	fscore	#
BASELINE	96.0913	96.3991	96.2450	92.7148	93.0117	92.8630	0
ALL	96.3339	96.6612	96.4973	93.2192	93.5359	93.3773	2603
TOP5	96.3257	96.6903	**96.5076**	93.1831	93.5359	93.3592	2466
SUBJ_OBJ	96.2658	96.5835	96.4244	93.0057	93.3126	93.1589	1198

Table 7.5: Parse accuracy scores for Type 2 Duplicate Multiple Function annotated treebanks.

Type 3 Minimal Multiple Function Annotations

Scores for Type 3 multi-function annotated experiments are shown in Table 7.6. Scores for Type 3 multi-function annotations show the same trend as for Type 2: all annotation types outperform the BASELINE figures for both unlabelled and labelled f-scores. Improvements over the unlabelled BASELINE range from 0.2085% (SUBJ_OBJ) to 0.282% (ALL), and for the labelled BASELINE range from 0.325% (SUBJ_OBJ) to 0.4865% (ALL). The average increase in unlabelled f-scores is 0.2415%, and in labelled f-scores is 0.4201%.

Across all three annotation types, both unlabelled and labelled multi-function annotation f-scores show increases over BASELINE scores; an average (across all 3 types) unlabelled increase of 0.218266% and an average labelled increase of 0.343966%.

	unlabelled			labelled			occ
	precision	recall	fscore	precision	recall	fscore	#
BASELINE	96.0913	96.3991	96.2450	92.7148	93.0117	92.8630	0
ALL	96.3450	96.7097	96.5270	93.1735	93.5262	93.3495	2445
TOP5	96.2788	96.6806	**96.4792**	93.1181	93.5067	93.3120	2306
SUBJ_OBJ	96.2949	96.6126	96.4535	93.0347	93.3417	93.1880	1107

Table 7.6: Parse accuracy scores for Type 3 Minimal Multiple Function annotated treebanks.

The smallest average increase noted in multi-function annotations was the Type 1 labelled f-score, 0.176266%. The largest average increase was the Type 2 labelled f-score, 0.435466%; this is the largest average increase of scores to this point. None of the combinations of features tested in these experiments yielded scores exactly equal to, or less than, the BASELINE f-scores. This indicates that annotating with combinations of frequently occurring functions decidedly improves a parser's ability to generate highly accurate phrase-structure trees.

7.1.2 Atomic Feature Annotations

Atomic Preterminal Annotations

The result of this experiment is shown in Table 7.7. Focusing initially on the unlabelled f-scores, of 8 atomic pre-terminal features, 4 outperform the baseline score (ABBREV, MOOD, PERS, PRON-TYPE, average increase of 0.053825%): PRON-TYPE and PERS give the greatest improvements (of 0.131% and 0.0402% respectively). Of the remaining features, none maintain the BASELINE score exactly, while 4 yield a decrease (GRAIN, NUM, NUMBER-TYPE, PROPER average decrease of -0.04335%). The largest decrease is seen in the unlabelled f-score for NUM, with a drop of 0.0858%.

The labelled f-scores show that only 2 of 8 atomic pre-terminal features outperform the baseline (PERS, PRON-TYPE, average increase of 0.21715%), with the greatest improvement achieved by PERS, with an increase of 0.2353%. Of the remainder, a further 2 maintain the BASELINE score (ABBREV, NUMBER-TYPE) and again, 4 yield a decrease (GRAIN, MOOD, NUM, PROPER, average decrease of -0.2087%). The

	unlabelled			labelled			occ
	precision	recall	fscore	precision	recall	fscore	#
BASELINE	96.0913	96.3991	96.2450	92.7148	93.0117	92.8630	0
ABBREV	96.1300	96.4379	96.2837	92.7148	93.0117	92.8630	30
GRAIN	96.0337	96.3506	96.1919	92.3672	92.6720	**92.5194**	2035
MOOD	96.0828	96.4185	96.2504	92.6589	92.9826	92.8205	678
NUM	95.9687	96.3506	**96.1592**	92.4207	92.7885	92.6043	2678
NUMBER-TYPE	96.0720	96.3797	96.2256	92.7148	93.0117	92.8630	95
PERS	96.0944	96.4768	96.2852	92.9138	93.2835	**93.0983**	2843
PRON-TYPE	96.2175	96.5350	**96.3760**	92.9090	93.2156	93.0620	845
PROPER	96.0902	96.3700	96.2299	92.5385	92.8079	92.6730	34

Table 7.7: Parse accuracy scores for Atomic Preterminal annotated treebanks.

largest decrease is seen in the labelled f-score for GRAIN, with a drop of -0.3436%.

Although unlabelled scores show a greater number of improvements over the BASELINE, labelled scores show the greatest average increase (unlabelled average increase of 0.053825% versus labelled average increase of 0.21715%). In addition, the absolute average increases for both evaluation types (unlabelled 0.053825%, labelled 0.21715%) are greater than absolute average decreases (unlabelled 0.04335%, labelled 0.2087%). Thus far, this annotation type shows promising improvement over the parser's BASELINE scores; this is particularly evident in the labelled f-scores.

Atomic Root Annotations

A section of the results of this experiment are shown in Table 7.8. Unlabelled f-scores show that of 25 atomic root features, 14 outperformed the BASELINE figure (ADJUNCT-TYPE, ADEGREE, ADV-TYPE, ANIM, ATYPE, CASE, GEND, POL, PTYPE, SPEC-TYPE, STMT-TYPE, TEMPORAL, TENSE, TIME, average increase of 0.05449%), none maintained the score exactly and 11 yielded a decrease (ADEG-DIM, DEIXIS, GERUND, LAYOUT-TYPE, PASSIVE, PERF, PREDET-TYPE, PROG, PSEM, TYPE, VTYPE, average decrease of -0.06008%). This time ADJUNCT-TYPE and ANIM give the greatest overall improvements (of 0.1744% and 0.1356% respectively). The largest decrease is seen in the unlabelled f-score for PASSIVE, with a drop of -0.1031%.

Looking now at labelled scores, we see that 16 atomic root features outperform

	unlabelled			labelled			occ
	precision	recall	fscore	precision	recall	fscore	#
BASELINE	96.0913	96.3991	96.2450	92.7148	93.0117	92.8630	0
ADJUNCT-TYPE	96.2655	96.5738	**96.4194**	92.9373	93.2350	**93.0859**	467
ADEGREE	96.1501	96.4768	96.3132	92.8129	93.1282	92.9703	265
ADV-TYPE	96.1018	96.4282	96.2647	92.6485	92.9632	92.8056	416
ANIM	96.2268	96.5350	96.3806	92.7728	93.0700	92.9212	451
ATYPE	96.1401	96.4573	96.2984	92.8219	93.1282	92.9748	260
CASE	96.0847	96.4670	96.2755	92.8654	93.2350	93.0498	1155
GEND	96.1777	96.4670	96.3221	92.8005	93.0797	92.9399	34
GERUND	96.0724	96.3894	96.2306	92.8122	93.1185	92.9651	108
PASSIVE	96.0209	96.2632	**96.1419**	92.7679	93.0020	92.8848	772
PERF	96.0403	96.2826	96.1613	92.7583	92.9923	92.8751	788
POL	96.1204	96.4282	96.2740	92.7148	93.0117	92.8630	3
PREDET-TYPE	96.0155	96.3603	96.1876	92.6015	92.9341	**92.7675**	4
PROG	96.1084	96.3603	96.2342	92.7880	93.0312	92.9094	800
PTYPE	96.0735	96.4185	96.2457	92.6499	92.9826	92.8160	308
SPEC-TYPE	96.1312	96.4670	96.2988	92.8136	93.1379	92.9755	1009
STMT-TYPE	96.1096	96.3894	96.2493	92.7998	93.0700	92.9347	884
TEMPORAL	96.1587	96.4573	96.3078	92.8108	93.0991	92.9547	1
TENSE	96.0824	96.4088	96.2453	92.7162	93.0312	92.8734	218
TIME	96.1490	96.4476	96.2981	92.8108	93.0991	92.9547	1
TYPE	96.0724	96.3894	96.2306	92.8122	93.1185	92.9651	108
VTYPE	96.0321	96.3118	96.1717	92.7223	92.9923	92.8571	796

Table 7.8: Selected parse accuracy scores for Atomic Root annotated treebanks.

the BASELINE (ADJUNCT-TYPE, ADEGREE, ANIM, ATYPE, CASE, GEND, GERUND, PASSIVE, PERF, PROG, SPEC-TYPE, STMT-TYPE, TEMPORAL, TENSE, TIME, TYPE, average increase of 0.08915%), 2 maintain the score (DEIXIS, POL) and 7 features cause a decrease (ADEG-DIM, ADV-TYPE, LAYOUT-TYPE, PREDET-TYPE, PSEM, PTYPE, VTYPE, average decrease of -0.04611%). PREDET-TYPE yields the greatest drop in f-score, with a decrease of -0.0955%.

This is the only (English) experiment where labelled scores achieved a greater number of improvements over the BASELINE scores than unlabelled scores. Promisingly, the labelled f-scores also show a greater average increase than unlabelled f-scores (average unlabelled increase of 0.05449% versus average labelled increase of 0.08915%), and the average labelled decrease is less than the average unlabelled decrease (average unlabelled decrease of -0.06008% versus average labelled decrease of -0.0461%). This is a promising indication that feature annotations, particularly

this group of atomic root feature annotations, not only improve the parser's chunking ability, but also its labelling accuracy; this feature appears to boost the overall quality of phrase-structure trees generated.

Multiple Atomic Annotations

The scores for this experiment are shown in Table 7.9. Unlabelled f-scores show that of the 3 multi-annotated treebanks (NUM_PERS, PERF_PROG_TENSE_PASSIVE_MOOD, PERF_PROG_TENSE), none outperform the BASELINE figure. In fact, all 3 score below the BASELINE, with an average decrease of -0.3243%; this is the largest decrease we note to this point. It is approximately 5.5 times greater than the next largest unlabelled average decrease of 0.06008, yielded by the atomic root group, and approximately 1.5 times the greatest labelled decrease (-0.2087% yielded by the atomic pre-terminal group).

	unlabelled			labelled			occ
	precision	recall	fscore	precision	recall	fscore	#
BASELINE	96.0913	96.3991	96.2450	92.7148	93.0117	92.8630	0
NUM_PERS	95.9695	96.3700	**96.1693**	92.5382	92.9244	**92.7309**	4195
PERF_PROG_TENSE	96.1088	96.3700	96.2392	92.7597	93.0117	92.8855	802
PERF_PROG_TENSE _PASSIVE_MOOD	96.1080	96.3506	96.2292	92.8551	93.0894	**92.9721**	807

Table 7.9: Parse accuracy scores for Multiple Atomic annotated treebanks.

Labelled scores show an improvement in the case of 2 out of the 3 features: PERF_PROG_TENSE and PERF_PROG_TENSE_PASSIVE_MOOD improve over the BASELINE by an average of 0.0658%. Only NUM_PERS yields a decrease of -0.1321%. That PERF_PROG_TENSE_PASSIVE_MOOD performs better than PERF_PROG_TENSE is an indication that additional detailed information proves useful in assisting the parser to generate better quality phrase-structure trees.

7.1.3 Lexical Feature Annotations

Lexical Preterminal Annotations

Only 1 feature is classified under this heading, PRON-FORM. It outperforms both the unlabelled and labelled scores for the BASELINE, by 0.0581% and 0.0485% respectively.

	unlabelled			labelled			occ
	precision	recall	fscore	precision	recall	fscore	#
BASELINE	96.0913	96.3991	96.2450	92.7148	93.0117	92.8630	0
PRON-FORM	96.1494	96.4573	96.3031	92.7632	93.0603	92.9115	194

Table 7.10: Parse accuracy scores for Lexical Preterminal annotated treebanks.

Lexical Root Annotations

The result of this experiment is shown in Table 7.11. Focusing initially on the unlabelled f-scores, of 7 lexical root features, 3 outperform the baseline score (COMP-FORM, PCASE, SPEC-FORM, average increase of 0.0505%): SPEC-TYPE gives the greatest improvement (of 0.0926%) overall. Of the remaining features, none maintain the BASELINE score exactly, while 4 yield a decrease (CONJ-FORM, CONJ-FORM-COMP, PREDET-FORM, PRT-FORM average decrease of -0.0227%).

	unlabelled			labelled			occ
	precision	recall	fscore	precision	recall	fscore	#
BASELINE	96.0913	96.3991	96.2450	92.7148	93.0117	92.8630	0
COMP-FORM	96.1107	96.4185	96.2644	92.6761	92.9729	92.8243	124
CONJ-FORM	96.1096	96.3894	96.2493	92.6546	92.9244	92.7893	155
CONJ-FORM-COMP	96.0817	96.3894	96.2353	92.7148	93.0117	92.8630	3
PCASE	96.1122	96.4573	96.2845	92.8627	93.1962	**93.0291**	329
PREDET-FORM	96.0155	96.3603	**96.1876**	92.6015	92.9341	**92.7675**	4
PRT-FORM	96.0720	96.3797	96.2256	92.7148	93.0117	92.8630	12
SPEC-FORM	96.1698	96.5059	**96.3376**	92.8426	93.1670	93.0046	599

Table 7.11: Parse accuracy scores for Lexical Root annotated treebanks.

The labelled f-scores show that only 2 of the 7 lexical root features outperform

the baseline (PCASE, SPEC-FORM, average increase of 0.15385%), the greatest improvement achieved by PCASE, with an increase of 0.1661%. Of the remainder, a further 2 maintain the BASELINE score (CONJ-FORM-COMP, PRT-FORM) and 9 yield a decrease (COMP-FORM, CONJ-FORM, PREDET-FORM, average decrease of -0.0693%). The largest decrease in both unlabelled and labelled f-scores is yielded by just one feature, PREDET-FORM, with a drop of -0.0574% for unlabelled scores and -0.0955% for labelled.

Again here unlabelled scores show a greater number of improvements over the BASELINE, but labelled scores show the greatest average increase (unlabelled average increase of 0.0505% versus labelled average increase of 0.15385%). In addition, the absolute average increases for both evaluation types (unlabelled 0.0505%, labelled 0.15385%) are greater than absolute average decreases (unlabelled 0.0227%, labelled 0.0693%). This pattern has occurred for several different annotation types and appears to be a common trend.

Multiple Atomic Lexical Annotations

The result of this experiment is shown in Table 7.12. Only 2 combined atomic-lexical treebanks were generated as we set out to test groups of related features, or features which commonly co-occur in f-structures. For both unlabelled and labelled f-scores,

	unlabelled			labelled			occ
	precision	recall	fscore	precision	recall	fscore	#
BASELINE	96.0913	96.3991	96.2450	92.7148	93.0117	92.8630	0
PREDET-FORM_PREDET-TYPE	96.0155	96.3603	96.1876	92.6015	92.9341	92.7675	4
SPEC-TYPE_SPEC-FORM	96.1505	96.4865	96.3182	92.8523	93.1767	93.0142	1022

Table 7.12: Parse accuracy scores for Multiple Atomic Lexical annotated treebanks.

