GF-DOP: Grammatical Feature Data-Oriented Parsing

Ríona Finn, Mary Hearne, Andy Way and Josef van Genabith

National Centre for Language Technology,
School of Computing,
Dublin City University

Proceedings of the LFG06 Conference

Universität Konstanz

Miriam Butt and Tracy Holloway King (Editors)

2006

CSLI Publications

http://csli-publications.stanford.edu/

Abstract

This paper proposes an extension of Tree-DOP which approximates the LFG-DOP model. GF-DOP combines the robustness of the DOP model with some of the linguistic competence of LFG. LFG c-structure trees are augmented with LFG functional information, with the aim of (i) generating more informative parses than Tree-DOP; (ii) improving overall parse ranking by modelling grammatical features; and (iii) avoiding the inconsistent probability models of LFG-DOP. In a number of experiments on the HomeCentre corpus, we report on which (groups of) features most heavily influence parse quality, both positively and negatively.

1 Introduction

This paper proposes an extension of Tree-DOP [e.g. Bod, 1992, 1998] which approximates the LFG-DOP model [Bod and Kaplan, 1998, 2003]. GF-DOP combines the robustness of the DOP model with some of the linguistic competence of LFG. LFG c-structure trees are augmented with LFG functional information, with the aim of (i) generating more informative parses than Tree-DOP; (ii) improving overall parse ranking by modelling grammatical features; and (iii) avoiding the inconsistent probability models of LFG-DOP. In a number of experiments on the HomeCentre corpus, we report on which (groups of) features most heavily influence parse quality, both positively and negatively.

The remainder of the paper is organised as follows. In section 2, we describe the Tree-DOP model. An overview of LFG-DOP is provided in section 3, and the new GF-DOP model is described in section 4. We motivate the experiments carried out on the HomeCentre corpus in section 5, and provide results and evaluation in section 6. Finally, we conclude and list a number of avenues for further work.

2 Tree-DOP

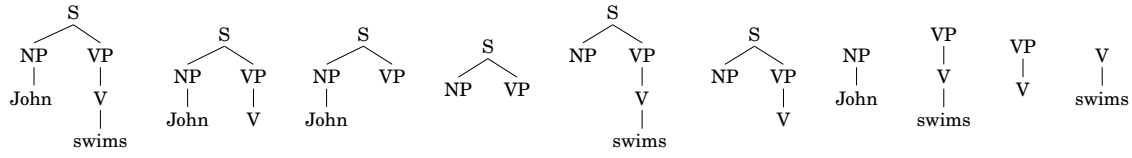
Data-oriented models of language [e.g. Bod, 1992, 1998] are based on the assumption that humans perceive and produce language by availing of previous language experiences rather than abstract grammar rules. These models exploit large treebanks comprising linguistic representations of previously occurring utterances. Analyses of new input sentences are produced by combining fragments from the treebank; the most probable analysis is determined using the relative frequencies of these fragments.

The tree fragments used in Tree-DOP are called subtrees. Two decomposition operators are used in order to produce subtrees from sentence representations:

1. the *root operator* which takes any node in a tree to be the root of a subtree and deletes all nodes except this new root and all nodes dominated by it;

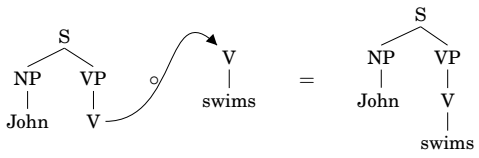
2. the *frontier operator* which selects a (possibly empty) set of nodes in the newly created subtree, excluding the root, and deletes all subtrees dominated by these nodes.

As an example, the complete set of DOP fragments which can be derived from the representation of *John swims* is shown in (1).



(1)

Representations for new input are formed by combining other fragments using the composition operator, namely leftmost substitution, which ensures that each derivation in DOP is unique. The composition of trees t_1 and t_2 ($t_1 \circ t_2$) is only possible if the leftmost frontier node of t_1 and the root node of t_2 are of the same category. The resulting tree is a copy of t_1 where t_2 has been substituted at its leftmost nonterminal frontier node, as demonstrated in (2).



(2)

The probability of a derivation is the joint probability of choosing each of the subtrees involved in that derivation. Letting $|e|$ be the number of times subtree e occurs in the corpus and r(e) be the root node category of e, the probability assigned to e is as in (3).[1]

$$P(e) = \frac{|e|}{\sum_{u:r(u)=r(e)} |u|} \tag{3}$$

The probability of a derivation is the product of the probabilities of choosing each of the subtrees involved in that derivation. Thus, the probability of a derivation $t_1 \circ ... \circ t_n$ is given by (4).

$$P(t_1 \circ ... \circ t_n) = \prod_i P(t_i) \tag{4}$$

A parse tree can potentially be generated by many different derivations, each of which has its own probability of being generated. Therefore, the probability of a parse tree T is the sum of the probabilities of its distinct derivations as in (5).

$$P(T) = \sum_{D \ derives \ T} P(D) \tag{5}$$

3 LFG-DOP

The LFG-DOP model differs from the Tree-DOP model in that the assumed corpus is annotated with LFG representations, i.e. <c,ϕ,f> triples comprising c-structure, ϕ-links and f-structure. The definitions of the fragmentation operators and composition operator, along with the probability model, must be adapted accordingly.

The fragmentation operators for LFG-DOP are extensions of those used in Tree-DOP as we wish to extract exactly the same set of generalised c-structure fragments as before. However, we also wish to extract the corresponding f-structure fragment to go with each c-structure. Consequently, the original *root* and *frontier* operators must be extended to take f-structure into account. Many different extensions can be envisaged; those defined in [Bod and Kaplan, 1998, 2003, Bod, 2000b,a] are as follows:

Root Given a copy of the example-base representation <c,ϕ,f> named <c_{copy},ϕ_{copy},f_{copy}>:

1. select a node in c_{copy} to be *root* and delete all nodes except this node and the nodes it dominates;
2. delete all links in ϕ_{copy} which link deleted c-structure nodes to f_{copy};

[1] We use this estimation method for all experiments presented in this paper, although others (e.g. [Zollmann and Sima'an, 2005]) are possible.

3. delete all f-structure units in f_{copy} which are not ϕ-*accessible* from c_{copy};

4. delete all semantic forms in f_{copy} which are local to f-structure units corresponding to erased c-structure terminals.

Frontier Given a representation of the form $<c_{copy},\phi_{copy},f_{copy}>$ created by the root operation:

1. select a (possibly empty) set of nodes in c_{copy} to be *frontier* nodes and delete all nodes dominated by these newly-created frontier nodes;

2. delete all links in ϕ_{copy} which link deleted c-structure nodes to f_{copy};

3. *(delete all f-structure units in f_{copy} which are not* ϕ-accessible *from c_{copy};)*

4. delete all semantic forms in f_{copy} which are local to f-structure units corresponding to erased c-structure terminals.

