Hard Constraints for Grammatical Function Labelling

Wolfgang Seeker
University of Stuttgart
Institut für Maschinelle Sprachverarbeitung
seeker@ims.uni-stuttgart.de

Ines Rehbein
University of Saarland
Dep. for Comp. Linguistics & Phonetics
rehbein@coli.uni-sb.de

Jonas Kuhn
University of Stuttgart
Institut für Maschinelle Sprachverarbeitung
jonas@ims.uni-stuttgart.de

Josef van Genabith
Dublin City University
CNGL and School of Computing
josef@computing.dcu.ie

Abstract

For languages with (semi-) free word order (such as German), labelling grammatical functions on top of phrase-structural constituent analyses is crucial for making them interpretable. Unfortunately, most statistical classifiers consider only local information for function labelling and fail to capture important restrictions on the distribution of core argument functions such as subject, object etc., namely that there is at most one subject (etc.) per clause. We augment a statistical classifier with an integer linear program imposing hard linguistic constraints on the solution space output by the classifier, capturing global distributional restrictions. We show that this improves labelling quality, in particular for argument grammatical functions, in an intrinsic evaluation, and, importantly, grammar coverage for treebank-based (Lexical-Functional) grammar acquisition and parsing, in an extrinsic evaluation.

1 Introduction

Phrase or constituent structure is often regarded as an analysis step guiding semantic interpretation, while grammatical functions (i. e. subject, object, modifier etc.) provide important information relevant to determining predicate-argument structure.

In languages with restricted word order (e. g. English), core grammatical functions can often be recovered from configurational information in constituent structure analyses. By contrast, simple constituent structures are not sufficient for less configurational languages, which tend to encode grammatical functions by morphological means

(Bresnan, 2001). Case features, for instance, can be important indicators of grammatical functions. Unfortunately, many of these languages (including German) exhibit strong syncretism where morphological cues can be highly ambiguous with respect to functional information.

Statistical classifiers have been successfully used to label constituent structure parser output with grammatical function information (Blaheta and Charniak, 2000; Chrupała and Van Genabith, 2006). However, as these approaches tend to use only limited and local context information for learning and prediction, they often fail to enforce simple yet important global linguistic constraints that exist for most languages, e. g. that there will be at most one subject (object) per sentence/clause.[1]

"Hard" linguistic constraints, such as these, tend to affect mostly the "core grammatical functions", i. e. the argument functions (rather than e. g. adjuncts) of a particular predicate. As these functions constitute the core meaning of a sentence (as in: who did what to whom), it is important to get them right. We present a system that adds grammatical function labels to constituent parser output for German in a postprocessing step. We combine a statistical classifier with an integer linear program (ILP) to model non-violable global linguistic constraints, restricting the solution space of the classifier to those labellings that comply with our set of global constraints. There are, of course, many other ways of including functional information into the output of a syntactic parser. Klein and Manning (2003) show that merging some linguistically motivated function labels with specific syntactic categories can improve the performance of a PCFG model on Penn-II En-

---

[1]Coordinate subjects/objects form a constituent that functions as a joint subject/object.

glish data.[2] Tsarfaty and Sim'aan (2008) present a statistical model (Relational-Realizational Parsing) that alternates between functional and configurational information for constituency tree parsing and Hebrew data. Dependency parsers like the MST parser (McDonald and Pereira, 2006) and Malt parser (Nivre et al., 2007) use function labels as core part of their underlying formalism. In this paper, we focus on phrase structure parsing with function labelling as a post-processing step.

Integer linear programs have already been successfully used in related fields including semantic role labelling (Punyakanok et al., 2004), relation and entity classification (Roth and Yih, 2004), sentence compression (Clarke and Lapata, 2008) and dependency parsing (Martins et al., 2009). Early work on function labelling for German (Brants et al., 1997) reports 94.2% accuracy on gold data (a very early version of the TiGer Treebank (Brants et al., 2002)) using Markov models. Klenner (2007) uses a system similar to – but more restricted than – ours to label syntactic chunks derived from the TiGer Treebank. His research focusses on the correct selection of predefined subcategorisation frames for a verb (see also Klenner (2005)). By contrast, our research does not involve subcategorisation frames as an external resource, instead opting for a less knowledge-intensive approach. Klenner's system was evaluated on gold treebank data and used a small set of 7 dependency labels. We show that an ILP-based approach can be scaled to a large and comprehensive set of 42 labels, achieving 97.99% label accuracy on gold standard trees. Furthermore, we apply the system to automatically parsed data using a state-of-the-art statistical phrase-structure parser with a label accuracy of 94.10%. In both cases, the ILP-based approach improves the quality of argument function labelling when compared with a non-ILP-approach. Finally, we show that the approach substantially improves the quality and coverage (from 93.6% to 98.4%) of treebank-based Lexical-Functional Grammars for German over previous work in Rehbein and van Genabith (2009).