1 feature combination outperformed the BASELINE figure, SPEC-TYPE_SPEC_FORM, with an unlabelled increase of 0.0732% and a labelled increase of 0.1512%. The other feature combination, PREDET-FORM_PREDET-TYPE underperformed for both evaluation types, with an unlabelled decrease of -0.0574% and a labelled decrease

of -0.0955%. This raises the suggestion that some combinations of features provide more useful detail than others, although for this pair of treebanks, it appears that the number of occurrences of features could provide more clarity. The feature combination which outperformed the BASELINE had a high number of feature occurrences, 1022, while the feature which scored lower had only 4. Given the large difference in the number of features present for each experiment, it is difficult to say conclusively which feature combination is most useful for a parser. From the number of annotations, we surmise that the more frequent feature is the more useful feature within a treebank; given that it occurs more often, it provides clarification more often than the lower occurring feature combination.

7.1.4 English: Discussion

We summarise the average increases and decreases for unlabelled and labelled scores for each of the twelve groups of experiments in Table 7.13 for ease of reference during this discussion. The greatest increases and decreases are highlighted in bold. Where there was no average increase or decrease, we indicate this was not applicable.

	unlabelled (%)		labelled (%)	
	average increase	average decrease	average increase	average decrease
TYPE 1 SINGLE FUNCTIONS	0.0489	0.026	0.0873	0.05334
TYPE 2 SINGLE FUNCTIONS	0.0908	0.0144	0.2665	0.0578
TYPE 3 SINGLE FUNCTIONS	0.0814	0.0144	0.3127	0.0559
TYPE 1 MULTIPLE FUNCTIONS	0.1818	n/a	0.17626	n/a
TYPE 2 MULTIPLE FUNCTIONS	0.2314	n/a	**0.4354**	n/a
TYPE 3 MULTIPLE FUNCTIONS	**0.2415**	n/a	0.4201	n/a
ATOMIC PRETERMINAL	0.05382	0.0433	0.2171	**0.2087**
ATOMIC ROOT	0.0544	0.06008	0.0891	0.0461
MULTIPLE ATOMIC	n/a	**0.3244**	0.0658	0.1321
LEXICAL PRETERMINAL	0.0581	n/a	0.0485	n/a
LEXICAL ROOT	0.0505	0.0227	0.1538	0.0693
MULTIPLE ATOMIC LEXICAL	0.0732	0.0574	0.1512	0.0955

Table 7.13: Summary of average increases and decreases for unlabelled and labelled scores for each annotation type.

A first glance at this table shows that for unlabelled evaluation, 11 of the 12

experiments showed some improvement over the BASELINE, with average increases ranging from 0.0505% (LEXICAL ROOT) to 0.2415% (TYPE 3 MULTIPLE FUNCTION). The only experiment which did not yield an average increase was MULTIPLE ATOMIC which in fact showed the greatest average decrease. Encouragingly, only 8 out of 12 experiments yielded an average decrease, ranging from -0.0144% (TYPE 2 SINGLE FUNCTIONS and TYPE 3 SINGLE FUNCTIONS) to -0.3244% (MULTIPLE ATOMIC), meaning that the remaining 4 experiments consistently outperformed the BASELINE; that is, for each of the remaining experiments (TYPE 1 MULTIPLE FUNCTIONS, TYPE 2 MULTIPLE FUNCTIONS, TYPE 3 MULTIPLE FUNCTIONS, LEXICAL PRETERMINAL) incorporation of feature annotations lead to a consistent overall improvement in chunking performance.

It is interesting to note that the average increase is almost always greater in magnitude than the average decrease, with only two exceptions to this pattern, ATOMIC ROOT and MULTIPLE ATOMIC. We conclude that annotation with grammatical features assists the parser to determine the sentence structure, with functions, particularly combinations of frequently occurring functions, providing the most useful information.

Consider now the scores for labelled evaluation; we see that each experiment yielded some average increase, ranging from 0.0485% (LEXICAL PRETERMINAL) to 0.4354% (TYPE 2 MULTIPLE FUNCTIONS); the greatest average increase here is almost nine times the smallest average increase. As for unlabelled scores, only 8 out of 12 experiments yielded an average decrease, ranging from -0.0461% (ATOMIC ROOT) to -0.2087% (ATOMIC PRETERMINAL). These figures illustrate a similar trend to those for unlabelled evaluation; the net increase in scores is greater than net decrease. This is evidence that annotation with features improves the quality of phrase-structure trees generated by the parser. Thus the GF-DOP hypothesis has been shown to hold true for the second task; the GF-DOP model can produce more accurate phrase-structure trees than the Tree-DOP model.

Having concluded thus far that the GF-DOP model, through the incorporation

of functions and feature annotations, can indeed produce more accurate phrase-structure trees than the Tree-DOP model alone, we focus for a moment on what type of features produce the best results. The summary of results in Table 7.13 shows that the largest increases were found in the MULTIPLE FUNCTION group; the biggest unlabelled increase was the result of the TYPE 3 MULTI-FUNCTIONS experiment, while the biggest labelled increase was achieved by TYPE 2 MULTIPLE FUNCTIONS. Score trends in Table 7.13 suggest that SINGLE FUNCTION experiments yielded the next best parses. This suggests that annotation with functions provides considerable assistance to the parser, and that combinations of frequently occurring functions makes the best use of these features. That the MULTIPLE FUNCTION group yields the best overall unlabelled and labelled scores indicates that this is the function type which best supports the GF-DOP hypothesis; it facilitates the best structure assignation and the best phrase-structure trees are generated.

The result we found most surprising was the comparatively poor performance achieved using ATOMIC annotations. This annotation type did not yield as large increases as we had hoped. This expectation was based on the fact that ATOMIC annotations provide such detailed information. Some of the ATOMIC annotations used provided us with descriptions ranging from sentence/phrase-structure, such as STMT-TYPE, (which we expected might be particularly useful given the data set used in these experiments) to grammatical agreements such as number, person and case. We had expected this type of information, which forms the bulk of the features in f-structures, to prove more helpful. We suggest that some features which performed well individually, such as PERS, ADJUNCT-TYPE, PRON-TYPE and CASE, were averaged with features which did not prove as useful (GRAIN, PROPER and, most unexpectedly, NUM), thus bringing down the average performance of the group.

Given our initial high expectation for ATOMIC annotations, we did not suppose that LEXICAL annotations would perform quite so well. However we were pleasantly surprised by this category, whose promising average results are shown in Table 7.13. LEXICAL annotations indicate the required surface form. It appears that this

concrete information is of greater assistance than was initially expected. Unlabelled scores show slightly better performance for the LEXICAL group compared to the ATOMIC group. Labelled scores show a comparable average increase, and slightly smaller average decreases for the LEXICAL group.

7.1.5 English: Other Points of Interest

We now present some more focused points of interest which arose from these experiments.

An interesting variation was noted for unlabelled evaluation of the function COMPOUND. As a Type 1 Lexicalised Duplicate Function annotation, COMPOUND scored lower than the BASELINE with a decrease in f-score of -0.0298%. As a Type 2 Duplicate Function annotation, it matched the BASELINE score exactly, and as a Type 3 Minimal Function annotation it exceeded the BASELINE by 0.194%. This was the only feature which exhibited this behaviour. As coverage over all experiments remains constant, we hypothesise that this increase in score was influenced by feature sparseness, with the score rising when fewer distinct annotations were present. This leads us to reflect on the size of our data set and the implications it has on the ideal functional annotation style for the GF-DOP model. We conclude that better parses will be achieved by training the parser on a data set which has a lower ratio of feature types to feature tokens; that is, we need high numbers of repeated features to reduce the impact of feature sparseness.

As mentioned in section 7.1.1 some interesting behaviour was observed on the treebank annotated with APP (TYPE 1 SINGLE FUNCTIONS). Although there were no features present in the reference set (that is we were not aiming to identify any features in our test sentences), the few features present in the original annotated treebank (4 for APP) showed a significant impact on the probability distribution of the fragment set. This alteration in the probability distribution led the parser to select a different fragment set to parse some input strings; this different fragment set yielded an increase in score over the BASELINE figure. This improvement was

repeated across Type 2 and Type 3 treebanks.

This behaviour appears to be the exception rather than the rule, as other features which had no occurrences in the reference sets, but a few occurrences in the annotated treebank (COMP-EX 3, OBJ2 3, OBL-AGT 2), did not achieve the same improvements.

As described in section 5.1, the data set used in these experiments was limited to a total of 980 English sentences, allowing us 890 sentences for training, and 90 for testing. Even for our limited data set, Type 1 functional annotations managed to achieve an overall improvement in scores, with average increases for both unlabelled and labelled evaluation exceeding the absolute value for average decreases. However, the considerable increase in average scores between Type 1 and Type 2 (unlabelled average increase Type 1 0.0489% versus Type 2 0.090875%, labelled average increase Type 1 0.0873% versus Type 2 0.2665%) for which there are exactly the same number of annotations (although dramatically fewer distinct types relative to the number of feature tokens), acts as an indicator that feature sparseness has negatively impacted upon our Type 1 performance. We have seen that Type 1 annotations can improve the parser's performance, and given enough data, we suggest that Type 1 experiments would perform as well as Type 2. Scaling up these experiments would be an interesting avenue for future work, with particular attention to Type 1 annotation experiments, and a comparison of the proportional improvements over each of the three functional annotation types.

Despite the somewhat limited scale, we believe these experiments have shown the first part of the GF-DOP hypothesis to hold true (for English): the GF-DOP model has produced better quality parses than the Tree-DOP model alone. We have identified functions, in particular combinations of the most frequently occurring functions, as the feature classification which leads to the greatest overall improvement, although we feel that even better results might be achieved on a larger data set with a more generous feature distribution.

7.2 French: Parse Accuracy

For all experiments in this section, the unlabelled BASELINE f-score was 96.3590%. The labelled BASELINE f-score was 93.3002%. Coverage for all experiments was 95.6944%.

7.2.1 Functional Annotations

As described in section 5.2, for each functional annotation we generate three treebanks which we call Type 1 Lexical Duplicate Function, Type 2 Duplicate Function and Type 3 Minimal Function (samples illustrated in equations 5.1, 5.2 and 5.3 respectively). Here we examine the results from our preferred annotation type, Type 1, and compare its performance to our Type 2 and Type 3 treebanks.

Type 1 Lexicalised Duplicate Function Annotations

A selection of the scores for Type 1 experiments are shown in Table 7.14. Focusing initially on the unlabelled f-scores, of 13 functions, 5 outperform the baseline score (ADJUNCT, COMPOUND, OBJ, SUBJ, XCOMP, average increase of 0.06768%); ADJUNCT and XCOMP give the highest improvements (of 0.1561% and 0.0779% respectively). Of the remaining functions, 6 maintain the BASELINE score (OBJ2, OBL, OBL-AGT, OBL-COMP, PRON-REL, SPEC, TOPIC-REL), while 2 yield a decrease (COMP, OBL, average decrease of -0.03465%).

The labelled f-scores show that 6 of 13 functions outperform the baseline (ADJUNCT, COMPOUND, OBJ, OBJ2, SPEC, XCOMP, average increase of 0.1075%), with the greatest improvement achieved by COMPOUND, with an increase of 0.2681%. Of the remainder, 4 maintain the BASELINE score (OBL-AGT, OBL-COMP, PRON-REL, TOPIC-REL) and 3 yield a decrease (COMP, OBL, SUBJ, average decrease of -0.0526%). A single annotation OBL yields the biggest drop for both unlabelled and labelled scores, with decreases of -0.0463% and -0.0693% respectively.

For this annotation type, labelled scores show the greatest number of improve-

	unlabelled			labelled			occ
	precision	recall	fscore	precision	recall	fscore	#
BASELINE	96.1723	96.5464	96.3590	93.1195	93.4816	93.3002	0
ADJUNCT	96.3058	96.7253	**96.5151**	93.2853	93.6917	93.4880	1259
COMPOUND	96.2191	96.6008	96.4096	93.3834	93.7539	**93.5683**	122
OBJ	96.2096	96.5464	96.3777	93.2253	93.5516	93.3882	1176
OBL	96.1187	96.5075	**96.3127**	93.0431	93.4194	**93.2309**	135
OBJ2	96.1723	96.5464	96.3590	93.1272	93.4894	93.3080	25
SPEC	96.1723	96.5464	96.3590	93.1272	93.4894	93.3080	1
SUBJ	96.2036	96.5853	96.3941	93.0580	93.4272	93.2422	453
XCOMP	96.2426	96.6319	96.4369	93.1980	93.5750	93.3861	61

Table 7.14: Selected parse accuracy scores for Type 1 Lexicalised Duplicate Function annotated treebanks.

ments over the BASELINE and the greatest average increase (unlabelled average increase of 0.06768% versus labelled average increase of 0.1075%.) We note that for both unlabelled and labelled scores, the increases are approximately twice the absolute decreases (unlabelled average increase 0.06768% versus unlabelled average decrease -0.03465%, labelled average increase 0.1075% versus labelled average decrease -0.0526%). Thus far, this annotation type shows promising improvement over the parser's BASELINE scores; this is particularly evident in the labelled f-scores.

Type 2 Duplicate Function Annotations

A section of the scores for Type 2 experiments are shown in Table 7.15. Unlabelled Type 2 f-scores show that of 13 functions, 5 outperformed the BASELINE figure (ADJUNCT, COMPOUND, OBJ, SUBJ, XCOMP, average increase of 0.0738%), 6 maintained the score and 2 yielded a decrease (COMP, OBL, average decrease of -0.03465%). This time ADJUNCT and OBJ give the greatest overall improvements (of 0.1172% and 0.0966% respectively).

Looking now at labelled scores, we see that 6 outperform the BASELINE (ADJUNCT, COMPOUND, OBJ, SPEC, SUBJ, XCOMP, average increase of 0.0849%), 4 maintain the score and 3 features cause a decrease (COMP, OBJ2, OBL, average decrease of -0.041%). The largest improvements are achieved by OBJ (0.1662%) and ADJUNCT (0.1102%). For both unlabelled and labelled f-scores, OBL again causes the greatest

	unlabelled			labelled			occ
	precision	recall	fscore	precision	recall	fscore	#
BASELINE	96.1723	96.5464	96.3590	93.1195	93.4816	93.3002	0
ADJUNCT	96.2670	96.6864	**96.4762**	93.2079	93.6139	93.4104	1297
COMPOUND	96.2185	96.5853	96.4015	93.2197	93.5750	93.3970	122
OBJ	96.2799	96.6319	96.4556	93.2961	93.6372	**93.4664**	1176
OBL	96.1187	96.5075	**96.3127**	93.0353	93.4116	**93.2231**	135
SPEC	96.1723	96.5464	96.3590	93.1272	93.4894	93.3080	1
SUBJ	96.2340	96.6008	96.4171	93.1964	93.5516	93.3737	453
XCOMP	96.2194	96.6086	96.4136	93.1670	93.5439	93.3551	61

Table 7.15: Selected parse accuracy scores for Type 2 Duplicate Function annotated treebanks.

decrease (-0.0463% and -0.0771% respectively).