Step 3 of the frontier operation is given here for the sake of completeness; as a consequence of the definition of ϕ-accessibility given in [Bod and Kaplan, 1998, 2003, Bod, 2000b,a] and described below, the root node of c_{copy} (selected during the root operation) accesses all f-structure units in f_{copy} regardless of which nodes are selected by the frontier operation. This definition of ϕ-accessibility is as follows:

ϕ-accessibility An f-structure unit f is ϕ-accessible from c-structure node n if and only if

1. n is ϕ-linked to f, i.e. $\phi(n) = f$, or

2. n is ϕ-linked to f_x and f contains f_x i.e. there is a chain of attributes leading from f to f_x.

The extended root and frontier operations for LFG-DOP yield precisely the same c-structures as are yielded by the root and frontier operations defined for Tree-DOP. However, it is also possible to extract further fragments from each fragment yielded by the root and frontier operations via the *discard* operation. Discard is used to delete attribute-value pairs from the f-structure whose values are not ϕ-linked to remaining nodes in the c-structure and are not surface forms corresponding to c-structure terminals. Thus, when discard is applied to any fragment $<c,\phi,f>$, a new fragment $<c,\phi,f_{d_x}>$ is extracted; the c-structure and ϕ-links are unchanged but f_{d_x} differs from f in that all attribute-value pairs in f_{d_x} are also in f but the reverse does not hold, i.e. it is not the case that all attribute-value pairs in f are also in f_{d_x}.

The LFG-DOP composition operation involves two stages: leftmost substitution over c-structure and recursive unification over f-structure such that the ϕ-links are not broken. That is, a pair of c-structures are first composed exactly as for Tree-DOP and, subsequently, the f-structure parts of those fragments are unified. Any sequence of composition operations yielding a complete derivation (i.e. one which contains no open c-structure substitution sites) is only valid if that derivation's f-structure adheres to the LFG well-formedness conditions. However, the presence in the fragment base of discard-generated fragments means that many input strings which are ill-formed with respect to the corpus can also be parsed.

We can estimate LFG-DOP fragment probabilities as for Tree-DOP, i.e. compute their empirical frequencies conditioned on the root node. However, this estimator draws no distinction between fragments generated by the root and frontier operators and discard-generated fragments. Fragments generated by discard effectively relax the constraints specified in the f-structure to allow the fragment to be used in a wider variety of contexts and so are very useful in constraint-based DOP

parsing. Intuitively, however, they should only be considered when no parse can be produced which satisfies all relevant constraints, i.e. they should be used only when the input is ill-formed with respect to the corpus. Consequently, this estimator does not seem entirely appropriate. Way [1999] and Bod and Kaplan [2003] propose an alternative method – termed 'discounted RF', as opposed to 'simple RF' – of estimating fragment probabilities whereby root and frontier fragments are treated as seen events and discard fragments as unseen events. The fragment set is partitioned using this distinction and two separate probability distributions induced. The probabilities of seen events are estimated by their relative frequencies as before. However, these probabilities are then discounted and the discounted mass distributed amongst the unseen events, i.e. the discard-generated fragments.

In the Tree-DOP model, valid derivations are constructed by composing fragments such that the category-matching condition is fulfilled. Each valid derivation is assigned a probability by calculating the product of the probabilities of the fragments used in the construction of that derivation. The probabilities of all valid derivations which can be constructed from a given DOP grammar for all of the strings which it recognises sum to 1.

Each LFG-DOP derivation probability is also calculated as the product of the probabilities of the fragments used in the construction of that derivation. In the LFG-DOP model, however, we have seen that valid derivations are constructed by composing fragments such that the category-matching, uniqueness, completeness and coherence conditions are fulfilled. If we calculate LFG-DOP fragment probability distributions in the same way as we did for Tree-DOP – i.e. define distributions over root node category – then the probabilities of all derivations for all the strings recognised by the grammar which adhere to the category-matching condition will sum to 1. However, it is not the case that all of these derivations are valid according to the LFG-DOP model as they may not fulfil the uniqueness, completeness and coherence conditions. Consequently, the probabilities of all *valid* derivations which can be constructed from a given LFG-DOP grammar for all of the strings which it recognises no longer sums to 1 and, therefore, do not constitute a probability distribution. [Bod and Kaplan, 1998, 2003] handle this by normalisation: parse probability is divided by the sum of the probabilities of all *valid* parses for the input string. However, Abney [1997] observes that normalisation serves only to mask the fact that, unlike for the context-free case, establishing probabilities for grammars encoding context-sensitive dependencies using relative frequency estimation does not yield the best weights.

4 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 treebank transformed by the addition of further linguistic information: features are extracted from f-structures and appended to the c-structure category labels to form a new, extended set of 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 category labels, so labels which incorporate features incur no extra processing and no changes to the model are required to handle the new set of extended category 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 informed output based on identification and incorporation of grammatical functions and features.

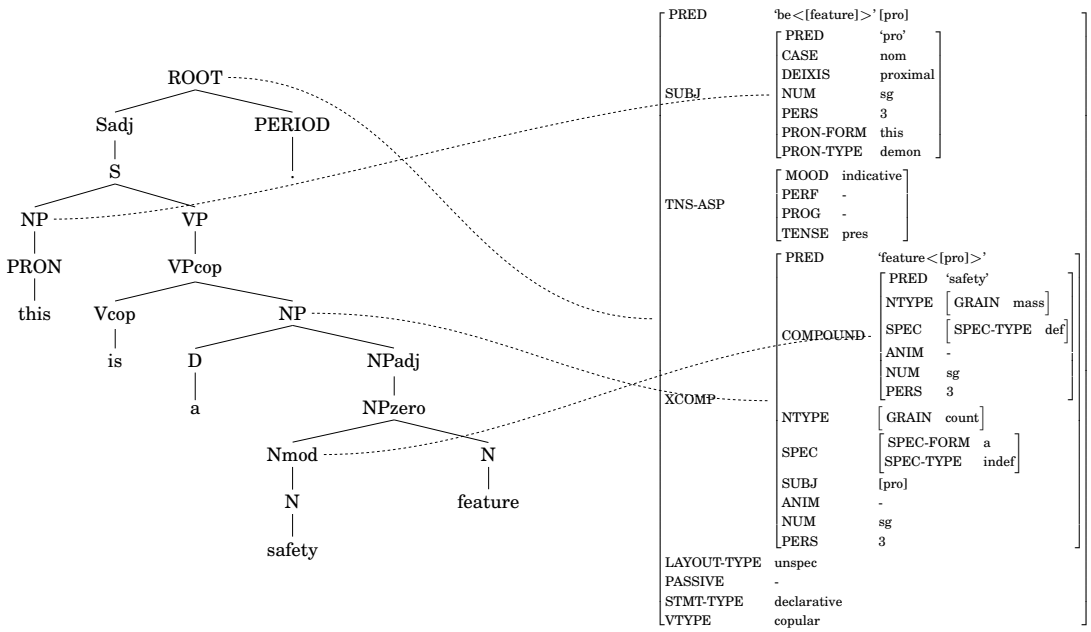


Figure 1: A c-structure with its corresponding ϕ-linked f-structure, from which we extract features.

