DESIGNING FEATURES FOR PARSE DISAMBIGUATION AND REALISATION RANKING

Aoife Cahill Martin Forst and Christian Rohrer
Universität Stuttgart

Proceedings of the LFG07 Conference

Miriam Butt and Tracy Holloway King (Editors)

2007

CSLI Publications

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

Abstract

We present log-linear models for use in the tasks of parse disambiguation and realisation ranking in German. Forst (2007a) shows that by extending the set of features used in parse disambiguation to include more linguistically motivated information, disambiguation results can be significantly improved for German data. The question we address in this paper is to what extent this improved set of features can also be used in realisation ranking. We carry out a number of experiments on German newspaper text. In parse disambiguation, we achieve an error reduction of 51%, compared to an error reduction of 34.5% with the original model that does not include the additional features of Forst (2007a). In realisation ranking, BLEU score increases from 0.7306 to 0.7939, and we achieve a 10 point improvement in exact match over a baseline language model. This being said, our results also show that further features need to be taken into account for realisation ranking in order to improve the quality of the corresponding model.

1 Introduction

Statistical disambiguation of syntactic structures has been extensively studied in recent years. Riezler et al. (2002) have successfully applied a log-linear model based on features referring to simple, mostly locally restricted c- and f-structure configurations to the task of LFG parse disambiguation for English. However, recent studies suggest that these types of features are not sufficient for the disambiguation of languages with relatively free word order, such as Japanese (Yoshimura et al., 2003) or German.

Forst (2007a) shows that by extending the set of features used in parse disambiguation to include more linguistically motivated information, disambiguation results can be significantly improved for German data. The question we address in this paper is to what extent this improved set of features can also be used in realisation ranking.[1] It is clear that some features designed for parse disambiguation will not be useful for realisation ranking and vice versa. For example, features that capture lexical dependencies will not be useful in generation ranking, since lexical dependencies are given in this task. Conversely, the log-linear model for realisation ranking, where the task is to determine the most natural sounding sequence of words, will need features that refer (only) to the surface string, and those features are, of course, not interesting for parse disambiguation. Nevertheless, it is reasonable to assume that c-structure features or features that refer to c-structure and f-structure simultaneously are useful for both tasks, and that taking the angle of both tasks may help to identify relevant features.

We present a model for realisation ranking similar to that of Velldal and Oepen (2005). The main differences between our work and theirs is that we are working

[1]The work described in this paper has been carried out as part of the COINS project of the linguistic Collaborative Research Centre (SFB 732) at the University of Stuttgart, which is funded by the German Research Foundation (DFG). Furthermore, we would like to thank John Maxwell of the Palo Alto Research Center for being so responsive to our requests for extensions of the XLE generator functionalities, some of which were crucial for our work.

within the LFG framework and concentrating on a less configurational language: German.

2 System Setup

2.1 A Broad-Coverage LFG for German

For the construction of our data, we use the German broad-coverage LFG documented in Dipper (2003) and Rohrer and Forst (2006). It is a hand-crafted grammar developed in and for the LFG grammar development and processing platform XLE (Crouch et al., 2006). It achieves parsing coverage of about 80% in terms of full parses on newspaper text, and for sentences out of coverage, the robustness techniques described in Riezler et al. (2002) (i.e. fragment grammar, 'skimming') are employed for the construction of partial analyses. The grammar is reversible, which means that the XLE generator can produce surface realisations for well-formed input f-structures.

2.2 Parse Disambiguation

We use a standard log-linear model for carrying out parse disambiguation (Toutanova et al., 2002; Riezler et al., 2002; Miyao and Tsujii, 2002; Malouf and van Noord, 2004; van Noord, 2006; Clark and Curran, 2004). A key factor in the success of these models is feature design. As a baseline, we design features based on the property set used for the disambiguation of English ParGram LFG parses (Riezler et al., 2002; Riezler and Vasserman, 2004). These properties are based on thirteen property templates, which can be parameterised for any combination of c-structure categories or f-structure attributes and their values. Forst (2007a) shows that by extending this set of features used in parse disambiguation to include more linguistically motivated information, disambiguation results can be significantly improved for German data.

2.3 Surface Realisation

As XLE comes with a full-fledged generator, the grammar can be used both for parsing and for surface realisation.[2] Figure 2 shows the set of 18 strings that are generated from the f-structure in Figure 1. In this case, the German parser only produces one parse, and so there is no parse disambiguation necessary. However there is some work to be done in ranking the alternative string realisations for the input f-structure. Note that all of the surface realisations are grammatical; however, some of them are clearly more likely or unmarked than others.

[2] At the moment it is not possible to generate from packed structures where ambiguity is preserved. However, in the future we hope to be able to do so. This would be particularly useful in an application such as machine translation, where some ambiguities transfer across languages.