The paper is structured as follows: Section 2 presents basic data demonstrating the challenges presented by German word order and case syncretism for the function labeller. Section 3 describes the labeller including the feature model of the classifier and the integer linear program used to pick the correct labelling. The evaluation part (Section 4) is split into an intrinsic evaluation measuring the quality of the labelling directly using the German TiGer Treebank (Brants et al., 2002), and an extrinsic evaluation where we test the impact of the constraint-based labelling on treebank-based automatic LFG grammar acquisition.

2 Data

Unlike English, German exhibits a relatively free word order, i. e. in main clauses, the verb occupies second position (the last position in subordinated clauses) and arguments and adjuncts can be placed (fairly) freely. The grammatical function of a noun phrase is marked morphologically on its constituting parts. Determiners, pronouns, adjectives and nouns carry case markings and in order to be well-formed, all parts of a noun phrase have to agree on their case features. German uses a nominative–accusative system to mark predicate arguments. Subjects are marked with nominative case, direct objects carry accusative case. Furthermore, indirect objects are mostly marked with dative case and sometimes genitive case.

(1) *Der Löwe   gibt   dem Wolf   einen Besen.*
     NOM   DAT   ACC
     the lion   gives the wolf   a broom
     The lion gives a broom to the wolf.

(1) shows a sentence containing the ditransitive verb *geben* (to give) with its three arguments. Here, the subject is unambiguously marked with nominative case (NOM), the indirect object with dative case (DAT) and the direct object with accusative case (ACC). (2) shows possible word orders for the arguments in this sentence.[3]

(2) Der Löwe gibt einen Besen dem Wolf.
     Dem Wolf gibt der Löwe einen Besen.
     Dem Wolf gibt einen Besen der Löwe.
     Einen Besen gibt der Löwe dem Wolf.
     Einen Besen gibt dem Wolf der Löwe.

Since all permutations of arguments are possible, there is no chance for a statistical classifier to decide on the correct function of a noun phrase by its position alone. Introducing adjuncts to this example makes matters even worse.

---

[2]Table 6 shows that for our data a model with merged category and function labels (but without hard constraints!) performs slightly worse than the ILP approach developed in this paper.

[3]Note that although (apart from the position of the finite verb) there are no syntactic restrictions on the word order, there are restrictions pertaining to phonological or information structure.

Case information for a given noun phrase can give a classifier some clue about the correct argument function, since functions are strongly related to case values. Unfortunately, the German case system is complex (see Eisenberg (2006) for a thorough description) and exhibits a high degree of case syncretism. (3) shows a sentence where both argument NPs are ambiguous between nominative or accusative case. In such cases, additional semantic or contextual information is required for disambiguation. A statistical classifier (with access to local information only) runs a high risk of incorrectly classifying both NPs as subjects, or both as direct objects or even as nominal predicates (which are also required to carry nominative case). This would leave us with uninterpretable results. Uninterpretability of this kind can be avoided if we are able to constrain the number of subjects and objects globally to one per clause.[4]

(3) *Das Schaf   sieht   das Mädchen.*
     NOM/ACC   NOM/ACC
     the sheep   sees   the girl

     EITHER The sheep sees the girl
     OR The girl sees the sheep.

3 Grammatical Function Labelling

Our function labeller was developed and tested on the TiGer Treebank (Brants et al., 2002). The TiGer Treebank is a phrase-structure and grammatical function annotated treebank with 50,000 newspaper sentences from the *Frankfurter Rundschau* (Release 2, July 2006). Its overall annotation scheme is quite flat to account for the relatively free word order of German and does not allow for unary branching. The annotations use non-projective trees modelling long distance dependencies directly by crossing branches. Words are lemmatised and part-of-speech tagged with the *Stuttgart-Tübingen Tag Set (STTS)* (Schiller et al., 1999) and contain morphological annotations (Release 2). TiGer uses 25 syntactic categories and a set of 42 function labels to annotate the grammatical function of a phrase.