4.1 Feature Classification

F-structures contain informative features (for example LAYOUT-TYPE may specify that this sentence is a header, a list item or is unspecified) and functional information (such as SUBJ and OBJ) which describe the grammatical functions of the constituents in question. A linked c-structure and f-structure representation can be seen in Figure 1; this representation contains examples of each of the 5 feature classes identified:

- grammatical functions, e.g. SUBJ, XCOMP,
- atomic features, e.g. NUM=sg, PERS=3,
- lexical features, e.g. PRON-FORM=this, SPEC-FORM=a,
- 'super features' (or 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<[feature]>'[pro], PRED 'feature<[pro]>'.

4.2 Annotating with Grammatical Features

4.2.1 Grammatical functions

Using ϕ-linked f-structure units, we identify functions of constituents within the c-structure. For example, the leftmost NP in the c-structure representation in Figure 1 functions as the SUBJ of *'be'*. We transform the tree by appending this information to the syntactic category label, giving 'NP_SUBJ'. We place the annotation on the topmost node in the constituent which corresponds to the

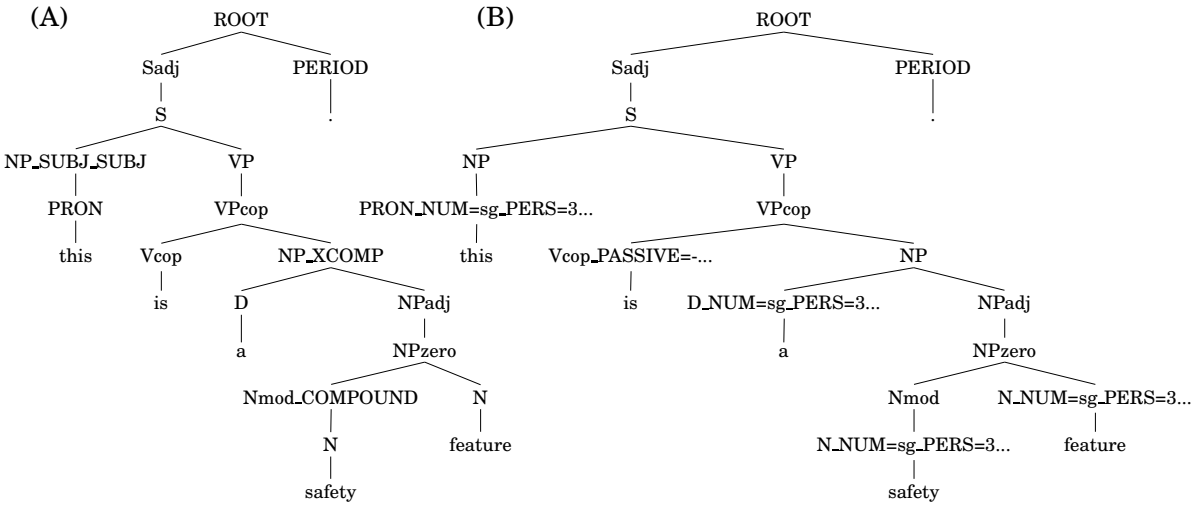


Figure 2: (A) Illustration of a c-structure with functional annotations on the top-most nodes of appropriate constituents. (B) Illustration of a c-structure with atomic annotations on the preterminal nodes.

function in question. All nodes dominated by this annotated node are part of the 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 top-most node in the constituent which serves 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_SUBJ'. A c-structure annotated with functions can be seen in Figure 2 (A).

4.2.2 Atomic features

The second class of features is atomic features. These features have a small set of closed class items as possible values; for example the feature PERS can only ever have the value 1, 2 or 3. We annotate pre-terminal nodes with atomic features, as these nodes are closest to the terminals to which the annotations specifically apply. A single atomic feature may apply to more than one node; in this case, each such node receives the atomic annotation.

Looking at the ϕ-linked f-structure for the sentence in Figure 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 features LAYOUT-TYPE, PASSIVE, STMT-TYPE and VTYPE should be annotated on all pre-terminal 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 a practical implementation of 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 1, only the Vcop node receives these annotations, as illustrated in Figure 2 (B).

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

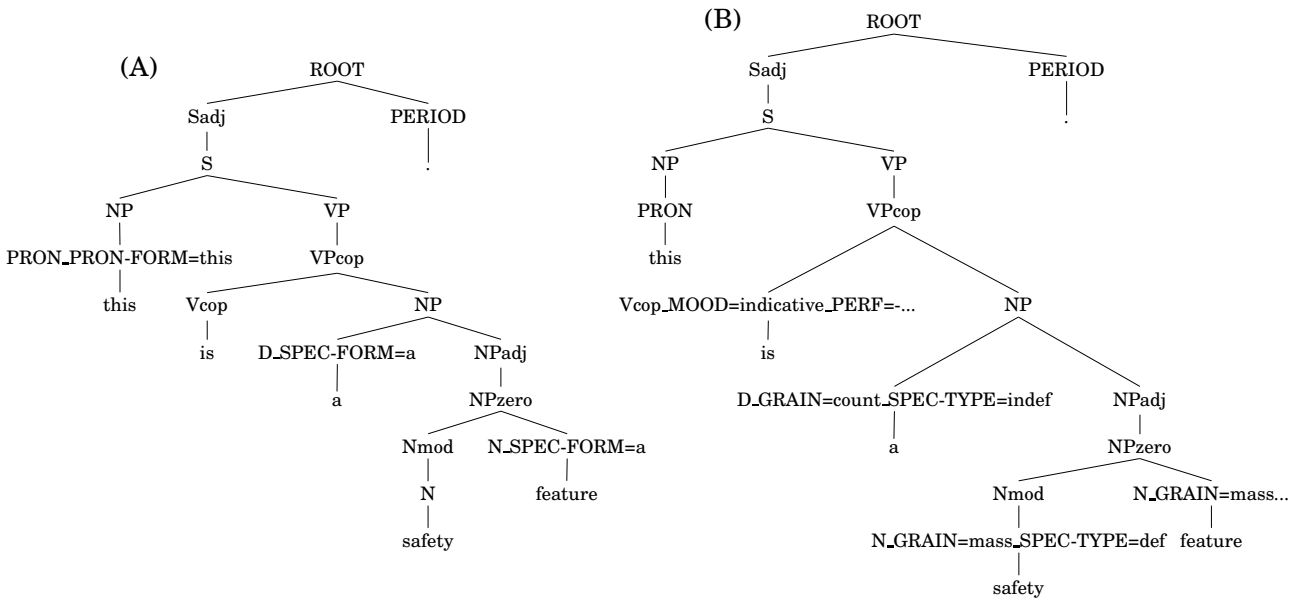


Figure 3: (A) Illustration of a c-structure with lexical annotations on pre-terminal nodes. (B) Illustration of a c-structure with super-feature value atomic annotations on pre-terminal nodes.