(1) Die Nato werde nicht von der EU geführt.
The NATO is not from the EU led.
'NATO is not led by the EU.'

```
"Die Nato werde nicht von der EU geführt."

[ PRED      'führen<[249:von], [21:Nato]>'
[           [PRED  'Nato'                                                ]
[           [CHECK [_SPEC-TYPE [_COUNT +, _DEF +, _DET attr]]
[           [      [_INFL      strong-det                    ]
[ SUBJ      [NTYPE [NSYN proper]                                        ]
[           [SPEC  [DET [PRED     'die'  ]]
[           [      [DET [DET-TYPE def    ]]
[        21 [CASE nom, GEND fem, NUM sg, PERS 3                         ]
[           [PRED   'von<[283:EU]>'                                     ]
[           [       [PRED  'EU'                                         ]
[           [       [CHECK [_SPEC-TYPE [_COUNT +, _DEF +, _DET attr]]
[           [       [      [_INFL      strong-det                    ]
[ OBL-AG    [ OBJ   [NTYPE [NSYN proper]                                ]
[           [       [SPEC  [DET [PRED     'die'  ]]
[           [       [      [DET [DET-TYPE def    ]]
[           [   283 [CASE dat, GEND fem, NUM sg, PERS 3                 ]
[       249 [PSEM dir, PTYPE sem                                        ]
[ ADJUNCT   { [PRED         'nicht'  ]}
[       215 {[ADJUNCT-TYPE neg      ]}
[           [_AUX-FORM {werden-pass}                                    ]
[ CHECK     [_VLEX     [_AUX-SELECT sein]                               ]
[           [_VMORPH   [_PARTICIPLE perfect]                            ]
[ TNS-ASP   [MOOD subjunctive, PASS-SEM dynamic, TENSE pres]
[ TOPIC     [21:Nato]
[  128 [CLAUSE-TYPE decl, PASSIVE +, STMT-TYPE decl, VTYPE main         ]
```

Figure 1: F-structure for (1)

Just as hand-crafted grammars, when used for parsing, are only useful for most applications when they have been complemented with a disambiguation module, their usefulness as a means of surface realisation depends on a reliable module for realisation ranking. A long list of arbitrarily ordered output strings is useless for practical applications such as summarisation, question answering, machine translation, etc.

```
Die Nato werde von der EU nicht geführt.
Die Nato werde nicht von der EU geführt.
Nicht von der EU geführt werde die Nato.
Nicht werde von der EU die Nato geführt.
Nicht werde die Nato von der EU geführt.
Nicht geführt werde von der EU die Nato.
Nicht geführt werde die Nato von der EU.
Von der EU nicht geführt werde die Nato.
Von der EU werde die Nato nicht geführt.
Von der EU werde nicht die Nato geführt.
Von der EU geführt werde nicht die Nato.
Von der EU geführt werde die Nato nicht.
Geführt werde die Nato nicht von der EU.
Geführt werde die Nato von der EU nicht.
Geführt werde nicht von der EU die Nato.
Geführt werde nicht die Nato von der EU.
Geführt werde von der EU nicht die Nato.
Geführt werde von der EU die Nato nicht.
```

Figure 2: The set of strings generated from the f-structure in Figure 1

Very regular preferences for certain realisation alternatives over others can be implemented by means of so-called optimality marks (Frank et al., 2001), which are implemented in XLE both for the parsing and the generation direction. For ranking string realisations on the basis of 'soft' and potentially contradictory constraints, however, the stochastic approach based on a log-linear model, as it has previously been implemented for English HPSGs (Nakanishi et al., 2005; Velldal and Oepen, 2005), seems more adequate.

3 Feature Design

3.1 Feature Design for Parse Disambiguation

Feature design for parse disambiguation is often carried out in a semi-automatic manner, i.e. by designing feature templates that are then instantiated automatically. Although the number of features built this way is often in the hundreds of thousands, nothing guarantees that the information relevant for disambiguation is actually captured by some feature(s). This is particularly true when the feature templates have been designed with little attention to typical ambiguities in the language under consideration. Forst (2007a) shows that linguistically motivated features that capture, e.g., the linear order of grammatical functions, the (surface and functional uncertainty path) distance of an extraposed constituent to its f-structure head, the nature of a DP in relation to its grammatical function (pronominal vs. full DP, animate vs. inanimate) etc. allow for a significantly improved disambiguation