The function labeller consists of two main components, a maximum entropy classifier and an integer linear program. This basic architecture was introduced by Punyakanok et al. (2004) for the task of semantic role labelling and since then has been applied to different NLP tasks without significant changes. In our case, its input is a bare tree

---

[4]Although the classifier may, of course, still identify the wrong phrase as subject or object.

structure (as obtained by a standard phrase structure parser) and it outputs a tree structure where every node is labelled with the grammatical relation it bears to its mother node. For each possible label and for each node, the classifier assigns a probability that this node is labelled by this label. This results in a complete probability distribution over all labels for each node. An integer linear program then tries to find the optimal overall tree labelling by picking for each node the label with the highest probability without violating any of its constraints. These constraints implement linguistic rules like the *one-subject-per-sentence* rule mentioned above. They can also be used to capture treebank particulars, such as for example that punctuation marks never receive a label.

3.1 The Feature Model

Maximum entropy classifiers have been used in a wide range of applications in NLP for a long time (Berger et al., 1996; Ratnaparkhi, 1998). They usually give good results while at the same time allowing for the inclusion of arbitrarily complex features. They also have the advantage that they directly output probability distributions over their set of labels (unlike e. g. SVMs).

The classifier uses the following features:

- the lemma (if terminal node)
- the category (the POS for terminal nodes)
- the number of left/right sisters
- the category of the two left/right sisters
- the number of daughters
- the number of terminals covered
- the lemma of the left/right corner terminal
- the category of the left/right corner terminal
- the category of the mother node
- the category of the mother's head node
- the lemma of the mother's head node
- the category of the grandmother node
- the category of the grandmother's head node
- the lemma of the grandmother's head node
- the case features for noun phrases
- the category for PP objects
- the lemma for PP objects (if terminal node)

These features are also computed for the head of the phrase, determined using a set of head-finding rules in the style of Magerman (1995) adapted to TiGer. For lemmatisation, we use Tree-Tagger (Schmid, 1994) and case features of noun

phrases are obtained from a full German morphological analyser based on (Schiller, 1994). If a noun phrase consists of a single word (e. g. pronouns, but also bare common nouns and proper nouns), all case values output by the analyser are used to reflect the case syncretism. For multi-word noun phrases, the case feature is computed by taking the intersection of all case-bearing words inside the noun phrase, i. e. determiners, pronouns, adjectives, common nouns and proper nouns. If, for some reason (e.g., due to a bracketing error in phrase structure parsing), the intersection turns out to be empty, all four case values are assigned to the phrase.[5]

3.2 Constrained Optimisation

In the second step, a binary integer linear program is used to select those labels that optimise the whole tree labelling. A linear program consists of a linear objective function that is to be maximised (or minimised) and a set of constraints which impose conditions on the variables of the objective function (see (Clarke and Lapata, 2008) for a short but readable introduction). Although solving a linear program has polynomial complexity, requiring the variables to be integral or binary makes finding a solution exponentially hard in the worst case. Fortunately, there are efficient algorithms which are capable of handling a large number of variables and constraints in practical applications.[6]

For the function labeller, we define the set of binary variables $V = N \times L$ to be the crossproduct of the set of nodes N and the set of labels L. Setting a variable $x_{n,l}$ to 1 means that node n is labelled by label l. Every variable is weighted by the probability $w_{n,l} = P(l|f(n))$ which the classifier has assigned to this node-label combination. The objective function that we seek to optimise is defined as the sum over all weighted variables:

$$\max \sum_{n \in N} \sum_{l \in L} w_{n,l} x_{n,l} \qquad (4)$$

Since we want every node to receive exactly one label, we add a constraint that for every node n, exactly one of its variables is set to 1.

$$\sum_{l \in L} x_{n,l} = 1 \qquad (5)$$

Up to now, the whole system is doing exactly the same as an ordinary classifier that always takes the most probable label for each node. We will now add additional global and local linguistic constraints.[7]