We have seen the identical trends in Type 1 and Type 2 scores; for unlabelled scores 5 features improve over the BASELINE, 6 features maintain the BASELINE and 2 yield a decrease. For labelled scores 6 improve over the BASELINE, 4 maintain those scores and 3 yield a decrease. Again, labelled scores show the greatest average increase (unlabelled average increase of 0.0738% versus labelled average increase of 0.08495%). For Type 2 annotations, the absolute unlabelled average increase is over twice the absolute unlabelled average decrease (increase of 0.0738% versus decrease of -0.03465%). This is also the case for labelled averages; the absolute average increase is over twice the average decrease (increase of 0.0849% versus decrease of -0.041%).

Overall, Type 2 annotations show a positive impact on the parser's chunking ability, illustrated by improved unlabelled scores when compared to Type 1; we see an increase in the unlabelled average improvement (Type 1 0.06768% versus Type 2 0.0738%) and no extra decrease (Type 1 -0.03465% and Type 2 -0.03465%). However, Type 2 annotations do not show the same improvement for labelling accuracy, determined by the labelled scores. Type 2 shows a slightly smaller average decrease than Type 1 (Type 1 labelled average decrease -0.0526% versus Type 2 labelled average decrease -0.041%) and a smaller average increase (Type 1 labelled average increase 0.1075% versus Type 2 labelled average increase 0.0849%). We note that

the magnitude lost by the Type 2 average increase is almost twice the magnitude of the improvement seen in the Type 2 average decrease.

Type 3 Minimal Function Annotations

A selection of the scores for Type 3 experiments are shown in Table 7.16. The final function annotation type shows similar trends to the previous two: unlabelled f-scores show 5 functions outperforming the BASELINE figure (ADJUNCT, COMPOUND, OBJ, SUBJ, XCOMP, average increase of 0.08%), 6 maintaining the score and 2 causing a decrease (COMP, OBL, average decrease of -0.03465%). Again, the greatest increases are yielded by ADJUNCT (0.1172%) and OBJ (0.0966%).

	unlabelled			labelled			occ
	precision	recall	fscore	precision	recall	fscore	#
BASELINE	96.1723	96.5464	96.3590	93.1195	93.4816	93.3002	0
ADJUNCT	96.2670	96.6864	**96.4762**	93.1924	93.5983	93.3949	1271
COMPOUND	96.2185	96.5853	96.4015	93.2197	93.5750	93.3970	122
OBJ	96.2799	96.6319	96.4556	93.2806	93.6217	**93.4508**	1140
OBL	96.1187	96.5075	**96.3127**	93.0353	93.4116	**93.2231**	135
SPEC	96.1723	96.5464	96.3590	93.1272	93.4894	93.3080	1
SUBJ	96.2650	96.6319	96.4481	93.1964	93.5516	93.3737	384
XCOMP	96.2194	96.6086	96.4136	93.1670	93.5439	93.3551	61

Table 7.16: Selected parse accuracy scores for Type 3 Minimal Function annotated treebanks.

Type 3 labelled f-scores also show similar patterns to those of Type 2: we see that, again, 6 outperform the BASELINE (ADJUNCT, COMPOUND, OBJ, SPEC, SUBJ, XCOMP, average increase of 0.0797%), 4 maintain the score and 3 features cause a decrease (COMP, OBJ2, OBL, average decrease of -0.041%). The greatest increases are achieved by OBJ and COMPOUND; OBJ has the greatest increase (of 0.1506%), followed by COMPOUND (0.0968%). As was noted for both Type 1 and Type 2 scores, the greatest decrease is yielded by OBL, for both unlabelled (-0.0463%) and labelled f-score (-0.0771%).

Two clear patterns emerge from the results for Type 1, Type 2 and Type 3 annotations; exactly the same number of features (although not the exact same sets

of features) show increases and decreases for both unlabelled and labelled scores across all three types. ADJUNCT, OBJ and COMPOUND generally perform the best across all three types for both unlabelled and labelled scores, while OBL consistently scores lowest in each category. Furthermore, average increases in scores are always very close to, or more than, twice the average decreases in scores. We conclude that annotation with each of the three function types assists the parser in both chunking accuracy and parse accuracy; at the very least, use of functions is approximately twice as beneficial as it is harmful.

Type 1 Lexicalised Duplicate Multiple Function Annotations

All scores for Type 1 multi-function annotated experiments are shown in Table 7.17. In the parallel English experiment we generated three feature-combination treebanks: ALL, TOP5 and SUBJ_OBJ. For French we have generated four treebanks as the top five most frequently occurring features in French were not the same as the top five most frequently occurring features in English. For English those features were ADJUNCT, COMPOUND, OBJ, SUBJ and XCOMP. For French the top five were ADJUNCT, COMPOUND, OBJ, OBL and SUBJ. In order to more directly compare our parser's performance in both languages, we generated treebanks with both the English top five features (called ENG5) and the French top five (called FRE5).

	unlabelled			labelled			occ
	precision	recall	fscore	precision	recall	fscore	#
BASELINE	96.1723	96.5464	96.3590	93.1195	93.4816	93.3002	0
ALL	95.9421	96.3675	96.1543	92.7050	93.1161	**92.9101**	3256
ENG5	95.9721	96.3752	96.1733	92.7730	93.1627	92.9675	3071
FRE5	95.9340	96.3519	**96.1425**	92.8748	93.2794	93.0767	3145
SUBJ_OBJ	96.1386	96.4452	96.2917	93.0526	93.3494	93.2008	629

Table 7.17: Parse accuracy scores for Type 1 Lexicalised Duplicate Multiple Function annotated treebanks.

From Table 7.17 we see disappointing results. No combination of features has managed to outperform the BASELINE figure. The average unlabelled decrease is -0.1685% and, surprisingly, FRE5 has scored the lowest of all combinations, with

a decrease of -0.2165%. The average labelled decrease is -0.2614%; this time ALL scores lowest, showing a decrease of 0.3901%. As FRE5 and ALL have very high numbers of occurrences, we must conclude that feature sparseness has had a very strong, negative impact.

Type 2 Duplicate Multiple Function Annotations

Scores for Type 2 multi-function annotated experiments are shown in Table 7.18. Scores for Type 2 multi-function annotation experiments are more encouraging that those of Type 1; unlabelled scores show 3 combinations outperforming the BASELINE with an average increase of 0.087%, the highest scoring combination being SUBJ_OBJ with an increase of 0.1081%. The only combination continuing to yield low unlabelled scores for Type 2 is FRE5, showing a decrease of -0.1398%. Labelled scores fare better, however; all four combinations outperforming the BASELINE score, with an average improvement of 0.07735%. The highest individual increase is again yielded by SUBJ_OBJ (0.1387%).

	unlabelled			labelled			occ
	precision	recall	fscore	precision	recall	fscore	#
BASELINE	96.1723	96.5464	96.3590	93.1195	93.4816	93.3002	0
ALL	96.1973	96.6164	96.4064	93.1691	93.5750	93.3716	3294
ENG5	96.2518	96.6786	96.4647	93.1929	93.6061	93.3990	3109
FRE5	96.0490	96.4375	**96.2428**	93.1128	93.4894	93.3007	3183
SUBJ_OBJ	96.2951	96.6397	**96.4671**	93.2724	93.6061	**93.4389**	1629

Table 7.18: Parse accuracy scores for Type 2 Duplicate Multiple Function annotated treebanks.

The reduction in feature sparseness has had a clear impact on scores with every score in every category (in Table 7.18) showing some improvement. Unfortunately, FRE5 does not provide the assistance we had expected, with ENG5 yielding greater increases in both unlabelled and labelled evaluations.

Type 3 Minimal Multiple Function Annotations

Scores for Type 3 multi-function annotated experiments are shown in Table 7.19. We see similar trends in the unlabelled scores as for Type 2, with 3 combinations outperforming the BASELINE (ALL, ENG5, SUBJ_OBJ) resulting in an average increase of 0.0971%. Again, only FRE5 yielded a decrease (of -0.1398%). The same pattern is seen in the labelled evaluation; the same 3 combinations yield increases, on average 0.0844%, and FRE5 shows a decrease of -0.0776%.

	unlabelled			labelled			occ
	precision	recall	fscore	precision	recall	fscore	#
BASELINE	96.0913	96.3991	96.2450	92.7148	93.0117	92.8630	0
ALL	96.2435	96.6553	96.4489	93.2151	93.6139	93.4141	3163
ENG5	96.2822	96.6942	96.4878	93.1919	93.5905	93.3908	2978
FRE5	96.0329	96.4063	96.2192	93.0420	93.4039	93.2226	3052
SUBJ_OBJ	96.2710	96.5930	96.4318	93.1933	93.5050	93.3489	1524

Table 7.19: Parse accuracy scores for Type 3 Minimal Multiple Function annotated treebanks.

7.2.2 Atomic Feature Annotations

Atomic Preterminal Annotations

The result of this experiment is shown in Table 7.20. Focusing initially on the unlabelled f-scores, of 9 atomic pre-terminal features, 4 outperform the baseline score (MOOD, PRON-TYPE, PROPER, REFL, average increase of 0.0876%): REFL and PRON-TYPE give the greatest improvements (of 0.1866% and 0.1127% respectively). Of the remaining features, 2 maintain the BASELINE score exactly (INV, NUMBER-TYPE), while 3 yield a decrease (GRAIN, NUM, PERS, average decrease of -0.051%). The largest decrease is seen in the unlabelled f-score for PERS, with a drop of 0.0886%. Again the unlabelled average increase is larger than the absolute unlabelled average decrease.

The labelled f-scores show that 4 of 9 atomic pre-terminal features outperform the baseline (GRAIN, PRON-TYPE, PROPER, REFL, average increase of 0.1185%), with

	unlabelled			labelled			occ
	precision	recall	fscore	precision	recall	fscore	#
BASELINE	96.1723	96.5464	96.3590	93.1195	93.4816	93.3002	0
GRAIN	96.1637	96.5152	96.3391	93.1644	93.5050	93.3344	2341
MOOD	96.1965	96.5930	96.3943	93.0978	93.4816	93.2893	606
NUM	96.0759	96.5541	96.3144	92.9721	93.4350	**93.2030**	3666
PERS	96.0654	96.4764	**96.2704**	93.0757	93.4739	93.2743	3332
PRON-TYPE	96.2811	96.6630	96.4717	93.2982	93.6683	93.4829	796
PROPER	96.1807	96.5697	96.3748	93.1515	93.5283	93.3395	244
REFL	96.3511	96.7408	**96.5456**	93.3297	93.7072	**93.5181**	709

Table 7.20: Parse accuracy scores for Atomic Preterminal annotated treebanks.

the greatest improvement achieved again by REFL, with an increase of 0.2179%. Of the remainder, none maintain the BASELINE score, and a further 5 yield a decrease (INV, MOOD, NUM, NUMBER-TYPE, PERS, average decrease of -0.0299%), with the largest decrease seen in the labelled f-score for NUM, with a drop of -0.0972%. We note that the labelled average increase is approximately 4 times the absolute labelled average decrease.

Although unlabelled scores show a greater number of improvements over the BASELINE, labelled scores show a much greater average increase (unlabelled average increase of 0.0876% versus labelled average increase of 0.1185%) and less that two thirds the average decrease (unlabelled average decrease of -0.051% versus labelled average decrease of -0.0299%). In addition, the absolute average increases for both evaluation types are greater than absolute average decreases. This annotation type shows a considerable improvement over the parser's BASELINE scores.

Atomic Root Annotations

A selection of the results of this experiment are shown in Table 7.21. Unlabelled f-scores show that of 31 atomic root features, 12 outperformed the BASELINE figure (ADEGREE, CASE, CONJTYPE, DEIXIS, LAYOUT-TYPE, PASSIVE, PCASE-TYPE, PREDET-TYPE, PSEM, STMT-TYPE, TENSE, TYPE, average increase of 0.0274%), 10 maintained the score exactly (ADJUNCT-LAYOUT, CONJOINED, ADEG-DIM, FOO,

NEG, PREVERB-OBJ, STRESSED, TIME) and 11 yielded a decrease (ADJUNCT-TYPE, ADV-TYPE, APOS, ATYPE, GEND, NE, PERF, PTYPE, SPEC-TYPE, STATUS, VTYPE, average decrease of -0.0376%). This time CONJTYPE and PSEM give the greatest overall improvements (of 0.0768% and 0.0615% respectively). The largest decrease is seen in the unlabelled f-score for PTYPE, with a drop of -0.1208%.

	unlabelled			labelled			occ
	precision	recall	fscore	precision	recall	fscore	#
BASELINE	96.1723	96.5464	96.3590	93.1195	93.4816	93.3002	0
ADEGREE	96.1652	96.5541	96.3593	93.0896	93.4661	93.2774	204
APOS	96.1494	96.5308	96.3397	93.1278	93.4972	93.3121	179
CASE	96.1807	96.5697	96.3748	93.1051	93.4816	93.2930	1124
CONJTYPE	96.2713	96.6008	**96.4358**	93.1938	93.5128	93.3530	49
DEIXIS	96.1878	96.5619	96.3745	93.1350	93.4972	93.3157	23
GEND	96.1416	96.5230	96.3319	93.3292	93.6994	**93.5140**	2018
LAYOUT-TYPE	96.2033	96.5775	96.3900	93.0342	93.3961	93.2148	573
PASSIVE	96.1899	96.6164	96.4027	93.0922	93.5050	93.2981	760
PCASE-TYPE	96.1878	96.5619	96.3745	93.1195	93.4816	93.3002	20
PREDET-TYPE	96.1878	96.5619	96.3745	93.1195	93.4816	93.3002	0
PSEM	96.2486	96.5930	96.4205	93.2414	93.5750	93.4079	322
PTYPE	96.0629	96.4141	**96.2382**	93.0016	93.3416	**93.1713**	918
STMT-TYPE	96.1881	96.5697	96.3785	93.0503	93.4194	93.2345	426
TENSE	96.2021	96.5464	96.3739	93.0708	93.4039	93.2371	212
TYPE	96.1953	96.5619	96.3782	93.1499	93.5050	93.3271	54

Table 7.21: Selected parse accuracy scores for Atomic Root annotated treebanks.

Looking now at labelled scores, we see that only 6 atomic root features outperform the BASELINE (APOS, CONJTYPE, DEIXIS, GEND, PSEM, TYPE, average increase of 0.0714%), 7 maintain the score (CONJOINED, ADEG-DIM, FOO, PCASE-TYPE, PREDET-TYPE, PREVERB-OBJ, STRESSED) and 18 features cause a decrease (ADEGREE, ADJUNCT-LAYOUT, ADJUNCT-TYPE, ADV-TYPE, ATYPE, CASE, LAYOUT-TYPE, NE, NEG, PASSIVE, PERF, PTYPE, SPEC-TYPE, STATUS, STMT-TYPE, TENSE, TIME, VTYPE, average decrease of -0.0488%). Again PTYPE yields the greatest drop in f-score, with a decrease of -0.1289%.