Name of feature template and parameters	Explanation
Features used for the disambiguation of English ParGram LFG parses (Riezler et al., 2002; Riezler and Vasserman, 2004)	
`fs_attrs <attrs>`	counts number of occurrences of attribute(s) <*attrs*> in the f-structure
`cs_label <cat>`	counts number of occurrences of category <*cat*> in the c-structure
`fs_attr_val <attr> <val>`	counts number of times f-structure attribute <*attr*> has value <*val*>
`cs_num_children <cat>`	counts number of children of all nodes of category <*cat*>
`fs_adj_attrs <attr1> <attr2>`	counts the number of times feature <*attr2*> is immediately embedded in feature <*attr1*>
`fs_sub_attrs <attr1> <attr2>`	counts the number of times feature <*attr2*> is embedded somewhere in <*attr1*>
`cs_adjacent_label <cat1> <cat2>`	counts the number of <*cat1*> nodes that immediately dominate <*cat2*> nodes
`cs_sub_label <cat1> <cat2>`	counts the number of <*cat1*> nodes that (not necessarily immediately) dominate <*cat2*> nodes
`cs_embedded <cat> <Depth>`	counts the number of <*cat*> nodes that dominate (at least) <*Depth*> other <*cat*> nodes
`cs_conj_nonpar <Depth>`	counts the number of coordinated c-structures that are not parallel at <*Depth*> levels under the coordinated constituent
`lex_subcat <Lemma> <SCFs>`	counts the number of times <*Lemma*> occurs with one of the subcategorisation frames in <*SCFs*>
Additional Linguistically Motivated Features	
`ADD-PROP MOD1 <Lemma>`	counts the number of times a given lemma occurs as a member of a MOD set
`ADD-PROP F2 <Lemma> <PoS>`	counts the number of times a given lemma occurs as a particular <*PoS*>
`ADD-PROP ACTIVE/PASSIVE <Lemma>`	counts the number of times a (verb) lemma occurs in active/passive voice
`ADD-PROP isCommon/Def/ Pronoun/... <GF>`	determines whether a DP with function <*GF*> is common, definite, pronominal, etc.
`ADD-PROP DEP11 <PoS1> <Dep> <PoS2>`	counts the number of times a sub-f-structure of type <*PoS2*> is embedded into a (sub-)f-structure of type <*PoS1*> as its <*Dep*>
`ADD-PROP PATH`	counts given instantiations of functional uncertainty paths
`ADD-PROP PRECEDES <GF1> <GF2>`	counts the number of times a <*GF1*> precedes a <*GF2*> of the same (sub-)f-structure
`DISTANCE-TO-ANTECEDENT %X`	distance between a relative clause and its antecedent
`ADD-PROP DEP12 <PoS1> <Dep> <PoS2> <Lemma2>`	counts the number of times a sub-f-structure of type <*PoS2*> and with <*Lemma2*> as its PRED is embedded into a (sub-)f-structure of type <*PoS1*>as its <*Dep*>
`ADD-PROP DEP21 <PoS1> <Lemma1> <Dep> <PoS2>`	counts the number of times a sub-f-structure of type <*PoS2*> is embedded as its <*Dep*> into a (sub-)f-structure of type <*PoS1*> and with <*Lemma1*> as its PRED
`ADD-PROP PRECEDES <Lemma> <GF1> <GF2>`	counts the number of times a <*GF1*> subcategorised for by a PRED <*Lemma*> precedes a <*GF2*> subcategorised for by the same PRED
`ADD-PROP MOD2 <Lemma1> <Lemma2>`	counts the number of times <*Lemma2*> occurs in the MOD set of a (sub-)f-structure with <*Lemma1*> as its PRED
`ADD-PROP VADJUNCT_PRECEDES <Prep1> <Prep2>`	counts the numbers of times an ADJUNCT PP headed by <Prep1> precedes an ADJUNCT PP headed by <Prep2>, both being in an f-structure with a VTYPE
`ADD-PROP DEP22 <PoS1> <Lemma1> <Dep> <PoS2> <Lemma2>`	counts the number of times a sub-f-structure of type <*PoS2*> and with <*Lemma2*> as its PRED is embedded as its <*Dep*> into a <*Dep*> into a <*Dep*> into a (sub-)f-structure of type <*PoS1*> and with <*Lemma1*> as its PRED

Table 1: Feature templates used for semi-automatic feature construction for parse disambiguation

of German LFG parses. Many of these features are inspired by studies on "soft" syntactic constraints, which are most often formulated within an OT framework (Aissen, 2003; Bresnan et al., 2001), but can also be captured as features of more general probabilistic models (Snider and Zaenen, 2006). Table 1 gives a description of the main types of features used in parse disambiguation.

The evaluation of the log-linear model for parse disambiguation is described in more detail in Forst (2007a) and Forst (2007b), so here we will be brief. The model is trained on 8,881 partially labelled structures and tested on a test set of 1,497 sentences (with 371 sentences held out to fine-tune the log-linear model parameters). Table 2 gives a summary of the results broken down by dependency. The overall F-score is significantly better with the disambiguation model that includes the linguistically motivated additional features than the disambiguation model that relies on the XLE template-based properties only. Overall error reduction increases from 34.5% to 51.0%.

3.2 Feature Design for Realisation Ranking

Most traditional approaches to stochastic realisation ranking involve applying language model n-gram statistics to rank alternatives. However, n-grams alone are often not a good enough measure for ranking candidate strings. For example, for the f-structure associated with the string *Verheugen habe die Worte des Generalinspekteurs falsch interpretiert.* ('Verheugen had wrongly interpreted the words of the inspector general'.), 144 strings can be generated. The original string is ranked 7th among all candidate strings by our language model. There are several features in the input f-structure that we can use to improve the ranking of the desired string. The following features could be useful: (1) Linear order of functions (SUBJ generally precedes OBJ), (2) Adjunct position (sentence beginning, distance from the verb, etc.), (3) Partial VP fronting (generally marked and thus dispreferred).

(2) Verheugen habe die Worte des Generalinspekteurs falsch interpretiert.
Verheugen had the words the-GEN inspector-general wrongly interpreted.
'Verheugen had mis-interpreted the words of the inspector-general.'