The first and most important constraint restricts the number of each argument function (as opposed to modifier functions) to at most one per clause. Let $D \subset N \times N$ be the direct dominance relation between the nodes of the current tree. For every node n with category S (sentence) or VP (verb phrase), at most one of its daughters is allowed to be labelled SB (subject). The single-subject-function condition is defined as:

$$cat(n) \in \{S, VP\} \longrightarrow \sum_{\langle n,m \rangle \in D} x_{m,SB} \leq 1 \quad (6)$$

Identical constraints are added for labels OA, OA2, DA, OG, OP, PD, OC, EP.[8]

We add further constraints to capture the following linguistic restrictions:

- Of all daughters of a phrase, only one is allowed to be labelled HD (head).

$$\sum_{\langle n,m \rangle \in D} x_{m,HD} \leq 1 \qquad (7)$$

- If a noun phrase carries no case feature for nominative case, it cannot be labelled SB, PD or EP.

$$case(n) \neq nom \longrightarrow \sum_{l \in \{SB,PD,EP\}} x_{n,l} = 0$$
$$(8)$$

- If a noun phrase carries no case feature for accusative case, it cannot be labelled OA or OA2.

- If a noun phrase carries no case feature for dative case, it cannot be labelled DA.

- If a noun phrase carries no case feature for genitive case, it cannot be labelled OG or AG[9].

---

[5]We decided to train the classifier on automatically assigned and possibly ambiguous morphological information instead of on the hand-annotated and manually disambiguated morphological information provided by TiGer because we want the classifier to learn the German case syncretism. This way, the classifier will perform better when presented with unseen data (e.g. from parser output) for which no hand-annotated morphological information is available.

[6]See lpsolve (http://lpsolve.sourceforge.net/) or GLPK (http://www.gnu.org/software/glpk/glpk.html) for open-source implementations

[7]Note that some of these constraints are language specific in that they represent linguistic facts about German and do not necessarily hold for other languages. Furthermore, the constraints are treebank specific to a certain degree in that they use a TiGer-specific set of labels and are conditioned on TiGer-specific configurations and categories.

[8]SB = subject, OA = accusative object, OA2 = second accusative object, DA = dative, OG = genitive object, OP = prepositional object, PD = predicate, OC = clausal object, EP = expletive es

[9]AG = genitive adjunct

Unlike Klenner (2007), we do not use predefined subcategorization frames, instead letting the statistical model choose arguments.

In TiGer, sentences whose main verbs are formed from auxiliary-participle combinations, are annotated by embedding the participle under an extra VP node and non-subject arguments are sisters to the participle. Therefore we add an extension of the constraint in (6) to the constraint set in order to also include the daughters of an embedded VP node in such a case.

Because of the particulars of the annotation scheme of TiGer, we can decide some labels in advance. As mentioned before, punctuation does not get a label in TiGer. We set the label for those nodes to $--$ (no label). Other examples are:

- If a node's category is PTKVZ (separated verb particle), it is labeled SVP (separable verb particle).

$$cat(n) = PTKVZ \longrightarrow x_{n,SVP} = 1 \quad (9)$$

- If a node's category is APPR, APPRART, APPO or APZR (prepositions), it is labeled AC (adpositional case marker).
- All daughters of an MTA node (multi-token adjective) are labeled ADC (adjective component).

These constraints are conditioned on part-of-speech tags and require high POS-tagging accuracy (when dealing with raw text).

Due to the constraints imposed on the classification, the function labeller can no longer assign two subjects to the same S node. Faced with two nodes whose most probable label is SB, it has to decide on one of them taking the next best label for the other. This way, it outputs the optimal solution with respect to the set of constraints. Note that this requires the feature model not only to rank the correct label highest but also to provide a reasonable ranking of the other labels as well.

4 Evaluation

We conducted a number of experiments using 1,866 sentences of the TiGer Dependency Bank (Forst et al., 2004) as our test set. The TiGerDB is a part of the TiGer Treebank semi-automatically converted into a dependency representation. We use the manually labelled TiGer trees corresponding to the sentences in the TiGerDB for assessing the labelling quality in the intrinsic evaluation, and the dependencies from TiGerDB for assessing the quality and coverage of the automatically acquired LFG resources in the extrinsic evaluation.