Although unlabelled scores show a greater number of improvements over the BASELINE, labelled scores show the greatest average increase (unlabelled average increase of 0.0274% versus labelled average increase of 0.0714%). Interestingly, PSEM

is the second highest scoring unlabelled and labelled feature, while PTYPE and SPECTYPE are the lowest and second lowest scoring unlabelled and labelled features.

Multiple Atomic Annotations

The scores for this experiment are shown in Table 7.22. For our French experiments, we have generated one additional treebank; previously we tested the feature combination NUM_PERS, but now we test this combination and also NUM_PERS_GEND as GEND has shown itself to be considerably more informative for French than for English. Furthermore, the English treebank PERF_PROG_TENSE generated is replicated for French, but called PERF_TENSE only; no PROG feature was present in the French data set.

	unlabelled			labelled			occ
	precision	recall	fscore	precision	recall	fscore	#
BASELINE	96.1723	96.5464	96.3590	93.1195	93.4816	93.3002	0
NUM_PERS	96.2805	96.6475	**96.4636**	93.2817	93.6372	93.4591	4948
NUM_PERS_GEND	96.1652	96.5541	96.3593	93.3607	93.7383	**93.5491**	5003
PERF_TENSE	96.0731	96.4841	**96.2782**	92.9905	93.3883	**93.1890**	695
PERF_TENSE_PASSIVE_MOOD	96.2277	96.6319	96.4294	93.1139	93.5050	93.3090	778

Table 7.22: Parse accuracy scores for Multiple Atomic annotated treebanks.

Unlabelled f-scores show that of the 4 multi-annotated treebanks, 3 outperform the BASELINE figure (NUM_PERS, NUM_PERS_GEND, PERF_TENSE_PASSIVE_MOOD), with an average increase of 0.0584%. The highest scoring unlabelled combination is NUM_PERS with an increase of 0.1046%. This is an unexpected ranking; we had expected that NUM_PERS_GEND would score higher than NUM_PERS. However, unlabelled scores are an indication of chunking performance only, so we expect that NUM_PERS_GEND will show an improvement in parse accuracy.

The only unlabelled feature combination which scores lower than the BASELINE is PERF_TENSE, with a decrease of -0.0808%. As noted in section 7.1.2, where we use combinations of related features, the parser performs better by incorporating all features; the incorporation of PASSIVE and MOOD with PERF_TENSE leads to an

improvement in f-score of 0.1512%.

Labelled f-scores show 3 feature combinations outperforming the BASELINE with an average increase of 0.1388%. As we hoped, NUM_PERS_GEND improves parse accuracy more than NUM_PERS (by 0.09%), and in fact is the highest scoring labelled feature combination, showing an increase over the BASELINE of 0.2489%. The incorporation of GEND has shown the expected improvement.

Again, PERF_TENSE is the only combination which scores lower than the BASELINE, showing a decrease of -0.1112%. We also see a difference of 0.12% between the labelled scores for PERF_TENSE and PERF_TENSE_PASSIVE_MOOD. That PERF_TENSE_PASSIVE_MOOD has again performed better than PERF_TENSE is confirmation that additional detailed information proves useful in assisting the parser to generate better quality phrase-structure trees.

7.2.3 Lexical Feature Annotations

Lexical Preterminal Annotations

For French we identified a larger group of lexical pre-terminal annotated treebanks than for English; there are two language-specific features (AUX-SELECT and NEG-FORM), as well as a greater distribution of the feature PRECONJ-FORM (for English PRECONJ-FORM occurred in only a single sentence, and so we could not have determined any useful patterns).

	unlabelled			labelled			occ
	precision	recall	fscore	precision	recall	fscore	#
BASELINE	96.1723	96.5464	96.3590	93.1195	93.4816	93.3002	0
AUX-SELECT	96.2584	96.6553	**96.4564**	93.2295	93.6139	**93.4213**	1040
FORM	96.1723	96.5464	96.3590	93.1195	93.4816	93.3002	3
NEG-FORM	96.1726	96.5541	96.3630	93.1123	93.4816	93.2966	30
PRECONJ-FORM	96.1723	96.5464	96.3590	93.1195	93.4816	93.3002	14
PRON-FORM	96.1267	96.5230	**96.3245**	93.0126	93.3961	**93.2040**	170

Table 7.23: Parse accuracy scores for Lexical Preterminal annotated treebanks.

Unlabelled scores show that the two language-specific features outperform the

BASELINE at chunking: AUX-SELECT and NEG-FORM, with an average increase of 0.0507%. AUX-SELECT by itself shows quite a high increase (actually the highest for this group) of 0.0974%. PRON-FORM is the only score which is lower than the BASELINE, showing a drop of -0.0345%.

AUX-SELECT also performs best according to labelled scores; it is the only feature which outperforms the BASELINE score, with an increase of 0.1211%. PRECONJ-FORM again maintains the BASELINE, while NEG-FORM and PRON-FORM yield an average labelled decrease of -0.0499%. Interestingly, NEG-FORM shows almost the same labelled decrease (-0.0036%) as it does labelled increase (0.004%).

Lexical Root Annotations

The result of this experiment is shown in Table 7.24. Focusing initially on the unlabelled f-scores, of 8 lexical root features, 3 outperform the baseline score (CONJ-FORM, PCASE, PREDET-FORM, average increase of 0.1244%): PCASE gives the greatest improvement (of 0.2853%) overall. Of the remaining features, 3 maintain the BASELINE score exactly (COMP-FORM-ANAPH, CONJ-FORM-COMP, FORM), while 2 yield a decrease (COMP-FORM, SPEC-COMP, average decrease of -0.4224%).

	unlabelled			labelled			occ
	precision	recall	fscore	precision	recall	fscore	#
BASELINE	96.1723	96.5464	96.3590	93.1195	93.4816	93.3002	0
COMP-FORM	95.3244	95.7752	**95.5493**	91.3278	91.7597	**91.5433**	181
COMP-FORM-ANAPH	96.1723	96.5464	96.3590	93.1195	93.4816	93.3002	0
CONJ-FORM	96.2782	96.5853	96.4315	93.2620	93.5594	93.4105	160
CONJ-FORM-COMP	96.1723	96.5464	96.3590	93.1195	93.4816	93.3002	0
FORM	96.1723	96.5464	96.3590	93.1195	93.4816	93.3002	3
PCASE	96.4012	96.8886	**96.6443**	93.4370	93.9095	**93.6727**	978
PREDET-FORM	96.1878	96.5619	96.3745	93.1195	93.4816	93.3002	0
SPEC-FORM	96.1410	96.5075	96.3239	93.0957	93.4505	93.2728	932

Table 7.24: Parse accuracy scores for Lexical Root annotated treebanks.

The labelled f-scores show that only 2 of the 8 lexical root features outperform the baseline (CONJ-FORM, PCASE, average increase of 0.2414%), the greatest improvement achieved again by PCASE, with an increase of 0.3725%. Of the remainder, a

further 4 maintain the BASELINE score (COMP-FORM-ANAPH, CONJ-FORM-COMP, FORM, PREDET-FORM) and 2 yield a decrease (COMP-FORM, SPEC-FORM, average decrease of -0.89215%). The largest decrease in both unlabelled and labelled f-scores is yielded by just one feature, COMP-FORM, with a drop of -0.8097% for unlabelled scores and -1.7569% for labelled.

Again here unlabelled scores show a greater number of improvements over the BASELINE, but labelled scores show the greatest average increase (unlabelled average increase of 0.1244% versus labelled average increase of 0.2414%). For the first time, absolute average decreases far outweigh average increases; unlabelled average decrease (0.4224%) is more than 3.5 times the unlabelled average increase (0.1244%), while absolute labelled average decrease (0.89215%) is more than 3.7 times the labelled average increase.

Multiple Atomic Lexical Annotations

The result of this experiment is shown in Table 7.25. Only 2 combined atomic-lexical treebanks were generated as we set out to test groups of related features, or features which commonly co-occur in f-structures. For both unlabelled and labelled f-scores, PREDET-FORM_PREDET-TYPE performs best; unlabelled scores show this combination outperforms the BASELINE by 0.0155%, and labelled scores show it maintaining the BASELINE scores. SPEC-FORM_SPEC-TYPE yields a decrease for both unlabelled and labelled scores, of -0.0667% and 0.0595% respectively.

	unlabelled			labelled			occ
	precision	recall	fscore	precision	recall	fscore	#
BASELINE	96.1723	96.5464	96.3590	93.1195	93.4816	93.3002	0
PREDET-FORM_PREDET-TYPE	96.1878	96.5619	**96.3745**	93.1195	93.4816	93.3002	0
SPEC-FORM_SPEC-TYPE	96.1243	96.4608	**96.2923**	93.0781	93.4039	**93.2407**	1854

Table 7.25: Parse accuracy scores for Multiple Atomic Lexical annotated treebanks.

This combination of results was most unexpected; if we consider the number of occurrences of features in the reference data, for PREDET-FORM_PREDET-TYPE there

were no instances of annotations in the reference data. Examination of the annotated treebank shows that only 8 annotations were present in the treebank. Although we were not aiming to identify any features in the test sentences, features in the training data have shown their influence over the parser's choice of fragments, improving the parser's chunking performance, yet having no impact on parse accuracy.

Although SPEC-FORM_SPEC-TYPE has a very high number of features, it scores lower than the BASELINE for both unlabelled and labelled evaluation. Furthermore, it performs slightly worse than the average scores of the features individually. The average unlabelled score for SPEC-FORM and SPEC-TYPE is 96.29255, 0.00025% more than the score for their combined use (96.2923). The average labelled score for SPEC-FORM and SPEC-TYPE is 93.2412, 0.0005% more than the score for their combined use (93.2407). Although this result conflicts with trends we have seen elsewhere (for most other combined groups of features, a high number of occurrences yielded improved f-scores, and a higher score than the average of individual scores) the differences in scores here are so small as to be insignificant.

7.2.4 French: Discussion

We summarise the average increases and decreases for unlabelled and labelled scores for each of the twelve groups of experiments in Table 7.26 for ease of reference during this discussion. The greatest increases and decreases are highlighted in bold. Where there was no average increase or decrease, we indicate this was not applicable.

Looking first at scores for unlabelled evaluation, we see that 11 of the 12 experiments shows some improvement over the BASELINE, with average increases ranging from 0.0155% (MULTIPLE ATOMIC LEXICAL) to 0.1244% (LEXICAL ROOT). The only experiment which did not achieve an average increase was TYPE 1 MULTIPLE FUNCTIONS. All experiments show some unlabelled average decrease, ranging from -0.0345% (LEXICAL PRETERMINAL) to -0.4224% (LEXICAL ROOT). Calculation of the overall average unlabelled increase (0.0641%) and the absolute average unlabelled decrease (-0.1452%) shows that the GF-DOP model's overall chunking

	unlabelled (%)		labelled (%)	
	average increase	average decrease	average increase	average decrease
TYPE 1 SINGLE FUNCTIONS	0.0676	0.0346	0.1075	0.0526
TYPE 2 SINGLE FUNCTIONS	0.0738	0.0346	0.0849	0.041
TYPE 3 SINGLE FUNCTIONS	0.08	0.0346	0.0797	0.041
TYPE 1 MULTIPLE FUNCTIONS	n/a	0.1685	n/a	0.2614
TYPE 2 MULTIPLE FUNCTIONS	0.087	0.1162	**0.3094**	0.0773
TYPE 3 MULTIPLE FUNCTIONS	0.0971	0.1398	0.0844	0.0776
ATOMIC PRETERMINAL	0.0876	0.051	0.1185	0.0299
ATOMIC ROOT	0.0274	0.0376	0.0714	0.0488
MULTIPLE ATOMIC	0.0584	0.0808	0.1388	0.112
LEXICAL PRETERMINAL	0.0507	0.0345	0.1211	0.0499
LEXICAL ROOT	**0.1244**	**0.4224**	0.2414	**0.8921**
MULTIPLE ATOMIC LEXICAL	0.0155	0.0667	n/a	0.0595

Table 7.26: Summary of average increases and decreases for unlabelled and labelled scores for each annotation type.

performance has not improved for French.

A pattern which was frequently observed for English was that the average increase was generally larger than the average decrease. For the unlabelled scores summarised in Table 7.26, we see that 5 times out of 12 the average increase is greater than the average decrease. Once there is no average increase (TYPE 1 MULTIPLE FUNCTIONS) and 6 times the average decrease is greater than the average increase. The magnitude of some average decreases (particularly LEXICAL ROOT) compared with their average increases suggests that the GF-DOP model has not performed well for task 2.

Moving on to labelled evaluation scores, we see that only 10 out of 12 experiments have shown an average increase, ranging from 0.0714% (ATOMIC ROOT) to 0.3094% (TYPE 2 MULTIPLE FUNCTIONS). Again TYPE 1 MULTIPLE FUNCTIONS and also MULTIPLE ATOMIC LEXICAL have failed to achieve an average increase. All experiments show some labelled average decrease, ranging from 0.0299% (ATOMIC PRETERMINAL) to 0.8921% (LEXICAL ROOT). Of the 12 experiments, 9 show a greater labelled average increase than average decrease, 2 show no average increase and twice the average decrease is the larger number. Although it appears that an

overall improvement has been achieved, a closer examination of the figures shows that the average decrease (-0.1452%) is in fact larger than the average increase achieved (0.1131%). Again, the magnitude of some of the decreases (particularly LEXICAL ROOT) far outweighs the gains achieved by the increases. Thus, it is clear that the GF-DOP model has not performed as well for French as it has for English on this task.

Although we have not seen the expected overall improvement in performance from the GF-DOP model applied to a French data set, we note that some promising results were observed for several feature groups. The summary of results in Table 7.26 shows some interesting trends.

For Type 1 single functions we noted some promising results; although Type 1 single functions are likely to suffer from feature sparseness, both unlabelled and labelled increases were double the magnitude of their respective decreases. Moving on to Type 2 and Type 3 single functions, we note that while each of their unlabelled average increases show improvement over the previous type, their unlabelled average decrease remains constant. The reduction in feature sparseness has led to larger average unlabelled increases but has not yielded a reduced average unlabelled decrease, as we would have expected.

The performance of all three types of multi-functions is also a little disappointing. Unlabelled scores show an overall impairment to the parser's chunking performance; average unlabelled decreases outweigh average unlabelled increases. Labelled scores fare slightly better; TYPE 1 MULTIPLE FUNCTIONS show only an average decrease, no increase. TYPE 2 MULTIPLE FUNCTIONS show an average increase four times the average decrease; this is in fact the largest average labelled increase. This result supports our assertion that for a generous distribution of frequently occurring features, particularly for combinations of co-occurring features, we can improve the quality of c-structures generated. TYPE 3 MULTIPLE FUNCTIONS result in a larger average increase than decrease, but this average increase is notably lower than that of TYPE 2 MULTIPLE FUNCTIONS.

Of all the groups of experiments, the ATOMIC group seems to be more beneficial to the quality of the phrase-structure trees generated rather than the chunking ability of the parser. Labelled average increases are consistently larger than absolute labelled average decreases. This is in line with our *a priori* expectations; given the addition of detailed grammatical features, we hoped to see an improvement in the quality of the parses generated.