grammatical relation/ morphosyntactic feature	upper bound F-sc.	stoch. select. all properties F-sc.	stoch. select. all properties err. red.	stoch. select. templ.-based pr. F-sc.	stoch. select. templ.-based pr. err. red.	lower bound F-sc.
all	85.50	83.01	51.0	82.17	34.5	80.42
PREDs only	79.36	75.74	46.5	74.69	31.0	72.59
`app` (close apposition)	63	60	63	61	75	55
`app_cl` (appositive clause)	53	53	100	52	86	46
`cc` (comparative complement)	28	19	-29	19	-29	21
`cj` (conjunct of coordination)	70	68	50	67	25	66
`da` (dative object)	67	63	67	62	58	55
`det` (determiner)	92	91	50	91	50	90
`gl` (genitive in specifier position)	89	88	75	88	75	85
`gr` (genitive attribute)	88	84	56	84	56	79
`mo` (modifier)	70	63	36	62	27	59
`mod` (non-head in compound)	94	89	29	89	29	87
`name_mod` (non-head in compl. name)	82	80	33	81	67	79
`number` (number as determiner)	83	81	33	81	33	80
`oa` (accusative object)	78	75	77	69	31	65
`obj` (argument of prep. or conj.)	90	88	50	87	25	86
`oc_fin` (finite clausal object)	67	64	0	64	0	64
`oc_inf` (infinite clausal object)	83	82	0	82	0	82
`op` (prepositional object)	57	54	40	54	40	52
`op_dir` (directional argument)	30	23	13	23	13	22
`op_loc` (local argument)	59	49	29	49	29	45
`pd` (predicative argument)	62	60	50	59	25	58
`pred_restr` (lemma of nom. adj.)	92	87	62	84	38	79
`quant` (quantifying determiner)	70	68	33	68	33	67
`rc` (relative clause)	74	62	20	59	0	59
`sb` (subject)	76	73	63	71	38	68
`sbp` (logical subj. in pass. constr.)	68	63	62	61	46	55
`case`	87	85	75	83	50	79
`comp_form` (complementizer form)	74	72	0	74	100	72
`coord_form` (coordinating conj.)	86	86	100	86	100	85
`degree`	89	88	50	87	0	87
`det_type` (determiner type)	95	95	–	95	–	95
`fut` (future)	86	86	–	86	–	86
`gend` (gender)	92	90	60	89	40	87
`mood`	90	90	–	90	–	90
`num` (number)	91	89	50	89	50	87
`pass_asp` (passive aspect)	80	80	100	79	0	79
`perf` (perfect)	86	85	0	86	100	85
`pers` (person)	85	84	83	82	50	79
`pron_form` (pronoun form)	73	73	–	73	–	73
`pron_type` (pronoun type)	71	70	0	71	100	70
`tense`	92	91	0	91	0	91

Table 2: F-scores (in %) in the 1,497 TiGer DB examples of our test set

"Verheugen habe die Worte des Generalinspekteurs falsch interpretiert

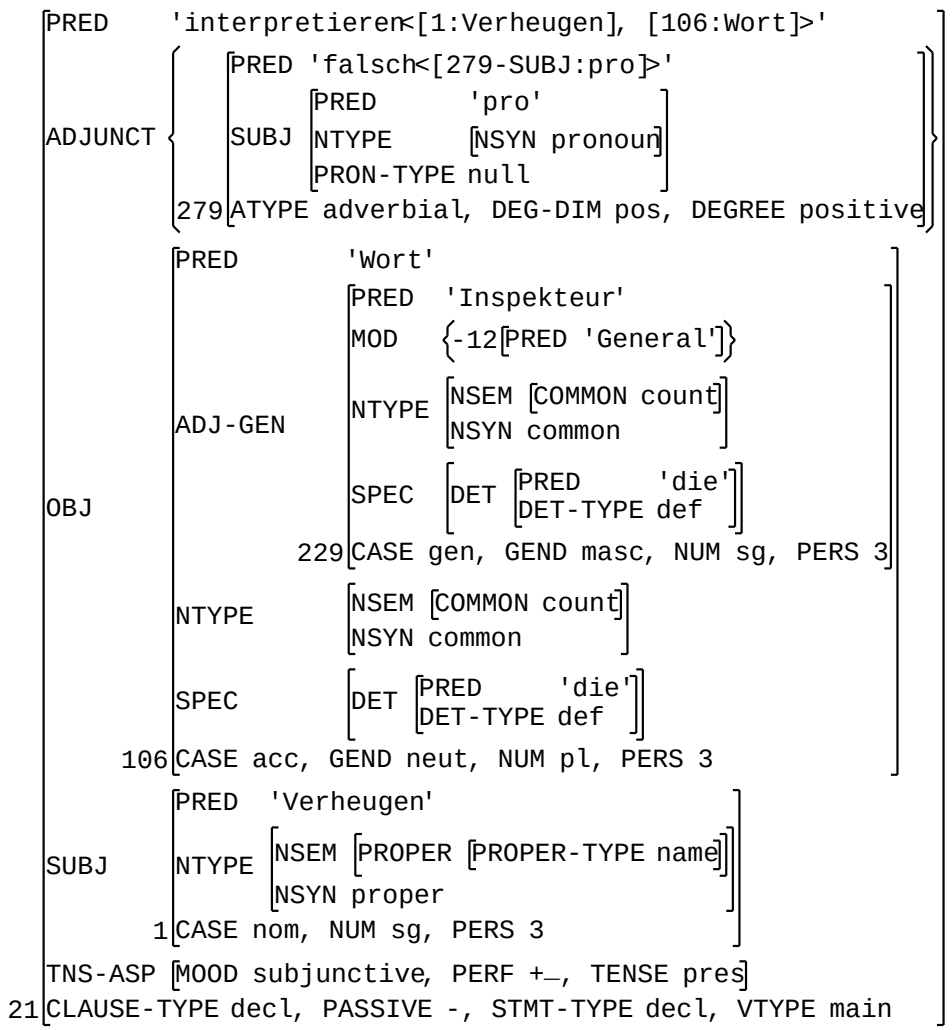

Figure 3: F-structure for (2)

```
1. Falsch interpretiert habe die Worte des
      Generalinspekteurs Verheugen.
2. Falsch interpretiert habe die Worte des
      Generalinspekteures Verheugen.
3. Die Worte des Generalinspekteurs falsch
      interpretiert habe Verheugen.
5. Die Worte des Generalinspekteurs habe Verheugen
      falsch interpretiert.
7. Verheugen habe die Worte des Generalinspekteurs
      falsch interpretiert.
11. Falsch interpretiert habe Verheugen die Worte des
      Generalinspekteurs.
15. Die Worte des Generalinspekteurs interpretiert habe
      Verheugen falsch.
17. Interpretiert habe die Worte des Generalinspekteurs
      Verheugen falsch.
```