4.2.3 Lexical features

The third class of features 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. Lexical annotations are placed on the pre-terminal nodes dominating the terminals they relate to. An example of a lexically annotated c-structure can be seen in Figure 3 (A); the PRON-FORM is specified as *this*. The SPEC-FORM used with *feature* is specified as *a*, also indicated on this c-structure. Where a lexical feature applies to two or more nodes, each node is annotated with this feature.

4.2.4 Super-features

We call the fourth class 'super-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 super-feature's f-structure value: rather than identifying that a node has, for example, tense and aspect, denoted by the feature 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 the other atomic features, as described in section 4.2.2. An example of these annotations can be seen in Figure 3 (B); 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, it would imply that the other constituents dominated by VP or VPcop also have tense and aspect.

4.2.5 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 4.2.3 because lexical features can have only

a small number of lemmas, essentially a closed class set, as their values, while PRED can have any lemma as its value. The PRED feature may also have subcategorisation arguments; this is a list of arguments which are required by the predicate.

Let us consider the annotation possibilities for this feature. From the PRED we can establish the lexical word, the list of obligatory arguments, and adjuncts. 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. However, the list of arguments 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, the GF-DOP model presented here does not make use of subcategorisation information.

4.3 Preserving Robustness

Data-sparseness is a prominent issue in parsing and is exacerbated by the highly specific node labels in GF-DOP. The GF-DOP model's use of additional feature information could lead to reduced coverage, i.e. there may be sentences which can be parsed by the Tree-DOP model but not the GF-DOP model. This would constitute a weakness in the GF-DOP model. To address this issue and preserve robustness, we further extend the GF-DOP model by introducing a 'backing-off'.

We achieve this via a two-step training procedure: we extract grammars from both the annotated treebank and a version of the treebank with the annotations stripped away. We merge these grammars by assigning the majority of the probability mass (W_1) to the annotated grammar and the remainder ($W_2 = 1 - W_1$) to the unannotated grammar. This ensures that the GF-DOP model maintains the same level of coverage as the Tree-DOP model.

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

When compared to the Tree-DOP model, GF-DOP has the following advantages:

1. it has the capacity to generate more informative parses;
2. its probability model is sensitive to grammatical feature information which can help to improve the overall parse ranking;
3. it displays the above advantages without losing any of the coverage of the Tree-DOP model.

The Tree-DOP model is limited by the representations it assumes. The GF-DOP model trains on data with a greater degree of linguistic information than the Tree-DOP model and, consequently, the parses it generates contain more information than those of Tree-DOP. This additional information also plays a part in determining the basic phrase-structure tree assigned to the input string: better parses which are supported by the additional grammatical feature information will have their probabilities boosted and, correspondingly, those which are not supported by these features will be ranked lower. Finally, we note that these advantages do not, as might have been expected, come at a cost in terms of robustness. Back-off is integral to the model and it ensures that all sentences which can be parsed by Tree-DOP can also be parsed by GF-DOP.

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

The GF-DOP model also improves on the LFG-DOP model:

1. Due to difficulties in establishing a valid probability model, there is currently no satisfactory realisation of an LFG-DOP system. In contrast, the GF-DOP model maintains the integrity of the original Tree-DOP probability model.

2. The implementation of discard in LFG-DOP is computationally expensive: exponentially many more fragments are generated than for Tree-DOP. Although the integration of back-off into the GF-DOP model increases the number of fragments, the resulting grammar contains at most double the number of fragments of the Tree-DOP model.

LFG-DOP's strength comes from the information contained in the assumed representations and the corresponding extensions to fragmentation and composition. 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 required to produce a valid parse. Because these ill-formed derivations are excluded as they are encountered, probability mass is lost; the probability distribution of derivations does not correspond to the probability model.

The GF-DOP model, in contrast, enforces only category-matching 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.

A correspondence can be drawn between the 'backing-off' technique employed in the GF-DOP model and the 'discounted RF' technique employed in the LFG-DOP model in that both assign a limited proportion of the available probability mass to fragments which are unseen in the treebank. There is, however a crucial difference: exponentially many fragments are generated using discard because all possible combinations of attribute/value pair deletion are applied whereas this is not the case for GF-DOP back-off because all feature annotations are deleted simultaneously. This renders the GF-DOP model less powerful but more computationally tractable.

4.6 The GF-DOP hypothesis

Having outlined the theoretical characteristics and advantages of the GF-DOP model, we hypothesise that this new model will:

1. give us better phrase-structure tree parse accuracy than the Tree-DOP model and

2. allow us to learn grammatical features with a high degree of accuracy.

5 Experiments

We have carried out a set of experiments in order to determine whether the theoretical advantages of GF-DOP outlined in Section 4 are reflected in the performance of the model. In this section, we describe the data and parser used to carry out these experiments.

5.1 The Data

The treebank used in the experiments presented here is the English section of the Xerox HomeCentre corpus. This treebank consists of 980 sentences annotated with c-structures and corresponding ϕ-linked f-structures. In this data-set, 75 features (excluding PRED) were identified. These features were divided into 4 classes. This classification can be seen in Table 1. As stated in section

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

Table 1: Classification of 75 features identified in the English section of the data set.

4.2.4, we do not annotate with super-features explicitly; rather we use the features listed within their f-structure values. In addition to these, there are 5 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, thus 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 corpus annotated with this feature. Any such corpus would essentially be the same as the baseline (original, unannotated) corpus.

- 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 corpus annotated with this feature.

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

Thus, the number of features we use in generating treebanks annotated with only a single feature is 61, and these features are listed and classified in Table 2. In addition, we generate 3 multi-feature

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

Table 2: List and classification of the 61 features for which we generated singly-annotated treebanks.

treebanks: one with all the functional annotations listed in Table 2, one with the five functions most prevalent in the data (ADJUNCT, OBJ, SUBJ, COMPOUND and XCOMP) and one annotated with the SUBJ and OBJ functions.