Features from the LEXICAL group have shown the greatest impact on parser performance; the greatest average unlabelled increases and decreases are yielded by LEXICAL ROOT features, as is the greatest average labelled decrease. Our initial expectation was that LEXICAL features would not exhibit such a strong influence.

7.2.5 French: Other Points of Interest

We now focus on some more particular points of interest which arose from these experiments.

In an English experiment, interesting behaviour was observed on the treebank annotated with the feature APP; although there were no features to identify when comparing output parses to the reference set, an increase in the unlabelled evaluation f-score showed some improvement in the parser's chunking performance. We note similar behaviour for two individual French features, PREDET-FORM and PREDET-TYPE (which have 4 occurrences each in their respective treebanks, and 0 in their reference sets), and one multi-annotated treebank PREDET-FORM_PREDET-TYPE (which has a total of 8 features in the annotated treebank and again 0 in the reference set). Each of these three experiments has shown an improvement in unlabelled evaluation scores; this suggests that the annotation of treebanks with even a few features has a striking influence over the parses output. A small number of features show enough impact on the fragment probability distribution to alter the fragments chosen by the parser, and transform the output parse.

Although our discussion of results in section 7.2.4 concludes that the GF-DOP model has not been shown to perform sufficiently well overall, on task 1 or task 2

for French, we suggest two reasons for this shortcoming.

Firstly, we suggest that our experiments have been negatively affected by data sparseness. As mentioned in section 7.1.5, the data set used in these experiments was very limited: only 930 sentences for French, incorporating 79 features. While data sparseness is indeed an issue for English, the French data set has more features distributed over fewer sentences; necessarily, these features have fewer occurrences each. Furthermore, these experiments allow 840 sentences for training and reserve 90 for testing and evaluation (the reference set). This is a particularly small data set for statistical work and data sparseness is inevitable. Furthermore, longer average test sentences are more likely to lead to reduced f-scores.

Secondly, the French BASELINE f-scores we compare the GF-DOP model's performance to are very high to begin with (unlabelled 96.359, labelled 93.3002). These scores are even higher than the BASELINE scores achieved for English: unlabelled 96.245, labelled 92.863. Given such high BASELINE figures, it is difficult to yield a significant improvement.

7.3 Summary

Some interesting trends were observed in this chapter. Experiments on the English data set achieved improvements in both chunking performance and labeling accuracy. These increases indicate an improvement to overall parse accuracy. Furthermore, average f-score increases were almost always greater than absolute average f-score decreases; that is, the benefits gained through the incorporation of features outweigh any losses incurred. In addition, for English, we identified treebanks with multiple function annotations as the features which produce the best results. We conclude that the GF-DOP model has succeeded at this task for English.

Experiments on the French data set showed quite different results; we did not see an overall improvement in chunking performance, although we note that we started with a higher baseline for French than for English. Although English average

increases were generally higher than average decreases, this was clearly not the case for French. Particularly for labelled scores, average decreases were generally larger than average increases. We note that atomic features have shown the best performance for chunking, while lexicals have achieved the best overall scores for labelling accuracy. However, the GF-DOP model has not performed as well for this task for French as it did for English.

One further point we observe for these experiments is the sizeable increase in score between Type 1 Lexicalised Duplicate Functions and Type 2 Duplicate Functions (for both singly and multiply annotated treebanks). Type 1 functions have a very high number of types relative to the number of feature tokens present. The reduction in the number of types as we move to Type 2 functions shows a dramatic increase in f-score. Clearly, type-token ratio has a significant impact on the parser's performance. We would like to scale these experiments up; by making use of a data set with a high number of feature occurrences and a lower type-token ratio, we feel that feature sparseness would show less impact on Type 1 functions, enabling the parser to achieve higher scores for this annotation type.

Chapter 8

Comparison: languages and task performance

This chapter presents an overview of the GF-DOP model's performance. As illustrated in Figure 8.1, we compare and contrast the GF-DOP model's performance for both English and French, before considering the GF-DOP model's competence in the two tasks set (feature detection accuracy and parse accuracy) when compared to another approach to these tasks, taken by (Chrupała and van Genabith, 2006).

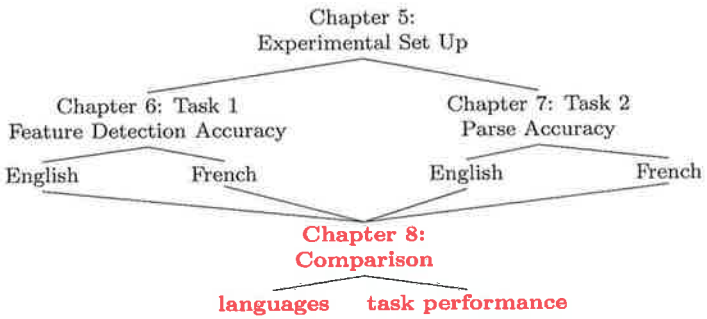


Figure 8.1: Illustration of relationships between Chapters 5, 6, 7 and 8.

8.1 English vs. French

Although our discussions in sections 6.1.4 and 6.2.4 conclude that the GF-DOP model performs reasonably well at task one (feature detection accuracy) for both English and French, and our discussions in sections 7.1.4 and 7.2.4 conclude that the GF-DOP model performs better for English than for French at task two (parse accuracy), we now concentrate our attention on some interesting points of comparison.

We begin with a juxtaposition of the starting points for each language. For English, we trained on eight training sets 890 sentences, producing average unlabelled baseline f-scores of 96.245 and labelled baseline f-scores of 92.863. For French, we trained on eight sets of 840 sentences, producing average unlabelled baseline f-scores of 96.359 and labelled baseline f-scores of 93.3002. As a result of fewer training sentences, we might have expected slightly lower baseline scores for French. However, as stated in section 5.1, the average number of words per English sentence is 8.54, while the average number of words per French sentence is 9.87. Upon scaling the number of sentences by the average sentence length, we see that there are approximately 7600 words in each English training set and 8921 in each French training set. This difference probably explains why the French baselines are higher than the English baselines even though there are fewer training sentences. However, we must appreciate that the average test sentence length was longer for French than for English, thus lower f-scores are more probable. This conflicting evidence makes it difficult to deduce conclusively why the parser achieves higher overall scores for French. Although such high baseline scores are a credit to the Tree-DOP model, the GF-DOP model must score very highly to show any positive impact of feature annotations; it is quite difficult to outperform the baseline.

Now we compare the impact of features on each language, beginning with TYPE 1 LEXICALISED DUPLICATE FUNCTION annotations. Comparison of scores for this annotation type shows that ADJUNCT is universally the most 'helpful' feature, yielding

the greatest unlabelled increase for both English and French, the greatest labelled increase for English and the second highest labelled increase for French. Knowledge and presence of 'helpful' features may be useful in applications which attempt to decompose sentences into smaller, more manageable chunks: for example in order to apply machine translation to simpler constituents, such as that of Mellebeek et al. (2005).

However, some features which perform particularly well for one language are considerably weaker for the other; for example, COMPOUND yields the highest labelled increase for French, and also outperforms the unlabelled baseline, but yields lower unlabelled and labelled scores compared to the English baseline; this trend might be expected given the differing linguistic forms of compounds in English and French. Similarly, SUBJ consistently underperformed for English, while improving unlabelled scores for French. Conversely, OBL shows some overall improvements for English, while yielding the largest overall decreases for French.

Examination of feature detection scores for TYPE 1 LEXICALISED DUPLICATION FUNCTION annotations show considerably higher scores for French than for English (for commonly occurring features). The following features score higher for French than for English:

- ADJUNCT scores 19.24% higher
- COMPOUND scores 3.66% higher
- OBJ scores 15.86% higher
- OBL scores 23.26% higher
- SUBJ scores 16.26% higher

We note that XCOMP is the only commonly occurring feature whose detection score is higher for English than for French (by 2.8%); however, XCOMP shows an improvement in both unlabelled and labelled scores for French, but only unlabelled scores for English.

For TYPE 2 DUPLICATE FUNCTION annotations we note a slight alteration of rank in features; ADJUNCT is now the second most 'helpful' feature overall, yielding the second highest improvement for both unlabelled and labelled scores in English, the second highest improvement for labelled French scores and the highest improvement for unlabelled French scores. OBJ is now the most 'useful' feature overall, being ranked first for unlabelled and labelled scores in English, first for labelled scores in French and second for unlabelled scores in French. We note that while SUBJ has again been the lowest scoring feature for English, it continues to outperform the French BASELINE scores.

Feature detection scores for TYPE 2 DUPLICATE FUNCTIONS show a much narrower, difference than for TYPE 1 LEXICALISED DUPLICATE FUNCTIONS. For most commonly occurring features an evenly spread difference of less than 4.5% is observed; OBL shows an exceptional difference of 23.59% between scores (the higher score was yielded by the French experiment). Similar peaks and troughs, are noted for TYPE 3 MINIMAL FUNCTION annotations.

Comparison of MULTIPLE FUNCTION annotations shows several large differences between languages; for TYPE 1 English experiments all combinations of functions outperformed the baseline. For TYPE 1 French experiments no combination outperformed the baseline. We note much higher numbers of features in each of the French treebanks, and also higher French feature detection scores (in the range 72.4–74.2 for French, 56.4–57.7 for English).

TYPE 2 DUPLICATE MULTIPLE FUNCTION features show considerable improvement for French; three of the four function combinations outperform the BASELINE, with only FRE5 lagging behind, again underperforming when compared to the unlabelled BASELINE, but managing to show a small increase over the labelled BASELINE. English feature detection scores show a much larger actual increase (approximately 27–28%) than French (approximately 12–15%). TYPE 3 MULTIPLE FUNCTION features show the same trends as TYPE 2 MULTIPLE FUNCTION, although English feature detection scores each show actual increases of just over 1% (compared to

TYPE 2) while French features show actual increases of only 0.3–0.5%.

Although MULTIPLE FUNCTION annotations were seen to have the greatest impact on English parse accuracy scores, LEXICAL features appear to show the most influence on French scores, with very high feature detection scores for this group (from 95.7–96 for LEXICAL PRETERMINALS and 78.1–88.8 for LEXICAL ROOTS with a good feature distribution) compared to their English equivalents (92.1 for the single LEXICAL PRETERMINAL feature and 61.4–85.5 for LEXICAL ROOTS). This was an unexpected result from the smallest group of features (only 8 English and 12 French features).

As stated in section 7.1.4, we had expected good performance from ATOMIC features as they form the bulk of the information in the f-structure and provide a lot of grammatical and structural detail. However, we did not quite achieve the large increases we hoped for; scores for French show only slightly better parse accuracy performance than for English. Feature detection scores for ATOMIC features fare better, showing results which are comparable with LEXICAL and FUNCTIONAL annotations; English scores range from 88–95.5 for ATOMIC PRETERMINALS and 81.1–92.2 for ATOMIC ROOTS for a total of 33 features. French scores range from 92.4–96.4 for ATOMIC PRETERMINALS and 82.7–94.4 for ATOMIC ROOTS for a total of 39 features.

From a first glance at this comparison, the GF-DOP model appears to perform best overall for English, yielding satisfactory overall increases on parse accuracy and competently high feature-detection scores. This consistent performance on tasks one and two shows solid support for the GF-DOP hypothesis.

Experiments on the French data set yielded some increases in parse accuracy although the benefit of these appears to be negated by the larger average decreases in scores. Performance at feature detection accuracy restores our confidence in the GF-DOP model when applied to a French data set, yielding generally higher f-scores than English; this improved performance on task two shows support for the hypothesis that the GF-DOP model can accurately learn grammatical features.

8.2 Comparing the GF-DOP Model's Task Performance

We have seen from the experiments described in Chapters 6 and 7 that the GF-DOP hypothesis holds true; the GF-DOP model produces more accurate phrase-structure trees than the Tree-DOP model. In addition to this, the GF-DOP model can learn grammatical features accurately. Now we turn our focus to comparing the GF-DOP model's performance on the two tasks described to recent work by Chrupała and van Genabith (2006).

Chrupała and van Genabith (2006) describe a variety of parsing experiments carried out using Bikel (2002)'s parser on the Cast3LB treebank, a Spanish treebank containing around 3,500 trees annotated with comprehensive grammatical functions. 17 simple format labels are described; that is, the functions indicate the function fulfilled only, but there is no indication as to the dominating predicate. Chrupała and van Genabith (2006) train the parser on 80% of the Cast3LB data, reserving 10% for development and 10% for testing. The parser is configured to produce both annotated parses and plain parses, with no functional labels.

Their experiments most closely resemble the experiment we carried out on a treebank annotated with all functions (17 for English, 13 for French), using our simplest style of function annotations, TYPE 3 MINIMAL FUNCTIONS. Our function detection scores are calculated based on the parser's output with all syntactic categories removed; our labelled scores are calculated based on the parser's output with all functional information removed.

8.2.1 Task 1: Accurately Identifying Features

Chrupała and van Genabith (2006) run a baseline experiment which parses sentences with Bikel (2002)'s parser trained on an annotated treebank; functions are output as part of the parse. They then run a comparative experiment which removes all func-

tions assigned by the parser, and re-annotate using machine learning techniques.[1] They evaluate these experiments for function accuracy only. The scores achieved are comparable with our feature detection accuracy scores.

The score for function identification achieved by Bikel (2002)'s parser alone is 59.93. Chrupała and van Genabith (2006) find that using machine learning techniques to annotate parse trees in a preprocessing step outperforms this figure significantly; they score 66.67 using this approach. We compare this to our function detection scores for the treebanks annotated with all functions using Type 3 annotations; we score (English) 85.5652 and (French) 87.3167. It is difficult to make a direct comparison between these scores as they have been derived from different parsers trained on different corpora in different languages. However we note that our experiment, which relies solely on the parser to identify functions correctly, achieves a proportionally higher score than either of the two other experiments.

8.2.2 Task 2: Improving the Quality of Parses Produced

Next we compare the baseline scores achieved by Bikel (2002)'s parser trained on an annotated treebank to our parser trained on an annotated treebank. The f-score reported in Chrupała and van Genabith (2006) is 83.96. The scores for our experiment are (English) 93.3495 and (French) 93.4141. It is difficult to compare the two systems directly, as stated in section 8.2.1, although given that Bikel (2002)'s parser was trained on more than three times the data our parser was trained on, we are content that our system has shown at least some improvement over our baseline score (English 92.863, French 93.3002).

[1] (Chrupała and van Genabith, 2006) test three machine learning techniques: TiMBL (Daelemans et al., 2004) for Memory-Based Learning, the MaxEnt Toolkit (Le, 2004) for Maximum Entropy and LIBSVM (Chang and Lin, 2001) for Support Vector Machines. The highest scores were achieved by the latter approach and these are the scores we use for comparison.

8.3 Summary

Having discussed our experiments' results in Chapters 6 and 7, we consider the GF-DOP model's overall performance by examining how the model operates in English and French, comparatively and contrastively. According to these experiments, the model appears to fare slightly better for English than for French. Subsequently, we compared the GF-DOP model's performance to that of other approaches to improving parse quality and feature identification; although it is difficult to draw a direct comparison, we find that our model has performed more than adequately when compared to another model. Finally, we conclude that the GF-DOP model has shown satisfactory performance at both feature detection accuracy and parse accuracy tasks. We are content that we have seen sufficient evident to support the GF-DOP hypothesis.