Using the feature templates presented in Riezler et al. (2002), Riezler and Vasserman (2004) and Forst (2007a), we construct a list of 186,731 features that can be used for training our log-linear model.[3] Out of these, only 1,471 actually occur in our training data. In the feature selection process of our training regime (see Subsection 4.2), 360 features are chosen as the most discriminating; these are used to rank alternative solutions when the model is applied. Table 3 gives a list of the types of features used for realisation ranking.

We divide the features into three distinct categories: language model features (LM), c-structure features (CF) and additional features (AF). For realisation ranking, we do not use f-structure features, since the f-structure is given in the input. Examples of c-structure features are the number of times a particular category label occurs in a given c-structure, the number of children the nodes of a particular category have, or the number of times one particular category label dominates another. Examples of features that take both c- and f-structure information into account are the relative order of grammatical functions (e.g. 'SUBJ precedes OBJ'). As in Velldal and Oepen (2005), we incorporate the language model score associated with the string realisation for a particular structure as a feature in our model.

[3]For technical reasons, we were not able to include all the additional features we would have liked to include. For example, we could not use features that capture the relative order of ADJUNCT PPs headed by given prepositions.

Name of feature template and parameters	Explanation
C-structure Features	
cs_label <cat>	counts number of occurrences of category <*cat*> in the c-structure
cs_right_branch	counts number of right children
cs_num_children <cat>	counts number of children of all nodes of category <*cat*>
cs_adjacent_label <cat1> <cat2>	counts the number of <*cat1*> nodes that immediately dominate <*cat2*> nodes
cs_sub_label <cat1> <cat2>	counts the number of <*cat1*> nodes that (not necessarily immediately) dominate <*cat2*> nodes
cs_embedded <cat> <Depth>	counts the number of <*cat*> nodes that dominate (at least) <*Depth*> other <*cat*> nodes
cs_conj_nonpar <Depth>	counts the number of coordinated c-structures that are not parallel at <*Depth*> levels under the coordinated constituent
Additional Linguistically Motivated Features	
ADD-PROP PATH	counts given instantiations of functional uncertainty paths
ADD-PROP PRECEDES <GF1> <GF2>	counts the number of times a <*GF1*> precedes a <*GF2*> of the same (sub-)f-structure
ADD-PROP PRECEDES <Lemma> <GF1> <GF2>	counts the number of times a <*GF1*> subcategorised for by a PRED <*Lemma*> precedes a <*GF2*> subcategorised for by the same PRED
DISTANCE-TO-ANTECEDENT %X	distance between a relative clause and its antecedent
Language Model Features	
GEN_NGRAM_SCORE %X	3-gram language model score assigned to the generated sentence
GEN_WORD_COUNT %X	number of words in the generate sentence

Table 3: Feature templates used for semi-automatic feature construction in realisation ranking

4 Realisation Ranking Experimental Setup

4.1 Data

We use the TIGER Treebank (Brants et al., 2002) to train and test our model. It consists of just over 50,000 annotated sentences of German newspaper text. The sentences have been annotated with morphological and syntactic information in the form of functionally labelled graphs that may contain crossing and secondary edges.

We split the data into training and test data using the same data split as in Forst (2007a), i.e. sentences 8,001–10,000 of the TIGER Treebank are reserved for evaluation. Within this section, we have 422 TIGER-annotation-compatible f-structures, which are further divided into 86 development and 336 test structures. We use the development set to tune the parameters of the log-linear model. Of the 86 heldout sentences and the 336 test sentences, 78 and 323 respectively are of length >3 and hence are actually used for our final evaluation.

For training, we build a symmetric treebank of 8,609 packed c/f-structure representations in a similar manner to Velldal et al. (2004). We do not include struc-

tures for which only one string is generated, since the log-linear model for realisation ranking cannot learn anything from them. The symmetric treebank was established using the following strategy:

1. Parse the input sentence from the TIGER Treebank.
2. Select all of the analyses that are compatible with the TIGER Treebank annotation.
3. Of all the TIGER-compatible analyses, choose the most likely c-/f-structure pair according to the log-linear model for parse disambiguation.
4. Generate from the f-structure part of this analysis.
5. If the input string is contained in the set of output strings, add this sentence and all of its corresponding c-/f-structure pairs to the training set. The pair(s) that correspond(s) to the original corpus sentence is/are marked as the intended structure(s), while all others are marked as unintended.

Theoretically all strings that can be parsed should be generated by the system, but for reasons of efficiency, punctuation is often not generated in all possible positions, therefore resulting in an input string not being contained in the set of output strings. Whenever this is the case for a given sentence, the c-/f-structure pairs associated with it cannot be used for training. Evaluation can be carried out regardless of this problem, but it has to be kept in mind that the original corpus string cannot be generated for all input f-structures. In our test set, it is generated for only 62% of them.