5.2 Experimental set-up

From the training treebank we generated eight training sets of 890 sentences each and eight corresponding test sets of 90 sentences each. The splits were generated at random[2] such that every word in the test set is present in the corresponding training set, thus avoiding the issue of unknown words at this time.

For each feature presented in Table 2, an annotated corpus was created and the eight pre-established splits applied. For each split, the parser is trained on the training set, tested on the test set and evaluated on the corresponding reference set. Scores are then averaged over the eight splits for each annotation type.

5.3 Parser details

Training our DOP parser involves extracting the DOP fragment set and associating probabilities with each fragment. Testing then involves submitting one or more sentences to the parser, applying the fragment set to establish a parse forest and selecting the best parse from that forest to output. There are a number of methodologies available to us (e.g. [Bod, 1995], [Sima'an, 1995, 1999], Goodman [1996, 1998, 2003]) in implementing our DOP system. Details of the parser used to perform the GF-DOP experiments presented in this paper are given below.

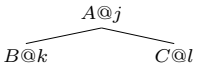
5.3.1 Training

Goodman [1996, 1998, 2003] describes a method by which the DOP grammar projected from a treebank in which all trees are binary branching is reduced to a PCFG containing at most eight rules for each node in the training data. This PCFG is equivalent to the DOP grammar in that a) it generates the same strings with the same probabilities and b) it generates the same parse trees with the same probabilities, although one must sum over several PCFG trees for each DOP tree.

Goodman PCFG-reductions are constructed as follows. Every node in every tree in the treebank is assigned a unique address: $A@k$ is the node labelled A at address k. One new non-terminal A_k

[2]The eight test sets of 90 sentences each are not disjoint; because they were extracted at random, it is entirely possible that they overlap to some extent.

is created for every node in the treebank; such non-terminals are called "interior" nodes and the original nodes "exterior" nodes. a_k is the number of subtrees with root node $A@k$ and a the number of subtrees with root node label A, i.e. $a = \sum_j a_j$. Given node $A@k$ with a set CH of two or more children $CH = \{B@l...C@m\}$, the number of fragments a_k which have root node $A@k$ is calculated by multiplying the numbers of fragments yielded by each of its children: $a_k = \prod_{X@n \in CH}(x_n + 1)$.

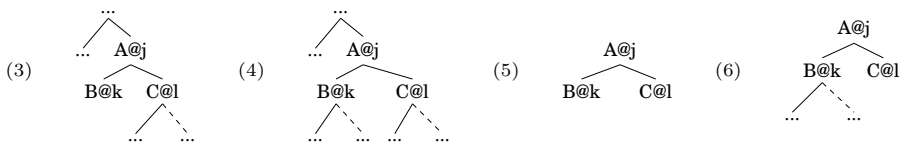


(6)

For any node grouping such as the one in (6), the eight PCFG rules and their corresponding probabilities in (7) are then extracted; Goodman provides proofs by induction that the rule probabilities are valid.

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

These rules correspond to the eight possible contexts in which the node grouping in (6) can occur in fragments extracted from the corresponding treebank tree; each of the three nodes can be either interior or exterior (i.e. root node or substitution site) to any fragment in which the grouping occurs. The examples in (8) illustrate the contexts to which rules (3) – (6) in (7) correspond. Node *A@j* is an interior (i.e. non-root) node in rules 3 and 4 and an exterior (i.e. root) node in rules 5 and 6 – the parent node of any grouping (the node which appears on the left-hand side of the rule) corresponds to either a root or internal node but not a substitution site. Conversely, the child nodes of each grouping, which appear on the right-hand side of the corresponding rules, can be either internal nodes or substitution sites but never root nodes as shown in (8). As previously stated, Goodman's PCFG reduction requires the projection of *at most* eight rules for each node in the treebank. The maximum number of rules are projected from each node which is internal to a treebank tree and dominates two non-terminal children; four rules are projected from each node corresponding to the root node of a treebank tree, as this node can never be internal to a fragment, and two rules are projected from nodes dominating a single terminal symbol as terminal symbols are never substitution sites.



(8)

A PCFG-reduction derivation is isomorphic to a DOP derivation if for every substitution of a DOP fragment there is a corresponding sub-derivation in the PCFG. In other words, each PCFG sub-derivation yielding a subtree whose internal nodes are all of the form X_y, whose root node is of the form X and whose frontier nodes are either of the form X or are terminal symbols, corresponds exactly to a DOP fragment when the subscripts are removed. Furthermore, each such PCFG sub-derivation has exactly the same probability as the DOP fragment to which it corresponds.

Thus, for each of the eight training splits for each annotation type, we induced a GF-DOP grammar as follows:

1. binarise the training trees;

2. extract a PCFG-reduction from the annotated trees;

3. strip away the feature annotations from the treebank and extract a PCFG-reduction from the unannotated trees;

4. merge the extracted grammars with weights $W_1 = 0.99$ and $W_2 = 0.01$.

5.3.2 Parsing

During parsing, the GF-DOP grammar described above is applied to the input string and a derivation forest is built using the CKY and Viterbi algorithms in combination. Viterbi allows us to prune the derivation space as it is built such that the final derivation space contains the single best derivation for the input string. This is achieved by pruning sub-derivations with low probabilities from the PCFG-reduction derivation space in a bottom-up manner. Two different sub-derivations which have the same root node and span the same portion of the input string are used in building derivations of the entire input string in exactly the same way. This means that parses containing the more probable of these sub-derivations will always be more probable than those derivations containing the less probable sub-derivation. Consequently, the less probable sub-derivation will never be used to build the most probable derivation/parse and can be removed from the derivation space.

To this Viterbi derivation space we then apply the method of Jiménez and Marzal [2000] in order to determine the n-best derivations for the input string, where we set n to 2,000. We sum over the probabilities of the parses yielded by the n-best derivations, and return the one with the highest probability.