In order to test on real parser output, the test set was parsed with the Berkeley Parser (Petrov et al., 2006) trained on 48k sentences of the TiGer corpus (Table 1), excluding the test set. Since the Berkeley Parser assumes projective structures, the training data and test data were made projective by raising non-projective nodes in the tree (Kübler, 2005).

precision	83.60	recall	82.81
f-score	83.20	tagging acc.	97.97

Table 1: evalb unlabelled parsing scores on test set for Berkeley Parser trained on 48,000 sentences (sentence length ≤ 40)

The maximum entropy classifier of the function labeller was trained on 46,473 sentences of the TiGer Treebank (excluding the test set) which yields about 1.2 million nodes as training samples. For training the Maximum Entropy Model, we used the BLMVM algorithm (Benson and More, 2001) with a width factor of 1.0 (Kazama and Tsujii, 2005) implemented in an open-source C++ library from Tsujii Laboratory.[10] The integer linear program was solved with the simplex algorithm in combination with a branch-and-bound method using the freely available GLPK.[11]

4.1 Intrinsic Evaluation

In the intrinsic evaluation, we measured the quality of the labelling itself. We used the node span evaluation method of (Blaheta and Charniak, 2000) which takes only those nodes into account which have been recognised correctly by the parser, i.e. if there are two nodes in the parse and the reference treebank tree which cover the same word span. Unlike Blaheta and Charniak (2000) however, we do not require the two nodes to carry the same syntactic category label.[12]

Table 2 shows the results of the node span evaluation. The labeller achieves close to 98% label accuracy on gold treebank trees which shows that the feature model captures the differences between the individual labels well. Results on parser output are about 4 percentage points (absolute) lower as parsing errors can distort local context features for the classifier even if the node itself has been parsed

---

[10]http://www-tsujii.is.s.u-tokyo.ac.jp/~tsuruoka/maxent/

[11]http://www.gnu.org/software/glpk/glpk.html

[12]We also excluded the root node, all punctuation marks and both nodes in unary branching sub-trees from evaluation.

correctly. The addition of the ILP constraints improves results only slightly since the constraints affect only (a small number of) argument labels while the evaluation considers all 40 labels occurring in the test set. Since the constraints restrict the selection of certain labels, a less probable label has to be picked by the labeller if the most probable is not available. If the classifier is ranking labels sensibly, the correct label should emerge. However, with an incorrect ranking, the ILP constraints might also introduce new errors.

	label accuracy	error red.
	without constraints	
gold	44689/45691 = 97.81%	–
parser	40578/43140 = 94.06%	–
	with constraints	
gold	44773/45691 = 97.99%*	8.21%
parser	40593/43140 = 94.10%	0.68%

Table 2: label accuracy and error reduction (all labels) for node span evaluation, * statistically significant, sign test, α = 0.01 (Koo and Collins, 2005)

As the main target of the constraint set are argument functions, we also tested the quality of argument labels. Table 3 shows the node span evaluation in terms of precision, recall and f-score for argument functions only, with clear statistically significant improvements.

	prec.	rec.	f-score
	without constraints		
gold standard	92.41	91.86	92.13
parser output	88.14	86.43	87.28
	with constraints		
gold standard	94.31	92.76	93.53*
parser output	89.51	86.73	88.09*

Table 3: node span results for the test set, argument functions only (SB, EP, PD, OA, OA2, DA, OG, OP, OC), * statistically significant, sign test, α = 0.01 (Koo and Collins, 2005)

For comparison and to establish a highly competitive baseline, we use the best-scoring system in (Chrupała and Van Genabith, 2006), trained and tested on exactly the same data sets. This purely statistical labeller achieves accuracy of 96.44% (gold) and 92.81% (parser) for all labels, and f-scores of 89.88% (gold) and 84.98% (parser) for argument labels. Tables 2 and 3 show that our system (with and even without ILP constraints) comprehensively outperforms all corresponding baseline scores.