Chapter 9

Conclusions and Future Work

9.1 Conclusions

In this thesis we have presented the GF-DOP model; despite being an approximation of the LFG-DOP model, its practical implementation is based on the Tree-DOP model. We describe each of these two models, discuss some approaches to various aspects of their implementation and consider some of the theoretical and practical issues which affect each of the models.

Having examined the Tree-DOP and LFG-DOP models in some depth, we proposed a new model which draws upon each of the earlier models' strengths, while managing to avoid the practical, implementational difficulties which arise. Following a detailed description of the GF-DOP model, we propose a hypothesis which states that through the incorporation of grammatical functions and features, the GF-DOP model can accurately learn grammatical features, and apply this acquired knowledge to model language better, producing more accurate and more informative phrase-structure trees than the Tree-DOP model.

An empirical investigation of the GF-DOP model, and the GF-DOP Hypothesis on the HomeCentre corpus shows some encouraging results, which are summarised below:

- GF-DOP improves parse accuracy over Tree-DOP;

- the GF-DOP model performs well at the feature detection accuracy task for both English and French, thus supporting the hypothesis that GF-DOP should perform well in learning grammatical features;
- the GF-DOP model performs well at overall parse accuracy for English, supporting the hypothesis that the GF-DOP model can improve the quality of phrase-structure trees produced;
- the GF-DOP model does not perform quite so well at the parse accuracy task for French. Any improvements in average score increases noted for French are negated by larger average decreases;
- GF-DOP models our English data better than our French data;
- overall, we have seen sufficient evidence to support the GF-DOP Hypothesis. We conclude that the GF-DOP model has shown it can accurately learn grammatical features and employ this knowledge to improve overall parse accuracy.

9.2 Future Work

From our evaluation of the GF-DOP model, we note some points which merit further investigation. The first point is that while the GF-DOP Hypothesis was shown to hold true (on both tasks) for English, for French the evidence was not so conclusive. Although the GF-DOP model is language-independent, the model did not achieve the same success at overall parse accuracy for French as for English. In addition, data sparseness has clearly had a significant impact on experiment results; this is most evident from the large differences in scores noted between Type 1 Lexicalised Duplicate Functions and Type 2 Duplicate Functions experiments, where the ratio of types to tokens is often very high. To this end, we propose further investigation of the GF-DOP model on (a) more uniform data sets, (b) larger data sets, preferably with minimum thresholds of feature occurrences, (c) and data sets in other language pairs.

Some additional experiments which we feel might yield interesting results, but were outside the scope of the current work, include a more detailed investigation of subcategorisation frames. Although some of the highest scores achieved were obtained as a result of treebanks annotated with all functions and the most frequently occurring functions, we expect that experiments including only those functions which are governed by predicates would yield positive results; we anticipate that by only using features which are essential to the training set, the parser will learn crucial features, and not be distracted by less important elements.

Data-Oriented Translation (DOT), the statistical approach to machine translation based on DOP, is presented by (Poutsma, 2000), (Hearne, 2005). Linked source and target language subtree pairs are composed to form bilingual derivations for an input sentence; translation is achieved through synchronous parsing. We feel that the GF-DOP model applied to translation, GF-DOT, is likely to improve over the DOT model. Given that the incorporation of features has been shown to improve monolingual parse accuracy, surely more accurate bilingual parsing will produce similar improvement in translation quality. Indeed, we expect that the incorporation of grammatical features is likely to show a greater improvement in translation than in parsing, due to the minimal occurrence of features across languages, e.g. a first person singular subject in the source language is very likely to translate as a first person singular subject in the target language.

Bibliography

Abney, S. (1997). Stochastic Attribute-Value Grammars. *Computational Linguistics*, **23**(4):597–618.

Bikel, D. (2002). Design of a Multi-Lingual, Parallel-Processing Statistical Parsing Engine. In *Proceedings of Human Language Technology Conference*, San Diego, CA.

Bod, R. (1992). A Computational Model of Language Performance: Data Oriented Parsing. In *Proceedings of the 15th [sic] International Conference on Computational Linguistics (COLING'92)*, pages 855–859, Nantes, France.

Bod, R. (2000). Parsing with the Shortest Derivation. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING'00)*, pages 69–75, Saarbruecken, Germany.

Bod, R. (2001). What is the Minimal Set of Fragments that Achieves Maximal Parse Accuracy? In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics (ACL'01) and the 10th Conference of the European Chapter of the Association for Computational Linguistics (EACL'01)*, pages 66–73, Toulouse, France.

Bod, R. (2003a). An Efficient Implementation of a New DOP Model. In *Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics (EACL'03)*, pages 19–26, Budapest, Hungary.

Bod, R. (2003b). Extracting Stochastic Grammars from Treebanks. In Abeillé, A.,

editor, *Treebanks: Building and Using Parallel Corpora*, pages 333–350. Kluwer Academic Publishers, Dordrecht, The Netherlands.

Bod, R. and Kaplan, R. (1998). A Probabilistic Corpus-Driven Model for Lexical-Functional Analysis. In *Proceedings of the 17th International Conference on Computational Linguistics and 36th Conference of the Association for Computational Linguistics*, pages 145–151, Montreal, Canada.

Bod, R. and Kaplan, R. (2003). A DOP model for Lexical-Functional Grammar. In Bod, R., Scha, R., and Sima'an, K., editors, *Data-Oriented Parsing*, pages 211–232. Stanford CA.: CSLI Publications.

Butt, M., King, T. H., Nino, M.-E., and Segond, F. (1999). *A Grammar Writer's Cookbook*. Stanford CA.: CSLI Publications.

Chang, C.-C. and Lin, C.-J. (2001). LIBSVL: a library for support vector machines. Technical report, National Taiwan University.

Chappelier, J.-C. and Rajman, M. (2003). Parsing DOP with Monte-Carlo Techniques. In Bod, R., Scha, R., and Sima'an, K., editors, *Data-Oriented Parsing*, pages 83–106. Stanford CA.: CSLI Publications.

Chrupała, G. and van Genabith, J. (2006). Using Machine-Learning to Assign Function Labels to Parser Output for Spanish. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, pages 136–143, Sydney, Australia. Association for Computational Linguistics.

Daelemans, W., Zavrel, J., van der Sloot, K., and van den Bosch, A. (2004). TiMBL: Tilburg Memory Based Learner. Technical Report version 5.1, Reference Guide, Tilburg University, University of Antwerp.

Goodman, J. (2003). Efficient Parsing of DOP with PCFG-Reductions. In Bod, R., Scha, R., and Sima'an, K., editors, *Data-Oriented Parsing*, pages 125–146. Stanford CA.: CSLI Publications.

Hearne, M. (2005). *Data-Oriented Models of Parsing and Translation.* PhD thesis, Dublin City University, Dublin, Ireland.

Johnson, M. (2002). The DOP estimation method is biased and inconsistent. *Computational Linguistics*, **28**(1):71–76.

Kaplan, R. and Bresnan, J. (1982). Lexical Functional Grammar, a Formal System for Grammatical Representation. In Bresnan, J., editor, *The Mental Representation of Grammatical Relations*, pages 173–281. MIT Press, Cambridge, MA.

Le, Z. (2004). Maximum Entropy Modeling Toolkit for Python and C++. Technical report, The University of Edinburgh.

Mellebeek, B., Khasin, A., Owczarzak, K., van Genabith, J., and Way, A. (2005). Improving Online Machine Translation Systems. In *Proceedings of 10th Machine Translation Summit*, pages 290–297, Phuket, Thailand.

Poutsma, A. (2000). Data-Oriented Translation: Using the Data-Oriented Parsing framework for Machine Translation. Master's thesis, University of Amsterdam, The Netherlands.

Scha, R. (1990). Language Theory and Language Technology: Competence and Performance. *Computertoepassingen in de Neerlandistiek*, pages 7–22.

Sima'an, K. (1995a). An optimized algorithm for Data Oriented Parsing. In *Proceedings of International Conference on Recent Advances in Natural Language Processing*, Tzigov Chark, Bulgaria.

Sima'an, K. (1995b). Computational Complexity of Probabilistic Disambiguation by means of Tree-Grammars. In *Proceedings of the 15th International Conference on Computational Linguistics (COLING'96)*, pages 1175–1180, Copenhagen, Denmark.

Younger, D. (1967). Recognition and parsing of context-free languages in time n^3. *Information Control*, **10**(2):189–208.

Appendix A

English Tables of Results

We include a note here on some of the parse accuracy scores; where there are no features in the reference set, and the parser has produced parses with no features, the experiment f-score is 100%. That is, the parser has not proposed any parses incorporating features, which would be incorrect. An example of this can be seen in section A.1.1, for the feature APP.

A.1 Functional Annotations

A.1.1 Type 1 Lexicalised Duplicate Function Annotations

	unlabelled			labelled			features			occ
	precision	recall	fscore	precision	recall	fscore	precision	recall	fscore	#
BASELINE	96.0913	96.3991	96.2450	92.7148	93.0117	92.8630	0.0000	0.0000	0.0000	0
ADJUNCT	96.1427	96.5253	96.3336	92.7204	93.0894	92.9045	58.1465	46.6427	51.7631	834
APP	96.1010	96.4088	96.2547	92.7148	93.0117	92.8630	100.0000	100.0000	100.0000	0
COMP	96.0913	96.3991	96.2450	92.7148	93.0117	92.8630	77.2727	77.2727	77.2727	22
COMP-EX	96.0913	96.3991	96.2450	92.7148	93.0117	92.8630	100.0000	100.0000	100.0000	0
COMPOUND	96.0801	96.3506	96.2152	92.7120	92.9729	92.8423	78.6822	66.3399	71.9858	306
OBJ	96.1141	96.5059	96.3096	92.8371	93.2156	93.0260	63.4615	59.5361	61.4362	776
OBJ2	96.0910	96.3894	96.2399	92.7141	93.0020	92.8578	100.0000	100.0000	100.0000	0
OBL	96.1107	96.4185	96.2644	92.7825	93.0797	92.9309	75.0000	56.2500	64.2857	16
OBL-AGT	96.0913	96.3991	96.2450	92.7051	93.0020	92.8533	100.0000	100.0000	100.0000	0
OBL-COMP	96.0913	96.3991	96.2450	92.7148	93.0117	92.8630	100.0000	100.0000	100.0000	3
PRON-INT	96.0913	96.3991	96.2450	92.7148	93.0117	92.8630	100.0000	91.6667	95.6522	24
PRON-REL	96.0913	96.3991	96.2450	92.7148	93.0117	92.8630	100.0000	75.0000	85.7143	4
SPEC	96.1390	96.4282	96.2834	92.8005	93.0797	92.9399	85.5072	83.0986	84.2857	71
SUBJ	96.0344	96.3700	96.2019	92.5041	92.8273	92.6654	55.8480	45.6938	50.2632	418
TOPIC-INT	96.0913	96.3991	96.2450	92.7148	93.0117	92.8630	100.0000	100.0000	100.0000	25
TOPIC-REL	96.0913	96.3991	96.2450	92.7148	93.0117	92.8630	100.0000	66.6667	80.0000	3
XCOMP	96.1594	96.4768	96.3178	92.6768	92.9826	92.8295	75.0000	58.5366	65.7534	123

A.1.2 Type 2 Duplicate Function Annotations

	unlabelled			labelled			features			occ
	precision	recall	fscore	precision	recall	fscore	precision	recall	fscore	#
BASELINE	96.0913	96.3991	96.2450	92.7148	93.0117	92.8630	0.0000	0.0000	0.0000	0
ADJUNCT	96.2365	96.5447	96.3903	92.9373	93.2350	93.0859	87.8173	82.9736	85.3268	834
APP	96.1010	96.4088	96.2547	92.7148	93.0117	92.8630	100.0000	100.0000	100.0000	0
COMP	96.0913	96.3991	96.2450	92.7148	93.0117	92.8630	77.2727	77.2727	77.2727	22
COMP-EX	96.0913	96.3991	96.2450	92.7148	93.0117	92.8630	100.0000	100.0000	100.0000	0
COMPOUND	96.0913	96.3991	96.2450	92.6471	92.9438	92.7952	87.2852	81.6720	84.3854	311
OBJ	96.2848	96.5932	96.4388	93.0244	93.3223	93.1731	88.4058	86.4691	87.4267	776
OBJ2	96.0910	96.3894	96.2399	92.7141	93.0020	92.8578	100.0000	100.0000	100.0000	0
OBL	96.0913	96.3991	96.2450	92.7148	93.0117	92.8630	75.0000	56.2500	64.2857	16
OBL-AGT	96.0913	96.3991	96.2450	92.7051	93.0020	92.8533	100.0000	100.0000	100.0000	0
OBL-COMP	96.0913	96.3991	96.2450	92.7148	93.0117	92.8630	100.0000	100.0000	100.0000	3
PRON-INT	96.0913	96.3991	96.2450	92.7148	93.0117	92.8630	100.0000	91.6667	95.6522	24
PRON-REL	96.0913	96.3991	96.2450	92.7148	93.0117	92.8630	100.0000	75.0000	85.7143	4
SPEC	96.0720	96.3797	96.2256	92.7148	93.0117	92.8630	100.0000	97.1831	98.5714	71
SUBJ	96.0542	96.3991	96.2263	92.5532	92.8856	92.7191	85.1175	77.2512	80.9938	422
TOPIC-INT	96.0913	96.3991	96.2450	92.7148	93.0117	92.8630	100.0000	100.0000	100.0000	25
TOPIC-REL	96.0913	96.3991	96.2450	92.7148	93.0117	92.8630	100.0000	66.6667	80.0000	3
XCOMP	96.1014	96.4185	96.2597	92.6478	92.9535	92.8004	77.5510	61.7886	68.7783	123