6 Evaluation

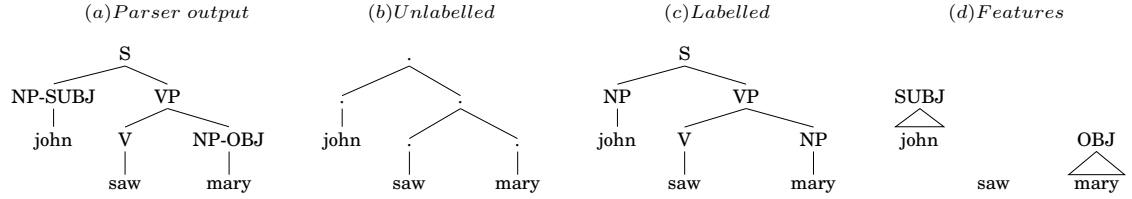
We evaluate our parser output using the standard precision, recall and f-score metrics as given in (9), (10) and (11).

$$Precision = \frac{\#\ correct\ constituents}{\#\ parse\ constituents} \tag{9}$$

$$Recall = \frac{\#\ correct\ constituents}{\#\ reference\ constituents} \tag{10}$$

$$F - Score = \frac{2 * Precision * Recall}{Precision + Recall} \tag{11}$$

We transform our parser output and reference trees in three different ways and then apply the above metrics to each. The purpose of these transformations is to facilitate evaluation of unlabelled parse accuracy, labelled parse accuracy and feature detection accuracy. The transformations are illustrated in (12), where (12)(a) gives the parser output and (12)(b) – (d) the three transformations applied to the parser output. In (12)(b) all labels have been replaced with the generic '.' label. The application of the evaluation metrics to these transformations gives unlabelled parse accuracy. In (12)(c) all feature annotations have been stripped away, allowing evaluation of labelled parse accuracy. Finally, in (12)(d) all labels with feature annotations are stripped of syntactic category so that only those feature annotations remain, and all other constituents are deleted. This last transformation allows us to evaluate feature annotation accuracy.



(12)

The results of our experiments with the GF-DOP model are given in Tables 3, 4 and 5. The second and third colums in each of these tables (marked unlabelled and labelled) give the results for parse accuracy while the fourth column (features) gives the results for grammatical feature detection. The rightmost column in each (marked occ) gives the number of occurrences of feature annotations in the reference representations. Essentially, this column tells us how many feature annotations we were looking to identify[3] across the 8 test sets (720 test sentences in total) for each annotation run. In each of the tables, the first line of results corresponds to the baseline, i.e. the results for the run with no grammatical feature annotations on the treebank. The baseline scores in each of the tables are identical, and are repeated for convenience only. Coverage for all runs including the baseline was 93.89%.[4]

6.1 Functional annotation

The results of our experiments with functional annotation are given in Table 3. Focusing firstly on single-function annotations, we observe that the unlabelled f-score is higher for every annotated treebank than it is for the baseline. OBJ and ADJUNCT give the highest improvements over the baseline (of 0.2279% and 0.1749% respectively). We observe similar trends for labelled f-score in that OBJ and ADJUNCT again give the highest improvements over the baseline (of 0.4219% and 0.3347% respectively). A single annotation type, SUBJ, yields a decrease in labelled f-score over the baseline (of -0.0321%); all others yield an increase. The f-scores for grammatical feature detection range from 64.2857% (OBL) up to 100% (OBL-COMP, TOPIC-INT). If we focus on those features with reference set occurrences of at least 100 (ADJUNCT, OBJ, SUBJ, COMPOUND, XCOMP) this range of accuracy narrows to 68.7783% – 87.4267%, with highest accuracy for ADJUNCT and lowest for XCOMP. In fact, we score significantly worse for XCOMP than for the next lowest performing feature, which is SUBJ at 80.9938%.

Multi-function annotation results are given in the last three lines of Table 3. Annotation type ALL refers to the treebank annotated with all 16 of the functions listed as single annotations. Annotation type TOP5 refers to the treebank annotated with the 5 most frequently occurring functions which, as above, are ADJUNCT, OBJ, SUBJ, COMPOUND and XCOMP. Finally, annotation type SUBJ_OBJ refers to the treebank annotated with those two functions only. We observe firstly that the unlabelled f-scores for these annotated treebanks are not only higher than the baseline (by 0.2573% for ALL, by 0.2967% for TOP5 and by 0.2135% for SUBJ_OBJ) but also higher than all single annotations with the exception of OBJ, which outperforms SUBJ_OBJ by 0.0144%. The same observations hold for labelled f-score, where ALL improves over the baseline by 0.6454%, TOP5 by 0.6080% and SUBJ_OBJ by 0.4077%. Furthermore, we note that this improvement in tree structure accuracy does not cause feature detection accuracy to suffer: the f-scores for grammatical feature detection hold up well, ranging between 84.4528% and 85.1899%.

6.2 Annotation with lexical features

The results of our experiments with lexical feature annotation are given in Table 4. We observe that all but one of the annotations (SPEC-FORM, decrease of -0.0122%) give an improvement over the baseline in terms of unlabelled f-score. The greatest improvement is gained by annotating with

[3] An occurrence count of 0 for an annotation type means that the feature annotation occurred in the training data but never in the test/reference data. Grammatical feature detection scores are nevertheless given as they reflect the presence or absence of false positives.

[4] Those sentences which could not be fully parsed were assigned the most probable sequence of partial parses linked together by a 'TOP' node.

	unlabelled			labelled			features			occ
	precision	recall	fscore	precision	recall	fscore	precision	recall	fscore	#
BASELINE	96.0619	96.3603	96.2109	92.6076	92.8953	92.7512	—	—	—	—
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	100.000	100.000	100.000	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	100.000	100.000	100.000	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	100.000	100.000	100.000	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	100.000	100.000	100.000	0
OBL-COMP	96.0913	96.3991	96.2450	92.7148	93.0117	92.8630	100.000	100.000	100.000	3
PRON-INT	96.0913	96.3991	96.2450	92.7148	93.0117	92.8630	100.000	91.6667	95.6522	24
PRON-REL	96.0913	96.3991	96.2450	92.7148	93.0117	92.8630	100.000	75.0000	85.7143	4
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	100.000	100.000	100.000	25
TOPIC-REL	96.0913	96.3991	96.2450	92.7148	93.0117	92.8630	100.000	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
ALL	96.3049	96.6320	96.4682	93.2385	93.5553	93.3966	87.5106	81.6351	84.4708	2532
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

Table 3: Evaluation of the DOP model with training data annotated with grammatical functions.

PCASE, where the increase is 0.2196%. When we look at the labelled f-scores we see that annotating with SPEC-FORM again leads to a tiny decrease in accuracy, this time of -0.0099%. In contrast, the greatest improvement is gained by annotating with COMP-FORM, where the increase is 0.2817%. The f-scores for grammatical feature detection range from 51.2821% to 100%. If we focus on those features with reference set occurrences of at least 100 (SPEC-FORM, PCASE, COMP-FORM, CONJ-FORM, PRON-FORM) this range of accuracy narrows to 70.4782% – 92.1833%, with highest accuracy for PRON-FORM and lowest for PCASE.