The node span evaluation defines a correct labelling by taking only those nodes (in parser output) into account that have a corresponding node in the reference tree. However, as this restricts attention to correctly parsed nodes, the results are somewhat over-optimistic. Table 4 provides the results obtained from an evalb evaluation of the same data sets.[13] The gold standard scores are high confirming our previous findings about the performance of the function labeller. However, the results on parser output are much worse. The evaluation scores are now taking the parsing quality into account (Table 1). The considerable drop in quality between gold trees and parser output clearly shows that a good parse tree is an important prerequisite for reasonable function labelling. This is in accordance with previous findings by Punyakanok et al. (2008) who emphasise the importance of syntactic parsing for the closely related task of semantic role labelling.

	prec.	rec.	f-score
	without constraints		
gold standard	95.94	95.94	95.94
parser output	76.27	75.55	75.91
	with constraints		
gold standard	96.21	96.21	96.21
parser output	76.36	75.64	76.00

Table 4: evalb results for the test set

4.1.1 Subcategorisation Frames

Early on in the paper we mention that, unlike e. g. Klenner (2007), we did not include predefined subcategorisation frames into the constraint set, but rather let the joint statistical and ILP models decide on the correct type of arguments assigned to a verb. The assumption is that if one uses predefined subcategorisation frames which fix the number and type of arguments for a verb, one runs the risk of excluding correct labellings due to missing subcat frames, unless a very comprehensive and high quality subcat lexicon resource is available.

In order to test this assumption, we run an additional experiment with about 10,000 verb frames for 4,508 verbs, which were automatically extracted from our training section. Following Klenner (2007), for each verb and for each subcat frame for this verb attested at least once in the training data, we introduce a new binary variable f_n to the ILP model representing the n-th frame (for the verb) weighted by its frequency.

We add an ILP constraint requiring exactly one of the frames to be set to one (each verb has to have a subcat frame) and replace the ILP constraint in (6) by:

---

[13]Function labels were merged with the category symbols.

$$\sum_{\langle n,m\rangle \in D} x_{m,SB} - \sum_{SB \in f_i} f_i = 0 \qquad (10)$$

This constraint requires the number of subjects in a phrase to be equal to the number of selected[14] verb frames that require a subject. As each verb is constrained to "select" exactly one subcat frame (see additional ILP constraint above), there is at most one subject per phrase, if the frame in question requires a subject. If the selected frame does not require a subject, then the constraint blocks the assignment of subjects for the entire phrase. The same was done for the other argument functions and as before we included an extension of this constraint to cover embedded VPs. For unseen verbs (i.e. verbs not attested in the training set) we keep the original constraints as a back-off.

	prec.	rec.	f-score
all labels (cmp. Table 2)			
gold standard	97.24	97.24	97.24
parser output	93.43	93.43	93.43
argument functions only (cmp. Table 3)			
gold standard	91.36	90.12	90.74
parser output	86.64	84.38	85.49

Table 5: node span results for the test set using constraints with automatically extracted subcat frames

Table 5 shows the results of the test set node span evaluation when using the ILP system enhanced with subcat frames. Compared to Tables 2 and 3, the results are clearly inferior, and particularly so for argument grammatical functions. This seems to confirm our assumption that, given our data, letting the joint statistical and ILP model decide argument functions is superior to an approach that involves subcat frames. However, and importantly, our results do not rule out that a more comprehensive subcat frame resource may in fact result in improvements.

4.2 Extrinsic Evaluation

Over the last number of years, treebank-based deep grammar acquisition has emerged as an attractive alternative to hand-crafting resources within the HPSG, CCG and LFG paradigms (Miyao et al., 2003; Clark and Hockenmaier, 2002; Cahill et al., 2004). While most of the initial development work focussed on English, more recently efforts have branched to other languages. Below we concentrate on LFG.

---

[14]The variable representing this frame has been set to 1.

Lexical-Functional Grammar (Bresnan, 2001) is a constraint-based theory of grammar with minimally two levels of representation: c(onstituent)-structure and f(unctional)-structure. C-structure (CFG trees) captures language specific surface configurations such as word order and the hierarchical grouping of words into phrases, while f-structure represents more abstract (and somewhat more language independent) grammatical relations (essentially bilexical labelled dependencies with some morphological and semantic information, approximating to basic predicate-argument structures) in the form of attribute-value structures. F-structures are defined in terms of equations annotated to nodes in c-structure trees (grammar rules). Treebank-based LFG acquisition was originally developed for English (Cahill, 2004; Cahill et al., 2008) and is based on an f-structure annotation algorithm that annotates c-structure trees (from a treebank or parser output) with f-structure equations, which are read off of the tree and passed on to a constraint solver producing an f-structure for the given sentence. The English annotation algorithm (for Penn-II treebank-style trees) relies heavily on configurational and categorial information, translating this into grammatical functional information (subject, object etc.) represented at f-structure. LFG is "functional" in the mathematical sense, in that argument grammatical functions have to be single valued (there cannot be two or more subjects etc. in the same clause). In fact, if two or more values are assigned to a single argument grammatical function in a local tree, the LFG constraint solver will produce a clash (i. e. it will fail to produce an f-structure) and the sentence will be considered ungrammatical (in other words, the corresponding c-structure tree will be uninterpretable).