Tables 4 and 5 give information about the ambiguity of the training and test data. For example, in the training data there are 1,206 structures with more than 100 string realisations. Most of the training and test structures have between 2 and 50 possible (and grammatical) string realisations. The average sentence length of the training data is 11.3 and it is 12.8 for the test data.[4] The tables also show that the structures with more potential string realisations correspond to longer sentences than the structures that are less ambiguous when generating.

4.2 Training

We train a log-linear model that maximises the conditional probability of the observed corpus sentence given the corresponding f-structure. The model is trained in a (semi-)supervised fashion on the 8,609 (partially) labelled structures of our training set using the `cometc` software provided with the XLE platform. `cometc` performs maximum likelihood estimation on standardised feature values and offers

[4]This is lower than the overall average sentence length of roughly 16 in TIGER because of the restriction that the structure produced by the reversible grammar for any TIGER sentence be compatible with the original TIGER graph. As the grammar develops further, we hope that longer sentences can be included in both training and test data.

String Realisations	# of Strings	Average # of Words
> 100	1206	18.3
$\geq 50, < 100$	709	14.3
$\geq 10, < 50$	3029	11.8
$> 1, < 10$	3665	7.6
Total	8609	11.3

Table 4: Number of structures and average sentence length according to ambiguity classes in the training set

String Realisations	# of Strings	Average # of Words
> 100	61	23.7
$\geq 50, < 100$	26	13.5
$\geq 10, < 50$	120	11.6
$> 1, < 10$	129	7.8
Total	336	12.8

Table 5: Number of structures and average sentence length according to ambiguity classes in the test set

several regularisation and/or feature selection techniques. We apply the combined method of incremental feature selection and l_1 regularisation presented in Riezler and Vasserman (2004), the corresponding parameters being adjusted on our heldout set.

For technical reasons, the training was carried out on unpacked structures. However, we hope to be able to train and test on packed structures in the future, which will greatly increase efficiency.

5 Analysis of Results by Feature Type

Given the three distinct types of features in Table 3, we carry out a number of smaller experiments on our heldout set, only training on a subset of features each time. This is done in order to see what effect each group of features has on the overall performance of the log-linear model, and to see what combination of feature types performs best. We evaluate the most likely string produced by our system in terms of two metrics: **exact match** and **BLEU score** (Papineni et al., 2002). Exact match measures what percentage of the most probable strings are exactly identical to the string from which the input structure was produced. BLEU score is a more relaxed metric which measures the similarity between the selected string realisation and the observed corpus string.

The results are given in Table 6. The results show that training on c-structure features alone achieves the worst exact match and BLEU score. This is possibly due to the nature of the c-structure features used, which were initially designed for parse disambiguation. Therefore, future work is required to investigate whether

	Exact Matches (%)	BLEU Score
Baseline	24	0.7291
LM	23	0.7034
CF	22	0.6824
AF	23	0.7060
LM + CF	27	0.7529
LM + AF	33	0.7705
CF + AF	33	0.7303
LM + CF + AF	35	0.7808

Table 6: Results on the heldout set of training only on subsets of feature types

c-structure features more appropriate for realisation ranking can be devised. Training on language model features alone, or additional features alone, also does not achieve very high results. Surprisingly, the log-linear model trained on language model features alone performs worse than the baseline language model applied directly. We cannot be sure what causes this, but one possible reason is that the number of words is taken into account as a feature in the log-linear model, while the language model does not use this feature. Another reason might be that because we are working with unpacked structures, we loose a lot of precision with the log-linear model, so that often more than one solution is ranked highest. When this happens, we choose a solution at random, which may not always reflect the original language model scores. This problem generally does not arise with the language model which assigns more precise scores. However, the combination of language model features and additional features is the one that leads to the greatest improvement in exact match and BLEU scores. It achieves a BLEU score of 0.7705, which is only a little less than the best result achieved by combining all three feature types. The results thus suggest that the language model features and the additional features contribute most to the model, while the c-structure features contribute less. Nevertheless, the c-structure features are beneficial, since the best results are achieved by combining the three feature types.

6 Final Evaluation

We first rank the generator output with a language model trained on the Huge German Corpus (a collection of 200 million words of newspaper and other text) using the SRILM toolkit. The results are given in Table 7, achieving exact match of 27% and BLEU score of 0.7306 on the test set. In comparison to the results reported by Velldal and Oepen (2005) for a similar experiment on English, these results are markedly lower, presumably because of the relatively free word order of German.

We then rank the output of the generator with our log-linear model as described

Exact Match Upper Bound	62%
Exact Matches	27%
BLEU score	0.7306

Table 7: Results on the test set with the language model

above and give the results in Table 8. There is a noticeable improvement in quality. Exact match increases from 27% to 37%, which corresponds to an error reduction of 29%,[5] and BLEU score increases from 0.7306 to 0.7939.

Exact Match Upper Bound	62%
Exact Matches	37%
BLEU score	0.7939

Table 8: Results on the test set with the log-linear model

There is very little comparable work on realization ranking for German. Gamon et al. (2002) present work on learning the contexts for a particular subset of linguistic operations; however, no evaluation of the overall system is given. The work that comes closest to ours is that of Filippova and Strube (2007) who present a two-step algorithm for determining constituent order in German. They predict the surface order of the major non-verbal constituents in a German sentence, given its dependency representation. They do not predict the position of the verb or the order within constituents, nor do they generate word forms from lemmas followed by morphological tags. Training and evaluation is carried out on Wikipedia data and their algorithm outperforms four baseline models. They achieve an exact match metric of 61%, i.e. for 61% of their corpus sentences, the order of the major constituents generated matches the original order. At first sight, this result looks very superior to the exact match metric of 37% we achieve, but when we take into account that our upper bound for exact match is 62% as opposed to theirs of 100%, the results become comparable. Furthermore, it has to be taken into account that many of the mismatches that we are penalized for result from generated word forms that diverge from the forms in the corpus, a problem Filippova and Strube (2007) do not deal with at all. This being said, this recent publication provides us with many useful ideas of how to design further features relevant for the task of realization ranking.