6.3 Annotation with atomic features

The results of our experiments with atomic feature annotation are given in Table 5. Of the 36 different atomic grammatical features we generated treebanks for, 27 give an increase in unlabelled f-score over the baseline and 9 a decrease. Those which give the greatest increases are ADJUNCT-TYPE (0.2082%), PERF (0.1690%), VTYPE (0.1593%), ANIM (0.1497%) and PASSIVE (0.1400%). Those which give the greatest decreases are STMT-TYPE (-0.0667%), PSEM (-0.0520%) and NUM (-0.0517%). When we focus on labelled f-score, we note that 33 of the features give an increase in accuracy and only 3 give a decrease. Those which give decreases are GRAIN (-0.2318%), NUM (-0.1469%) and PROPER (-0.0782%). The greatest increases in labelled f-score are gained by annotating with PASSIVE (0.5369%), VTYPE (0.4206%) and PERF (0.4109%). The f-scores for grammatical feature detection range from 0% to 100%. If we narrow our focus to those features with reference set occurrences of at least 100, this range of accuracy narrows to 81.0526% – 95.1879% with highest accuracy for PERS and lowest for ADJUNCT-TYPE.

	unlabelled			labelled			features			occ
	precision	recall	fscore	precision	recall	fscore	precision	recall	fscore	#
BASELINE	96.0619	96.3603	96.2109	92.6076	92.8953	92.7512	—	—	—	—
COMP-FORM	96.1401	96.4573	96.2984	92.8799	93.1865	93.0329	82.4468	63.5246	71.7593	244
CONJ-FORM	96.0910	96.3894	96.2399	92.6560	92.9438	92.7997	94.7368	75.0000	83.7209	240
CONJ-FORM-COMP	96.1010	96.4088	96.2547	92.7148	93.0117	92.8630	0.0000	0.0000	0.0000	6
PCASE	96.2394	96.6223	96.4305	92.7978	93.1670	92.9820	72.9032	68.2093	70.4782	497
PRECONJ-FORM	96.0913	96.3991	96.2450	92.7148	93.0117	92.8630	100.000	100.000	100.000	0
PREDET-FORM	96.1107	96.4185	96.2644	92.6858	92.9826	92.8340	100.000	50.0000	66.6667	12
PRON-FORM	96.1494	96.4573	96.3031	92.7632	93.0603	92.9115	96.6102	88.1443	92.1833	194
PRT-FORM	96.1021	96.4379	96.2697	92.7749	93.0991	92.9367	83.3333	37.0370	51.2821	27
SPEC-FORM	95.9896	96.4088	96.1987	92.5396	92.9438	92.7413	95.1774	86.5178	90.6412	1209

Table 4: Evaluation of the DOP model with training data annotated with lexical features.

6.4 Discussion

Looking first to general parse accuracy, we note that the best overall unlabelled f-scores are achieved using the TOP5 (96.5076%) and ALL (96.4682%) annotation types. Furthermore, best overall labelled f-scores are also achieved using the ALL (93.3966%) and TOP5 (93.3592%) annotation types. We conclude from this that annotating with multiple grammatical functions gives the greatest improvement in phrase-structure tree parse accuracy. In addition, we conclude that the GF-DOP model generally helps phrase-structure tree parse accuracy rather than hindering it. This conclusion is based on the following: of the 64 annotated runs we carried out, 84.37% gave improvements over the baseline in terms of unlabelled f-score and 93.75% gave improvements over the baseline in terms of labelled f-score.

We do reasonably well at detecting grammatical functions, particularly the 5 most frequent ones, where f-scores are in the range 68.7783% – 87.4267%. When we annotated with all 5 most frequently-occurring functions, grammatical function accuracy remained high at 84.4528%. We also do reasonably well at detecting the most frequent lexical features, where f-scores are in the range 70.4782% – 92.1833%. However, it is questionable as to whether it is really useful to be able to detect such features – they are useful for helping us get better phrase-structure tree accuracy, but do not add much additional information to the output parse. Finally, we do well at detecting the more frequent atomic features, achieving f-scores in the range 81.0526% – 95.1879%.

7 Conclusions and Future Work

This paper proposed a new model – GF-DOP – which combines the robustness of the DOP model with some of the linguistic competence of LFG-DOP. This model incorporates more detailed linguistic information than Tree-DOP and, consequently, improves on the Tree-DOP model in that the output parses are more informative and the probability model is sensitive to this additional information. Although GF-DOP incorporates only some of the linguistic competence of the LFG-DOP model, it nevertheless constitues an improvement over LFG-DOP from both theoretical and practical perspectives: it maintains the integrity of the probability model because there is no 'leaked' probability mass and is more computationally tractable because the increase in grammar size induced by backing-off is not exponential.

We hypothesised that the GF-DOP model would (i) give us better phrase-structure tree parse ac-