Rehbein (2009) and Rehbein and van Genabith (2009) develop an f-structure annotation algorithm for German based on the TiGer treebank resource. Unlike the English annotation algorithm and because of the language-particular properties of German (see Section 2), the German annotation algorithm cannot rely on c-structure configurational information, but instead heavily uses TiGer function labels in the treebank. Learning function labels is therefore crucial to the German LFG annotation algorithm, in particular when parsing raw text. Because of the strong case syncretism in German, traditional classification models using local

information only run the risk of predicting multiple occurences of the same function (subject, object etc.) at the same level, causing feature clashes in the constraint solver with no f-structure being produced. Rehbein (2009) and Rehbein and van Genabith (2009) identify this as a major problem resulting in a considerable loss in coverage of the German annotation algorithm compared to English, in particular for parsing raw text, where TiGer function labels have to be supplied by a machine-learning-based method and where the coverage of the LFG annotation algorithm drops to 93.62% with corresponding drops in recall and f-scores for the f-structure evaluations (Table 6).

Below we test whether the coverage problems caused by incorrect multiple assignments of grammatical functions can be addressed using the combination of classifier with ILP constraints developed in this paper. We report experiments where automatically parsed and labelled data are handed over to an LFG f-structure computation algorithm. The f-structures produced are converted into a dependency triple representation (Crouch et al., 2002) and evaluated against TiGerDB.

	cov.	prec.	rec.	f-score
upper bound	99.14	85.63	82.58	84.07
without constraints				
gold	95.82	84.71	76.68	80.49
parser	93.41	79.70	70.38	74.75
with constraints				
gold	99.30	84.62	82.15	83.37
parser	98.39	79.43	75.60	77.47
Rehbein 2009				
parser	93.62	79.20	68.86	73.67

Table 6: f-structure evaluation results for the test set against TigerDB

Table 6 shows the results of the f-structure evaluation against TiGerDB, with 84.07% f-score upper-bound results for the f-structure annotation algorithm on the original TiGer treebank trees with hand-annotated function labels. Using the function labeller without ILP constraints results in drastic drops in coverage (between 4.5% and 6.5% points absolute) and hence recall (6% and 12%) and f-score (3.5% and 9.5%) for both gold trees and parser output (compared to upper bounds). By contrast, with ILP constraints, the loss in coverage observed above almost completely disappears and recall and f-scores improve by between 4.4% and 5.5% (recall) and 3% (f-score) absolute (over without ILP constraints). For comparison, we repeated the experiment using the best-

scoring method of Rehbein (2009). Rehbein trains the Berkeley Parser to learn an extended category set, merging TiGer function labels with syntactic categories, where the parser outputs fully-labelled trees. The results show that this approach suffers from the same drop in coverage as the classifier without ILP constraints, with recall about 7% and f-score about 4% (absolute) lower than for the classifier with ILP constraints.

Table 7 shows the dramatic effect of the ILP constraints on the number of sentences in the test set that have multiple argument functions of the same type within the same clause. With ILP constraints, the problem disappears and therefore, less feature-clashes occur during f-structure computation.

	no constraints	constraints
gold	185	0
parser	212	0

Table 7: Number of sentences in the test set with doubly annotated argument functions

In order to assess whether ILP constraints help with coverage only or whether they affect the quality of the f-structures as well, we repeat the experiment in Table 6, however this time evaluating only on those sentences that receive an f-structure, ignoring the rest. Table 8 shows that the impact of ILP constraints on quality is much less dramatic than on coverage, with only very small variations in precison, recall and f-scores across the board, and small increases over Rehbein (2009).

	cov.	prec.	rec.	f-score
no constr.	93.41	79.70	77.89	78.79
constraints	98.39	79.43	77.85	78.64
Rehbein	93.62	79.20	76.43	77.79

Table 8: f-structure evaluation results for parser output excluding sentences without f-structures