A.1.3 Type 3 Minimal Function Annotations

	unlabelled			labelled			features			occ
	precision	recall	fscore	precision	recall	fscore	precision	recall	fscore	#
BASELINE	96.0913	96.3991	96.2450	92.7148	93.0117	92.8630	0.0000	0.0000	0.0000	0
ADJUNCT	96.2465	96.5641	96.4050	93.0251	93.3320	93.1783	88.5714	85.0374	86.7684	802
APP	96.1010	96.4088	96.2547	92.7148	93.0117	92.8630	100.0000	100.0000	100.0000	0
COMP	96.0913	96.3991	96.2450	92.7148	93.0117	92.8630	77.2727	77.2727	77.2727	22
COMP-EX	96.0913	96.3991	96.2450	92.7148	93.0117	92.8630	100.0000	100.0000	100.0000	0
COMPOUND	96.1107	96.4185	96.2644	92.6471	92.9438	92.7952	90.3475	85.4015	87.8049	274
OBJ	96.2945	96.6029	96.4485	93.0244	93.3223	93.1731	88.4058	86.4691	87.4267	776
OBJ2	96.0910	96.3894	96.2399	92.7141	93.0020	92.8578	100.0000	100.0000	100.0000	0
OBL	96.0913	96.3991	96.2450	92.7148	93.0117	92.8630	75.0000	56.2500	64.2857	16
OBL-AGT	96.0913	96.3991	96.2450	92.7051	93.0020	92.8533	100.0000	100.0000	100.0000	0
OBL-COMP	96.0913	96.3991	96.2450	92.7148	93.0117	92.8630	100.0000	100.0000	100.0000	3
PRON-INT	96.0913	96.3991	96.2450	92.7148	93.0117	92.8630	100.0000	91.6667	95.6522	24
PRON-REL	96.0913	96.3991	96.2450	92.7148	93.0117	92.8630	100.0000	75.0000	85.7143	4
SPEC	96.0720	96.3797	96.2256	92.7148	93.0117	92.8630	100.0000	97.1831	98.5714	71
SUBJ	96.0542	96.3991	96.2263	92.5629	92.8953	92.7288	87.1711	80.0604	83.4646	331
TOPIC-INT	96.0913	96.3991	96.2450	92.7148	93.0117	92.8630	100.0000	100.0000	100.0000	25
TOPIC-REL	96.0913	96.3991	96.2450	92.7148	93.0117	92.8630	100.0000	66.6667	80.0000	3
XCOMP	96.1014	96.4185	96.2597	92.6478	92.9535	92.8004	77.5510	61.7886	68.7783	123

A.1.4 Type 1 Lexicalised Duplicate Multiple Function Annotations

	unlabelled			labelled			features			occ
	precision	recall	fscore	precision	recall	fscore	precision	recall	fscore	#
BASELINE	96.0913	96.3991	96.2450	92.7148	93.0117	92.8630	0.0000	0.0000	0.0000	0
ALL	96.1814	96.5641	96.3724	92.8171	93.1865	93.0014	63.4943	51.7761	57.0396	2590
TOP5	96.2959	96.6418	96.4685	92.9014	93.2350	93.0679	62.6428	51.4064	56.4711	2453
SUBJ_OBJ	96.2669	96.6126	96.4395	92.8820	93.2156	93.0485	60.8574	54.8739	57.7110	1190

A.1.5 Type 2 Duplicate Multiple Function Annotations

	unlabelled			labelled			features			occ
	precision	recall	fscore	precision	recall	fscore	precision	recall	fscore	#
BASELINE	96.0913	96.3991	96.2450	92.7148	93.0117	92.8630	0.0000	0.0000	0.0000	0
ALL	96.3339	96.6612	96.4973	93.2192	93.5359	93.3773	87.7928	82.0592	84.8292	2603
TOP5	96.3257	96.6903	96.5076	93.1831	93.5359	93.3592	87.2460	81.8329	84.4528	2466
SUBJ_OBJ	96.2658	96.5835	96.4244	93.0057	93.3126	93.1589	87.1616	83.3055	85.1899	1198

A.1.6 Type 3 Minimal Multiple Function Annotations

	unlabelled			labelled			features			occ
	precision	recall	fscore	precision	recall	fscore	precision	recall	fscore	#
BASELINE	96.0913	96.3991	96.2450	92.7148	93.0117	92.8630	0.0000	0.0000	0.0000	0
ALL	96.3450	96.7097	96.5270	93.1735	93.5262	93.3495	88.7582	83.3129	85.9494	2445
TOP5	96.2788	96.6806	96.4792	93.1181	93.5067	93.3120	88.0220	83.1743	85.5295	2306
SUBJ_OBJ	96.2949	96.6126	96.4535	93.0347	93.3417	93.1880	87.8987	84.6432	86.2402	1107

A.2 Atomic Feature Annotations

A.2.1 Atomic Preterminal Annotations

	unlabelled			labelled			features			occ
	precision	recall	fscore	precision	recall	fscore	precision	recall	fscore	#
BASELINE	96.0913	96.3991	96.2450	92.7148	93.0117	92.8630	0.0000	0.0000	0.0000	0
ABBREV	96.1300	96.4379	96.2837	92.7148	93.0117	92.8630	100.0000	100.0000	100.0000	30
GRAIN	96.0337	96.3506	96.1919	92.3672	92.6720	92.5194	92.8678	91.4988	92.1782	2035
MOOD	96.0828	96.4185	96.2504	92.6589	92.9826	92.8205	90.4984	85.6932	88.0303	678
NUM	95.9687	96.3506	96.1592	92.4207	92.7885	92.6043	94.4909	92.8678	93.6723	2678
NUMBER-TYPE	96.0720	96.3797	96.2256	92.7148	93.0117	92.8630	100.0000	100.0000	100.0000	95
PERS	96.0944	96.4768	96.2852	92.9138	93.2835	93.0983	96.2170	94.8294	95.5182	2843
PRON-TYPE	96.2175	96.5350	96.3760	92.9090	93.2156	93.0620	95.2795	90.7692	92.9697	845
PROPER	96.0902	96.3700	96.2299	92.5385	92.8079	92.6730	100.0000	73.5294	84.7458	34

A.2.2 Atomic Root Annotations

	unlabelled			labelled			features			occ
	precision	recall	fscore	precision	recall	fscore	precision	recall	fscore	#
BASELINE	96.0913	96.3991	96.2450	92.7148	93.0117	92.8630	0.0000	0.0000	0.0000	0
ADEG-DIM	96.0251	96.3700	96.1973	92.6692	93.0020	92.8353	70.0000	63.6364	66.6667	11
ADEGREE	96.1501	96.4768	96.3132	92.8129	93.1282	92.9703	86.2454	87.5472	86.8914	265
ADJUNCT-TYPE	96.2655	96.5738	96.4194	92.9373	93.2350	93.0859	90.1442	80.2998	84.9377	467
ADV-TYPE	96.1018	96.4282	96.2647	92.6485	92.9632	92.8056	85.0394	77.8846	81.3049	416
ANIM	96.2268	96.5350	96.3806	92.7728	93.0700	92.9212	84.7826	77.8271	81.1561	451
ATYPE	96.1401	96.4573	96.2984	92.8219	93.1282	92.9748	87.7395	88.0769	87.9079	260
CASE	96.0847	96.4670	96.2755	92.8654	93.2350	93.0498	87.5676	84.1558	85.8278	1155
DEIXIS	96.0720	96.3797	96.2256	92.7148	93.0117	92.8630	75.0000	27.2727	40.0000	11
GEND	96.1777	96.4670	96.3221	92.8005	93.0797	92.9399	100.0000	73.5294	84.7458	34
GERUND	96.0724	96.3894	96.2306	92.8122	93.1185	92.9651	88.4956	92.5926	90.4977	108
LAYOUT-TYPE	96.0043	96.3118	96.1578	92.7051	93.0020	92.8533	91.8991	84.3902	87.9847	820
PASSIVE	96.0209	96.2632	96.1419	92.7679	93.0020	92.8848	84.9508	78.2383	81.4565	772
PERF	96.0403	96.2826	96.1613	92.7583	92.9923	92.8751	87.2456	81.5990	84.3279	788
POL	96.1204	96.4282	96.2740	92.7148	93.0117	92.8630	0.0000	0.0000	0.0000	3
PREDET-TYPE	96.0155	96.3603	96.1876	92.6015	92.9341	92.7675	40.0000	50.0000	44.4444	4
PROG	96.1084	96.3603	96.2342	92.7880	93.0312	92.9094	85.9079	79.2500	82.4447	800
PSEM	95.9780	96.3506	96.1639	92.6037	92.9632	92.7831	92.7835	87.3786	90.0000	309
PTYPE	96.0735	96.4185	96.2457	92.6499	92.9826	92.8160	91.4013	93.1818	92.2830	308
SPEC-TYPE	96.1312	96.4670	96.2988	92.8136	93.1379	92.9755	90.2045	83.0525	86.4809	1009
STMT-TYPE	96.1096	96.3894	96.2493	92.7998	93.0700	92.9347	87.7256	82.4661	85.0146	884
TEMPORAL	96.1587	96.4573	96.3078	92.8108	93.0991	92.9547	0.0000	0.0000	0.0000	1
TENSE	96.0824	96.4088	96.2453	92.7162	93.0312	92.8734	74.1117	66.9725	70.3614	218
TIME	96.1490	96.4476	96.2981	92.8108	93.0991	92.9547	0.0000	0.0000	0.0000	1
TYPE	96.0724	96.3894	96.2306	92.8122	93.1185	92.9651	88.4956	92.5926	90.4977	108
VTYPE	96.0321	96.3118	96.1717	92.7223	92.9923	92.8571	85.3061	78.7688	81.9073	796

A.2.3 Multiple Atomic Annotations

	unlabelled			labelled			features			occ
	precision	recall	fscore	precision	recall	fscore	precision	recall	fscore	#
BASELINE	96.0913	96.3991	96.2450	92.7148	93.0117	92.8630	0.0000	0.0000	0.0000	0
NUM_PERS	95.9695	96.3700	96.1693	92.5382	92.9244	92.7309	90.3162	86.4839	88.3585	4195
PERF_PROG_TENSE	96.1088	96.3700	96.2392	92.7597	93.0117	92.8855	88.0822	80.1746	83.9426	802
PERF_PROG_TENSE_PASSIVE_MOOD	96.1080	96.3506	96.2292	92.8551	93.0894	92.9721	87.7579	79.0582	83.1812	807

A.3 Lexical Feature Annotations

A.3.1 Lexical Preterminal Annotations

	unlabelled			labelled			features			occ
	precision	recall	fscore	precision	recall	fscore	precision	recall	fscore	#
BASELINE	96.0913	96.3991	96.2450	92.7148	93.0117	92.8630	0.0000	0.0000	0.0000	0
PRON-FORM	96.1494	96.4573	96.3031	92.7632	93.0603	92.9115	96.6102	88.1443	92.1833	194

A.3.2 Lexical Root Annotations

	unlabelled			labelled			features			occ
	precision	recall	fscore	precision	recall	fscore	precision	recall	fscore	#
BASELINE	96.0913	96.3991	96.2450	92.7148	93.0117	92.8630	0.0000	0.0000	0.0000	0
COMP-FORM	96.1107	96.4185	96.2644	92.6761	92.9729	92.8243	85.0467	73.3871	78.7879	124
CONJ-FORM	96.1096	96.3894	96.2493	92.6546	92.9244	92.7893	78.7879	50.3226	61.4173	155
CONJ-FORM-COMP	96.0817	96.3894	96.2353	92.7148	93.0117	92.8630	100.0000	100.0000	100.0000	3
PCASE	96.1122	96.4573	96.2845	92.8627	93.1962	93.0291	69.8171	69.6049	69.7108	329
PREDET-FORM	96.0155	96.3603	96.1876	92.6015	92.9341	92.7675	40.0000	50.0000	44.4444	4
PRT-FORM	96.0720	96.3797	96.2256	92.7148	93.0117	92.8630	55.5556	41.6667	47.6190	12
SPEC-FORM	96.1698	96.5059	96.3376	92.8426	93.1670	93.0046	87.7193	83.4725	85.5432	599

A.3.3 Multiple Atomic Lexical Annotations

	unlabelled			labelled			features			occ
	precision	recall	fscore	precision	recall	fscore	precision	recall	fscore	#
BASELINE	96.0913	96.3991	96.2450	92.7148	93.0117	92.8630	0.0000	0.0000	0.0000	0
PREDET-FORM_ PREDET-TYPE	96.0155	96.3603	96.1876	92.6015	92.9341	92.7675	40.0000	50.0000	44.4444	4
SPEC-TYPE_ SPEC-FORM	96.1505	96.4865	96.3182	92.8523	93.1767	93.0142	91.1514	83.6595	87.2449	1022

Appendix B

French Tables of Results

B.1 Functional Annotations

B.1.1 Type 1 Lexicalised Duplicate Function Annotations

	unlabelled			labelled			features			occ
	precision	recall	fscore	precision	recall	fscore	precision	recall	fscore	#
BASELINE	96.1723	96.5464	96.3590	93.1195	93.4816	93.3002	0.0000	0.0000	0.0000	0
ADJUNCT	96.3058	96.7253	96.5151	93.2853	93.6917	93.4880	73.6348	68.5465	70.9996	1259
COMP	96.1419	96.5308	96.3360	93.0818	93.4583	93.2697	54.1667	65.0000	59.0909	20
COMPOUND	96.2191	96.6008	96.4096	93.3834	93.7539	93.5683	82.5243	69.6721	75.5556	122
OBJ	96.2096	96.5464	96.3777	93.2253	93.5516	93.3882	79.2672	75.4252	77.2985	1176
OBJ2	96.1723	96.5464	96.3590	93.1272	93.4894	93.3080	94.1176	64.0000	76.1905	25
OBL	96.1187	96.5075	96.3127	93.0431	93.4194	93.2309	89.2308	85.9259	87.5472	135
OBL-AGT	96.1723	96.5464	96.3590	93.1195	93.4816	93.3002	100.0000	100.0000	100.0000	1
OBL-COMP	96.1723	96.5464	96.3590	93.1195	93.4816	93.3002	100.0000	100.0000	100.0000	0
PRON-REL	96.1723	96.5464	96.3590	93.1195	93.4816	93.3002	100.0000	100.0000	100.0000	15
SPEC	96.1723	96.5464	96.3590	93.1272	93.4894	93.3080	100.0000	100.0000	100.0000	1
SUBJ	96.2036	96.5853	96.3941	93.0580	93.4272	93.2422	66.8151	66.2252	66.5188	453
TOPIC-REL	96.1723	96.5464	96.3590	93.1195	93.4816	93.3002	100.0000	100.0000	100.0000	15
XCOMP	96.2426	96.6319	96.4369	93.1980	93.5750	93.3861	72.3404	55.7377	62.9630	61

B.1.2 Type 2 Duplicate Function Annotations

	unlabelled			labelled			features			occ
	precision	recall	fscore	precision	recall	fscore	precision	recall	fscore	#
BASELINE	96.1723	96.5464	96.3590	93.1195	93.4816	93.3002	0.0000	0.0000	0.0000	0
ADJUNCT	96.2670	96.6864	96.4762	93.2079	93.6139	93.4104	91.0095	88.9746	89.9805	1297
COMP	96.1419	96.5308	96.3360	93.0818	93.4583	93.2697	56.0000	70.0000	62.2222	20
COMPOUND	96.2185	96.5853	96.4015	93.2197	93.5750	93.3970	86.5546	84.4262	85.4772	122
OBJ	96.2799	96.6319	96.4556	93.2961	93.6372	93.4664	88.2096	85.8844	87.0315	1176
OBJ2	96.1723	96.5464	96.3590	93.1040	93.4661	93.2847	85.0000	68.0000	75.5556	25
OBL	96.1187	96.5075	96.3127	93.0353	93.4116	93.2231	89.9225	85.9259	87.8788	135
OBL-AGT	96.1723	96.5464	96.3590	93.1195	93.4816	93.3002	100.0000	100.0000	100.0000	1
OBL-COMP	96.1723	96.5464	96.3590	93.1195	93.4816	93.3002	100.0000	100.0000	100.0000	0
PRON-REL	96.1723	96.5464	96.3590	93.1195	93.4816	93.3002	100.0000	100.0000	100.0000	15
SPEC	96.1723	96.5464	96.3590	93.1272	93.4894	93.3080	100.0000	100.0000	100.0000	1
SUBJ	96.2340	96.6008	96.4171	93.1964	93.5516	93.3737	81.5678	84.9890	83.2432	453
TOPIC-REL	96.1723	96.5464	96.3590	93.1195	93.4816	93.3002	100.0000	100.0000	100.0000	15
XCOMP	96.2194	96.6086	96.4136	93.1670	93.5439	93.3551	78.7234	60.6557	68.5185	61