	unlabelled			labelled			features			occ
	precision	recall	fscore	precision	recall	fscore	precision	recall	fscore	#
BASELINE	96.0619	96.3603	96.2109	92.6076	92.8953	92.7512	—	—	—	—
ABBREV	96.1300	96.4379	96.2837	92.7148	93.0117	92.8630	100.000	100.000	100.000	30
ACONSTR	96.0817	96.3894	96.2353	92.7148	93.0117	92.8630	100.000	100.000	100.000	0
ADEG-DIM	96.0720	96.3797	96.2256	92.7051	93.0020	92.8533	100.000	64.7059	78.5714	17
ADEGREE	96.1591	96.4670	96.3128	92.8019	93.0991	92.9502	87.9859	88.9286	88.4547	280
ADJUNCT-TYPE	96.2744	96.5641	96.4191	92.8876	93.1670	93.0271	87.5000	75.4902	81.0526	612
ADV-TYPE	96.1107	96.4185	96.2644	92.8212	93.1185	92.9696	85.5691	77.2477	81.1958	545
ANIM	96.2253	96.4962	96.3606	92.7023	92.9632	92.8326	92.7273	83.0018	87.5954	553
ATYPE	96.1304	96.4476	96.2888	92.8122	93.1185	92.9651	87.0370	87.3606	87.1985	269
CASE	95.9965	96.3506	96.1732	92.5829	92.9244	92.7533	89.5514	87.3345	88.4290	1737
DEIXIS	96.0430	96.3506	96.1965	92.6954	92.9923	92.8436	100.000	25.0000	40.0000	16
EMPH	96.0720	96.3797	96.2256	92.7148	93.0117	92.8630	100.000	100.000	100.000	0
EMPHASIS	96.0720	96.3797	96.2256	92.7148	93.0117	92.8630	100.000	100.000	100.000	0
GEND	96.1777	96.4670	96.3221	92.8005	93.0797	92.9399	100.000	73.5294	84.7458	34
GERUND	96.0433	96.3603	96.2016	92.8219	93.1282	92.9748	89.2857	92.5926	90.9091	108
GRAIN	96.0240	96.3409	96.1822	92.3672	92.6720	92.5194	92.3945	91.2306	91.8089	2064
LAYOUT-TYPE	95.9981	96.3894	96.1933	92.6728	93.0506	92.8613	91.3136	81.5516	86.1569	1057
MOOD	96.1505	96.4865	96.3182	92.7459	93.0700	92.9077	90.7631	85.3904	87.9948	794
NUM	95.9687	96.3506	96.1592	92.4207	92.7885	92.6043	94.3262	92.5641	93.4369	2730
NUMBER-TYPE	96.0720	96.3797	96.2256	92.7148	93.0117	92.8630	100.000	95.9596	97.9381	99
PASSIVE	96.2156	96.4865	96.3509	93.1572	93.4194	93.2881	94.8529	91.6519	93.2249	1126
PERF	96.2447	96.5156	96.3799	93.0314	93.2932	93.1621	96.1015	92.6573	94.3480	1144
PERS	96.0460	96.4282	96.2367	92.8171	93.1865	93.0014	95.9372	94.4538	95.1897	2975
POL	96.1297	96.4282	96.2787	92.6851	92.9729	92.8288	0.0000	0.0000	0.0000	6
PREDET-TYPE	96.1107	96.4185	96.2644	92.6858	92.9826	92.8340	100.000	100.000	100.000	12
PROG	96.0928	96.4379	96.2651	92.8820	93.2156	93.0485	96.2557	90.3171	93.1919	1167
PRON-TYPE	96.1788	96.4962	96.3372	92.9186	93.2253	93.0717	94.9664	90.2232	92.5341	941
PROPER	96.0902	96.3700	96.2299	92.5385	92.8079	92.6730	100.000	73.5294	84.7458	34
PSEM	95.9776	96.3409	96.1589	92.6030	92.9535	92.7779	93.2886	88.2540	90.7015	315
PTYPE	96.0913	96.3991	96.2450	92.7245	93.0215	92.8727	92.3810	92.6752	92.5278	314
SPEC-TYPE	96.0746	96.4476	96.2608	92.6327	92.9923	92.8122	94.6950	87.5000	90.9554	1632
STMT-TYPE	95.9675	96.3215	96.1442	92.5926	92.9341	92.7630	92.1453	87.1406	89.5731	1252
TEMPORAL	96.0913	96.3991	96.2450	92.7148	93.0117	92.8630	0.0000	0.0000	0.0000	2
TENSE	96.0925	96.4282	96.2601	92.8233	93.1476	92.9852	87.0968	76.5464	81.4815	388
TIME	96.1107	96.4185	96.2644	92.7148	93.0117	92.8630	0.0000	0.0000	0.0000	2
TYPE	96.0433	96.3603	96.2016	92.8219	93.1282	92.9748	89.2857	92.5926	90.9091	108
VTYPE	96.2350	96.5059	96.3702	93.0410	93.3029	93.1718	93.5426	89.6819	91.5716	1163

Table 5: Evaluation of the DOP model with training data annotated with atomic grammatical features.

curacy than the Tree-DOP model and (ii) allow us to learn grammatical features with a high degree of accuracy. In a number of experiments on the HomeCentre corpus, we investigated the veracity of this hypothesis. We generated a number of versions of this treebank with varying grammatical feature annotations, and trained and tested our model on these treebanks. We evaluated the output parses in terms of unlabelled parse accuracy, labelled parse accuracy and feature detection accuracy.

Our experiments show that annotating with multiple grammatical functions gives the greatest improvement in phrase-structure tree parse accuracy and that, overall, the GF-DOP model generally improves phrase-structure tree parse accuracy: 93.75% of the runs conducted gave improvements over the baseline in terms of labelled f-score. Our experiments also show that performance in terms of detecting grammatical features where feature occurrence is greater than 100 ranges between 68.7783% and 95.1879%, depending on the feature or group of features being tested and how often those features were seen in the training data.

In future work, we would like to incorporate available subcategorisation information into the model, perhaps by distinguishing between those functions which are subcategorised for and those which are not. We would like to scale also to larger corpora, in particular to be able to investigate features which were too infrequent in the data used here to be able to draw strong conclusions about their usefulness in this context. We intend to achieve this using the resources of Cahill et al. [2004]. Finally, we would like to investigate further the characteristics of the model's back-off element.

Acknowledgements

This work was generously supported by the Irish Research Council for Science and Technology (IRCSET) and Science Foundation Ireland (SFI).

References

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

Rens Bod. An Improved Parser for Data-Oriented Lexical-Functional Analysis. In *Proceedings of the 38th Conference of the Association for Computational Linguistics*, pages 61–68, Hong Kong, 2000a.

Rens Bod. An Empirical Evaluation of LFG-DOP. In *Proceedings of the 19th International Conference on Computational Linguistics*, pages 62–68, Saarbrucken, Germany, 2000b.

Rens Bod. *Enriching Linguistics with Statistics: Performance Models of Natural Language*. PhD thesis, Institute for Logic, Language and Computation, University of Amsterdam, The Netherlands, 1995.

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

Rens Bod and Ronald Kaplan. 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, 1998.

Aoife Cahill, Michael Burke, Ruth O'Donovan, Josef van Genabith, and Andy Way. Long-Distance Dependency Resolution in Automatically Acquired Wide-Coverage PCFG-Based LFG Approximations. In *Proceedings of the 42th Conference of the Association for Computational Linguistics*, pages 320–327, Barcelona, Spain, 2004.

Joshua Goodman. Efficient Parsing of DOP with PCFG-Reductions. In Rens Bod, Remko Scha, and Khalil Sima'an, editors, *Data-Oriented Parsing*, pages 125–146. Stanford CA.: CSLI Publications, 2003.

Joshua Goodman. Efficient Algorithms for Parsing the DOP model. In *Proceedings of the 1st Conference on Empirical Methods in Natural Language Processing (EMNLP 1)*, pages 143–152, Philadelphia, PA., 1996.

Joshua Goodman. *Parsing inside-out*. PhD thesis, Harvard University, MA., 1998.

Víctor M. Jiménez and Andrés Marzal. Computation of the N Best Parse Trees for Weighted and Stochastic Context-Free Grammars. In *Proceedings of the Joint IAPR International Workshops on Advances in Pattern Recognition*, pages 183–192, London, UK, 2000. Springer-Verlag.

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

Khalil Sima'an. *Learning Efficient Disambiguation*. PhD thesis, University of Amsterdam, The Netherlands, 1999.

Andy Way. A Hybrid Architecture for Robust MT using LFG-DOP. *Journal of Experimental and Theoretical Artificial Intelligence*, **11**:441–471, 1999.

Andreas Zollmann and Khalil Sima'an. A Consistent and Efficient Estimator for Data-Oriented Parsing. *Journal of Automata, Languages and Combinatorics (JALC)*, **10**(2/3):367–388, 2005.