[5] Remember that the original corpus string is generated from only 62% of the f-structures of our test set, which fixes the upper bound for exact match at 62% rather than 100%.

7 Error Analysis

We had initially expected the increase in BLEU score to be greater than 0.0633, since German is far less configurational than English and therefore we thought the syntactic features used in the log-linear model would play an even greater role in realisation ranking. However, in our experiments, the improvement was only slightly greater than the improvement achieved by Velldal and Oepen (2005). In this section, we present some of the more common errors that our system still produces.

Word Choice Often there is more than one surface realisation for a particular sequence of morphological tags. Sometimes the system chooses an incorrect form for the sentence context, and sometimes it chooses a valid, though marked or dispreferred, form. For example, from the structure in Figure 3, the system chooses the following string as the most probable.

Verheugen habe die **Wörter** des **Generalinspekteures** falsch interpretiert.
Verheugen had the **words** of the **inspector-general** wrongly interpreted.

There are two mismatches in this output string with respect to the original corpus string. In the first case the system has chosen *Wörter* as the surface realisation for the morpheme sequence *Wort+NN.Neut.NGA.Pl* rather than the, in this case, correct form *Worte*. The difference between the two realisations is semantic; they both translate as *words* in English, but *Worte* is a more abstract concept referring to a meaningful stretch of text or speech, whereas *Wörter* is more concrete and can refer, e.g., to the words in a dictionary.

In the second (less critical) case, the system has chosen to mark the genitive case of *Generalinspekteur* with *es* rather than the *s* that is in the original corpus sentence. This is a relatively frequent alternation that is difficult to predict, and there are other similar alternations in the dative case, for example.

The second case is merely a phonological variation and does not alter the projected meaning. The first case, however, is completely incorrect and should not be generated. To correctly generate only *Worte* in this instance, the morphological component of the system needs to be improved. The most obvious solution is to have different lemmas for the different senses of (the plural of) *Wort*. In order to improve the selection of the most natural variant of the genitive and dative markings, one solution might be to try and learn the most frequent variant for a given lemma based on corpus statistics.

Placement of adjuncts Currently, there is no feature that captures the (relative) location of particular types of adjuncts. In German, there is a strong tendency for temporal adjuncts to appear early in the sentence, for example. Since the system was not provided with data from which it could learn this generalisation, it generated output like the following:

Frauenärzte haben die Einschränkung umstrittener Antibabypillen
Gynaecologists have the restriction controversial birth control pills
wegen erhöhter Thrombosegefahr **am Dienstag** kritisiert.
because of increased risk of thrombosis **on Tuesday** critisised.
'Gynaecologists criticised the restriction on controversial birth control pills due to increased risk of thrombosis on Tuesday.'

where the temporal adjunct *on Tuesday* was generated very late in the sentence, resulting in a highly marked utterance.

Discourse Information In many cases, the particular subtleties of an utterance can only be generated using knowledge of the context in which it occurs. For example, the following sentence appears in our development corpus:

Israel stellt den Friedensprozess nach Rabins Tod nicht in Frage
Israel puts the peace process after Rabin's death not in question
'Israel does not challenge the peace process after Rabin's death'

Our system generates the string:

Nach Rabins Tod stellt Israel den Friedensprozess nicht in Frage.
After Rabin's death puts Israel the peace process not in question.

which, taken on its own, gets a BLEU score of 0. The sentence produced by our system is a perfectly valid sentence and captures essentially the same information as the original corpus sentence. However, without knowing anything about the information structure within which this sentence is uttered, we have no way of telling where the emphasis of the sentence is.

7.1 Additional Features

It is clear from the errors outlined above that further features are required in order to achieve improved realisation ranking. For example, a feature is required that captures the placement of adjunct types so that the tendency of temporal adjuncts to appear before locatives is captured correctly. Including information structure features is also necessary for the improvement of the overall system. The work described in this paper is part of a much larger project, and future research is already planned to integrate information structure into the surface realisation process. It is yet to be seen whether these features could also be useful in parse disambiguation.

8 Conclusion

In this paper, we have presented the features used in log-linear models for parse disambiguation and realisation ranking for a large-scale German LFG. We train both parse disambiguation and realisation ranking systems on over 8,000 partially labelled structures and test on a heldout section of almost 2,000 sentences. In the parse disambiguation experiments, we achieve an increase in error reduction of 16.5 points with the additional features over the simple template-based features used in the parse disambiguation of English (Forst, 2007b). In the task of realisation ranking, we achieve an increase in exact match score from 27% to 37% and an increase in BLEU score from 0.7306 to 0.7939 over a baseline language model trained on a large corpus of German. We thus show that linguistically motivated features that were initially developed for the task of parse disambiguation carry over rather well to the task of realisation ranking. Despite these encouraging results, an error analysis of the realisation ranking shows that further features are required by the log-linear model in order to improve the quality of the output strings. It is also unclear how suitable the BLEU score as an evaluation metric is, and further research into other metrics and a comparison with human evaluation is necessary.