B.1.3 Type 3 Minimal Function Annotations

	unlabelled			labelled			features			occ
	precision	recall	fscore	precision	recall	fscore	precision	recall	fscore	#
BASELINE	96.1723	96.5464	96.3590	93.1195	93.4816	93.3002	0.0000	0.0000	0.0000	0
ADJUNCT	96.2670	96.6864	96.4762	93.1924	93.5983	93.3949	91.0556	88.9064	89.9682	1271
COMP	96.1419	96.5308	96.3360	93.0818	93.4583	93.2697	56.0000	70.0000	62.2222	20
COMPOUND	96.2185	96.5853	96.4015	93.2197	93.5750	93.3970	88.0342	84.4262	86.1925	122
OBJ	96.2799	96.6319	96.4556	93.2806	93.6217	93.4508	88.7884	86.1404	87.4443	1140
OBJ2	96.1723	96.5464	96.3590	93.1040	93.4661	93.2847	85.0000	68.0000	75.5556	25
OBL	96.1187	96.5075	96.3127	93.0353	93.4116	93.2231	89.9225	85.9259	87.8788	135
OBL-AGT	96.1723	96.5464	96.3590	93.1195	93.4816	93.3002	100.0000	100.0000	100.0000	1
OBL-COMP	96.1723	96.5464	96.3590	93.1195	93.4816	93.3002	100.0000	100.0000	100.0000	0
PRON-REL	96.1723	96.5464	96.3590	93.1195	93.4816	93.3002	100.0000	100.0000	100.0000	15
SPEC	96.1723	96.5464	96.3590	93.1272	93.4894	93.3080	100.0000	100.0000	100.0000	1
SUBJ	96.2650	96.6319	96.4481	93.1964	93.5516	93.3737	83.2061	85.1562	84.1699	384
TOPIC-REL	96.1723	96.5464	96.3590	93.1195	93.4816	93.3002	100.0000	100.0000	100.0000	15
XCOMP	96.2194	96.6086	96.4136	93.1670	93.5439	93.3551	78.7234	60.6557	68.5185	61

B.1.4 Type 1 Lexicalised Duplicate Multiple Function Annotations

	unlabelled			labelled			features			occ
	precision	recall	fscore	precision	recall	fscore	precision	recall	fscore	#
BASELINE	96.1723	96.5464	96.3590	93.1195	93.4816	93.3002	0.0000	0.0000	0.0000	0
ALL	95.9421	96.3675	96.1543	92.7050	93.1161	92.9101	77.0169	69.7789	73.2195	3256
ENG5	95.9721	96.3752	96.1733	92.7730	93.1627	92.9675	76.2230	69.0003	72.4321	3071
FRE5	95.9340	96.3519	96.1425	92.8748	93.2794	93.0767	77.0091	70.0795	73.3811	3145
SUBJ_OBJ	96.1386	96.4452	96.2917	93.0526	93.3494	93.2008	76.0618	72.5599	74.2696	1629

B.1.5 Type 2 Duplicate Multiple Function Annotations

	unlabelled			labelled			features			occ
	precision	recall	fscore	precision	recall	fscore	precision	recall	fscore	#
BASELINE	96.1723	96.5464	96.3590	93.1195	93.4816	93.3002	0.0000	0.0000	0.0000	0
ALL	96.1973	96.6164	96.4064	93.1691	93.5750	93.3716	88.5256	85.4888	86.9807	3294
ENG5	96.2518	96.6786	96.4647	93.1929	93.6061	93.3990	88.9701	86.1370	87.5306	3109
FRE5	96.0490	96.4375	96.2428	93.1128	93.4894	93.3007	88.7484	85.9881	87.3464	3183
SUBJ_OBJ	96.2951	96.6397	96.4671	93.2724	93.6061	93.4389	87.0603	85.0829	86.0602	1629

B.1.6 Type 3 Minimal Multiple Function Annotations

	unlabelled			labelled			features			occ
	precision	recall	fscore	precision	recall	fscore	precision	recall	fscore	#
BASELINE	96.1723	96.5464	96.3590	93.1195	93.4816	93.3002	0.0000	0.0000	0.0000	0
ALL	96.2435	96.6353	96.4389	93.2151	93.6139	93.4141	89.0533	85.6465	87.3167	3163
ENG5	96.2822	96.6942	96.4878	93.1919	93.5905	93.3908	89.6684	86.2660	87.9343	2978
FRE5	96.0329	96.4063	96.2192	93.0420	93.4039	93.2226	89.3733	85.9764	87.6420	3052
SUBJ_OBJ	96.2710	96.5930	96.4318	93.1933	93.5050	93.3489	87.9485	85.2362	86.5711	1524

B.2 Atomic Feature Annotations

B.2.1 Atomic Preterminal Annotations

	unlabelled			labelled			features			occ
	precision	recall	fscore	precision	recall	fscore	precision	recall	fscore	#
BASELINE	96.1723	96.5464	96.3590	93.1195	93.4816	93.3002	0.0000	0.0000	0.0000	0
GRAIN	96.1637	96.5152	96.3391	93.1644	93.5050	93.3344	96.5622	94.7886	95.6672	2341
INV	96.1723	96.5464	96.3590	93.1117	93.4739	93.2924	100.0000	100.0000	100.0000	3
MOOD	96.1965	96.5930	96.3943	93.0978	93.4816	93.2893	97.5652	92.5743	95.0042	606
NUM	96.0759	96.5541	96.3144	92.9721	93.4350	93.2030	95.2782	92.4714	93.8538	3666
NUMBER-TYPE	96.1723	96.5464	96.3590	93.1117	93.4739	93.2924	100.0000	98.8095	99.4012	84
PERS	96.0654	96.4764	96.2704	93.0757	93.4739	93.2743	96.6993	94.9580	95.8207	3332
PRON-TYPE	96.2811	96.6630	96.4717	93.2982	93.6683	93.4829	95.2135	92.4623	93.8177	796
PROPER	96.1807	96.5697	96.3748	93.1515	93.5283	93.3395	94.8276	90.1639	92.4370	244
REFL	96.3511	96.7408	96.5456	93.3297	93.7072	93.5181	97.6845	95.2045	96.4286	709

B.2.2 Atomic Root Annotations

	unlabelled			labelled			features			occ
	precision	recall	fscore	precision	recall	fscore	precision	recall	fscore	#
BASELINE	96.1723	96.5464	96.3590	93.1195	93.4816	93.3002	0.0000	0.0000	0.0000	0
ADEGREE	96.1652	96.5541	96.3593	93.0896	93.4661	93.2774	93.7500	95.5882	94.6602	204
ADEG-DIM	96.1723	96.5464	96.3590	93.1195	93.4816	93.3002	100.0000	60.0000	75.0000	5
ADJUNCT-LAYOUT	96.1723	96.5464	96.3590	93.1117	93.4739	93.2924	100.0000	100.0000	100.0000	55
ADJUNCT-TYPE	96.1697	96.4764	96.3228	93.0682	93.3650	93.2163	94.2073	87.7841	90.8824	352
ADV-TYPE	96.1711	96.5152	96.3429	93.0631	93.3961	93.2293	88.7550	80.9524	84.6743	273
APOS	96.1494	96.5308	96.3397	93.1278	93.4972	93.3121	95.4286	93.2961	94.3503	179
ATYPE	96.1649	96.5464	96.3552	93.0735	93.4428	93.2578	94.9458	94.6043	94.7748	278
CASE	96.1807	96.5697	96.3748	93.1051	93.4816	93.2930	90.8560	83.0961	86.8030	1124
CONJOINED	96.1723	96.5464	96.3590	93.1195	93.4816	93.3002	79.7297	93.6508	86.1314	63
CONJTYPE	96.2713	96.6008	96.4358	93.1938	93.5128	93.3530	89.5833	87.7551	88.6598	49
DEIXIS	96.1878	96.5619	96.3745	93.1350	93.4972	93.3157	95.6522	95.6522	95.6522	23
FOO	96.1723	96.5464	96.3590	93.1195	93.4816	93.3002	0.0000	0.0000	0.0000	2
GEND	96.1416	96.5230	96.3319	93.3292	93.6994	93.5140	86.3399	82.0614	84.1463	2018
INV	96.1723	96.5464	96.3590	93.1117	93.4739	93.2924	100.0000	100.0000	100.0000	3
LAYOUT-TYPE	96.2033	96.5775	96.3900	93.0342	93.3961	93.2148	85.9023	79.7557	82.7149	573
NE	96.1488	96.5152	96.3317	93.1034	93.4583	93.2805	93.2203	82.0896	87.3016	67
NEG	96.1723	96.5464	96.3590	93.1117	93.4739	93.2924	100.0000	100.0000	100.0000	4
PASSIVE	96.1899	96.6164	96.4027	93.0922	93.5050	93.2981	92.1127	86.0526	88.9796	760
PCASE-TYPE	96.1878	96.5619	96.3745	93.1195	93.4816	93.3002	100.0000	45.0000	62.0690	20
PERF	96.1270	96.5308	96.3285	93.0364	93.4272	93.2314	91.9685	85.0073	88.3510	687
PREDET-TYPE	96.1878	96.5619	96.3745	93.1195	93.4816	93.3002	100.0000	100.0000	100.0000	0
PREVERB-OBJ	96.1723	96.5464	96.3590	93.1195	93.4816	93.3002	0.0000	0.0000	0.0000	3
PSEM	96.2486	96.5930	96.4205	93.2414	93.5750	93.4079	95.5556	93.4783	94.5055	322
PTYPE	96.0629	96.4141	96.2382	93.0016	93.3416	93.1713	94.5736	93.0283	93.7946	918
SPEC-TYPE	96.0933	96.4297	96.2612	93.0471	93.3727	93.2096	90.0442	87.8575	88.9374	1853
STATUS	96.1485	96.5075	96.3276	93.0719	93.4194	93.2453	90.8676	85.5914	88.1506	1395
STMT-TYPE	96.1881	96.5697	96.3785	93.0503	93.4194	93.2345	93.0591	84.9765	88.8344	426
STRESSED	96.1723	96.5464	96.3590	93.1195	93.4816	93.3002	100.0000	100.0000	100.0000	0
TENSE	96.2021	96.5464	96.3739	93.0708	93.4039	93.2371	89.4737	80.1887	84.5771	212
TIME	96.1723	96.5464	96.3590	93.1117	93.4739	93.2924	100.0000	100.0000	100.0000	6
TYPE	96.1953	96.5619	96.3782	93.1499	93.5050	93.3271	98.0769	94.4444	96.2264	54
VFORM	96.1723	96.5464	96.3590	93.1195	93.4816	93.3002	100.0000	100.0000	100.0000	0
VTYPE	96.1649	96.5464	96.3552	93.0658	93.4350	93.2500	91.1807	85.5808	88.2920	749

B.2.3 Multiple Atomic Annotations

	unlabelled			labelled			features			occ
	precision	recall	fscore	precision	recall	fscore	precision	recall	fscore	#
BASELINE	96.1723	96.5464	96.3590	93.1195	93.4816	93.3002	0.0000	0.0000	0.0000	0
NUM_PERS	96.2805	96.6475	96.4636	93.2817	93.6372	93.4591	90.8413	86.1964	88.4579	4948
NUM_PERS_GEND	96.1652	96.5541	96.3593	93.3607	93.7383	93.5491	90.5897	85.0490	87.7320	5003
PERF_TENSE	96.0731	96.4841	96.2782	92.9905	93.3883	93.1890	92.0312	84.7482	88.2397	695
PERF_TENSE_PASSIVE_MOOD	96.2277	96.6319	96.4294	93.1139	93.5050	93.3090	92.7476	85.4756	88.9632	778

B.3 Lexical Feature Annotations

B.3.1 Lexical Preterminal Annotations

	unlabelled			labelled			features			occ
	precision	recall	fscore	precision	recall	fscore	precision	recall	fscore	#
BASELINE	96.1723	96.5464	96.3590	93.1195	93.4816	93.3002	0.0000	0.0000	0.0000	0
AUX-SELECT	96.2584	96.6553	96.4564	93.2295	93.6139	93.4213	97.9980	94.1346	96.0275	1040
NEG-FORM	96.1726	96.5541	96.3630	93.1123	93.4816	93.2966	100.0000	100.0000	100.0000	30
PRECONJ-FORM	96.1723	96.5464	96.3590	93.1195	93.4816	93.3002	100.0000	100.0000	100.0000	14
PRON-FORM	96.1267	96.5230	96.3245	93.0126	93.3961	93.2040	100.0000	91.7647	95.7055	170

B.3.2 Lexical Root Annotations

	unlabelled			labelled			features			occ
	precision	recall	fscore	precision	recall	fscore	precision	recall	fscore	#
BASELINE	96.1723	96.5464	96.3590	93.1195	93.4816	93.3002	0.0000	0.0000	0.0000	0
COMP-FORM	95.3244	95.7752	95.5493	91.3278	91.7597	91.5433	82.7160	74.0331	78.1341	181
COMP-FORM-ANAPH	96.1723	96.5464	96.3590	93.1195	93.4816	93.3002	100.0000	100.0000	100.0000	0
CONJ-FORM	96.2782	96.5853	96.4315	93.2620	93.5594	93.4105	90.3226	87.5000	88.8889	160
CONJ-FORM-COMP	96.1723	96.5464	96.3590	93.1195	93.4816	93.3002	100.0000	100.0000	100.0000	0
FORM	96.1723	96.5464	96.3590	93.1195	93.4816	93.3002	0.0000	0.0000	0.0000	3
PCASE	96.4012	96.8886	96.6443	93.4370	93.9095	93.6727	88.3173	84.2536	86.2376	978
PREDET-FORM	96.1878	96.5619	96.3745	93.1195	93.4816	93.3002	100.0000	100.0000	100.0000	0
SPEC-FORM	96.1410	96.5075	96.3239	93.0957	93.4505	93.2728	88.6541	85.5150	87.0563	932

B.3.3 Multiple Atomic Lexical Annotations

	unlabelled			labelled			features			occ
	precision	recall	fscore	precision	recall	fscore	precision	recall	fscore	#
BASELINE	96.1723	96.5464	96.3590	93.1195	93.4816	93.3002	0.0000	0.0000	0.0000	0
PREDET-FORM_PREDET-TYPE	96.1878	96.5619	96.3745	93.1195	93.4816	93.3002	100.0000	100.0000	100.0000	0
SPEC-TYPE_SPEC-FORM	96.1243	96.4608	96.2923	93.0781	93.4039	93.2407	90.0332	87.7023	88.8525	1854