References

Aissen, Judith. 2003. Differential Object Marking: Iconicity vs. Economy. *Natural Language and Linguistic Theory* .

Brants, Sabine, Dipper, Stefanie, Hansen, Silvia, Lezius, Wolfgang and Smith, George. 2002. The TIGER Treebank. In *Proceedings of the Workshop on Treebanks and Linguistic Theories*, Sozopol, Bulgaria.

Bresnan, Joan, Dingare, S and Manning, Christopher. 2001. Soft Constraints Mirror Hard Constraints. In *LFG 2001*.

Clark, Stephen and Curran, James R. 2004. Parsing the WSJ using CCJ and Log-Linear Models. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL '04)*, Barcelona, Spain.

Crouch, Dick, Dalrymple, Mary, Kaplan, Ron, King, Tracy, Maxwell, John and Newman, Paula. 2006. XLE Documentation. Technical Report, Palo Alto Research Center, CA.

Dipper, Stefanie. 2003. *Implementing and Documenting Large-scale Grammars – German LFG*. Ph. D.thesis, IMS, University of Stuttgart.

Filippova, Katja and Strube, Michael. 2007. Generating Constituent Order in German Clauses. In *Proceedings of the 45th Annual Meeting of the Association of*

Computational Linguistics, pages 320–327, Prague, Czech Republic: Association for Computational Linguistics.

Forst, Martin. 2007a. *Disambiguation for a Linguistically Precise German Parser*. Ph. D.thesis, University of Stuttgart.

Forst, Martin. 2007b. Filling Statistics with Linguistics – Property Design for the Disambiguation of German LFG Parses. In *Proceedings of the ACL Workshop on Deep Linguistic Processing*, Prague, Czech Republic.

Frank, Anette, King, Tracy Holloway, Kuhn, Jonas and Maxwell, John T. 2001. Optimality Theory Style Constraint Ranking in Large-Scale LFG Grammars. In Peter Sells (ed.), *Formal and Empirical Issues in Optimality Theoretic Syntax*, pages 367–397, Stanford, CA: CSLI Publications.

Gamon, Michael, Ringger, Eric, Corston-Oliver, Simon and Moore, Robert. 2002. Machine-learned contexts for linguistic operations in German sentence realization. In *Proceedings of ACL 2002*, pages 25–32, Philadelphia, PA.

Malouf, Robert and van Noord, Gertjan. 2004. Wide Coverage Parsing with Stochastic Attribute Value Grammars. In *Proceedings of the IJCNLP-04 Workshop "Beyond Shallow Analyses - Formalisms and statistical modeling for deep analyses"*.

Miyao, Yusuke and Tsujii, Jun'ichi. 2002. Maximum entropy estimation for feature forests. In *Proceedings of the Human Language Technology Conference*, San Diego, CA.

Nakanishi, Hiroko, Miyao, Yusuke and Tsujii, Jun'ichi. 2005. Probabilistic models for disambiguation of an HPSG-based chart generator. In *Proceedings of IWPT 2005*.

Papineni, Kishore, Roukos, Salim, Ward, Todd and Zhu, WeiJing. 2002. BLEU: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of ACL 2002*, pages 311–318, Philadelphia, PA.

Riezler, Stefan, King, Tracy Holloway, Kaplan, Ronald M., Crouch, Richard, Maxwell, John T. and Johnson, Mark. 2002. Parsing the Wall Street Journal using a Lexical-Functional Grammar and Discriminative Estimation Techniques. In *Proceedings of ACL 2002*, Philadelphia, PA.

Riezler, Stefan and Vasserman, Alexander. 2004. Gradient feature testing and l_1 regularization for maximum entropy parsing. In *Proceedings of EMNLP'04*, Barcelona, Spain.

Rohrer, Christian and Forst, Martin. 2006. Improving coverage and parsing quality of a large-scale LFG for German. In *Proceedings of LREC-2006*, Genoa, Italy.

Snider, Neil and Zaenen, Annie. 2006. Animacy and Syntactic Structure: Fronted NPs in English. In *Intelligent Linguistic Architectures – Variations on Themes by Ronald M. Kaplan*, CSLI Publications.

Toutanova, Kristina, Manning, Christopher D., Shieber, Stuart M., Flickinger, Dan and Oepen, Stephan. 2002. Parse Disambiguation for a Rich HPSG Grammar. In *First Workshop on Treebanks and Linguistic Theories (TLT2002)*, pages 253–263.

van Noord, Gertjan. 2006. At Last Parsing Is Now Operational. In Piet Mertens, Cedrick Fairon, Anne Dister and Patrick Watrin (eds.), *TALN06. Verbum Ex Machina. Actes de la 13e conférence sur le traitement automatique des langues naturelles*, pages 20–42, Leuven, Belgium.

Velldal, Erik and Oepen, Stephan. 2005. Maximum Entropy Models for Realization Ranking. In *Proceedings of the 10th MT Summit*, pages 109–116, Thailand.

Velldal, Erik, Oepen, Stephan and Flickinger, Dan. 2004. Paraphrasing Treebanks for Stochastic Realization Ranking. In *Proceedings of TLT Workshop*, pages 149–160, Tübingen, Germany.

Yoshimura, H, Masuichi, Hiroshi, Ohkuma, Tomoko and Sugihara, Daigo. 2003. Disambiguation of F-Structures based on Support Vector Machines. *Information Processing Society of Japan SIG Notes* pages 75–80.