Early work on automatic LFG acquisition and parsing for German is presented in Cahill et al. (2003) and Cahill (2004), adapting the English Annotation Algorithm to an earlier and smaller version of the TiGer treebank (without morphological information) and training a parser to learn merged Tiger function-category labels, and reporting 95.75% coverage and an f-score of 74.56% f-structure quality against 2,000 gold treebank trees automatically converted into f-structures. Rehbein (2009) uses the larger Release 2 of the treebank (with morphological information) reporting 77.79% f-score and coverage of 93.62% (Ta-

ble 8) against the dependencies in the TiGerDB test set. The only rule-based approach to German LFG-parsing we are aware of is the hand-crafted German grammar in the ParGram Project (Butt et al., 2002). Forst (2007) reports 83.01% dependency f-score evaluated against a set of 1,497 sentences of the TiGerDB. It is very difficult to compare results across the board, as individual papers use (i) different versions of the treebank, (ii) different (sections of) gold-standards to evaluate against (gold TiGer trees in TigerDB, the dependency representations provided by TigerDB, automatically generated gold-standards etc.) and (iii) different label/grammatical function sets. Furthermore, (iv) coverage differs drastically (with the hand-crafted LFG resources achieving about 80% full f-structures) and finally, (v) some of the grammars evaluated having been used in the generation of the gold standards, possibly introducing a bias towards these resources: the German hand-crafted LFG was used to produce TiGerDB (Forst et al., 2004). In order to put the results into some perspective, Table 9 shows an evaluation of our resources against a set of automatically generated gold standard f-structures produced by using the f-structure annotation algorithm on the original hand-labelled TiGer gold trees in the section corresponding to TiGerDB: without ILP constraints we achieve a dependency f-score of 84.35%, with ILP constraints 87.23% and 98.89% coverage.

	cov.	prec.	rec.	f-score
without constraints				
gold	95.24	97.76	90.93	94.22
parser	93.35	88.71	80.40	84.35
with constraints				
gold	99.30	97.66	97.33	97.50
parser	98.89	88.37	86.12	87.23

Table 9: f-structure evaluation results for the test set against automatically generated goldstandard (1,850 sentences)

5 Conclusion

In this paper, we addressed the problem of assigning grammatical functions to constituent structures. We have proposed an approach to grammatical function labelling that combines the flexibility of a statistical classifier with linguistic expert knowledge in the form of hard constraints implemented by an integer linear program. These constraints restrict the solution space of the classifier by blocking those solutions that cannot be correct. One of the strengths of an integer linear program

is the unlimited context it can take into account by optimising over the entire structure, providing an elegant way of supporting classifiers with explicit linguistic knowledge while at the same time keeping feature models small and comprehensible. Most of the constraints are direct formalizations of linguistic generalizations for German. Our approach should generalise to other languages for which linguistic expertise is available.

We evaluated our system on the TiGer corpus and the TiGerDB and gave results on gold standard trees and parser output. We also applied the German f-structure annotation algorithm to the automatically labelled data and evaluated the system by measuring the quality of the resulting f-structures. We found that by using the constraint set, the function labeller ensures the interpretability and thus the usefulness of the syntactic structure for a subsequently applied processing step. In our f-structure evaluation, that means, the f-structure computation algorithm is able to produce an f-structure for almost all sentences.

References

Steven J. Benson and Jorge J. More. 2001. A limited memory variable metric method in subspaces and bound constrained optimization problems. Technical report, Argonne National Laboratory.

Adam L. Berger, Vincent J.D. Pietra, and Stephen A.D. Pietra. 1996. A maximum entropy approach to natural language processing. *Computational linguistics*, 22(1):71.

Don Blaheta and Eugene Charniak. 2000. Assigning function tags to parsed text. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, pages 234 – 240, Seattle, Washington. Morgan Kaufmann Publishers Inc.

Thorsten Brants, Wojciech Skut, and Brigitte Krenn. 1997. Tagging grammatical functions. In *Proceedings of EMNLP*, volume 97, pages 64–74.

Sabine Brants, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George Smith. 2002. The TIGER treebank. In *Proceedings of the Workshop on Treebanks and Linguistic Theories*, page 2441.

Joan Bresnan. 2001. *Lexical-Functional Syntax.* Blackwell Publishers.

Miriam Butt, Helge Dyvik, Tracy Halloway King, Hiroshi Masuichi, and Christian Rohrer. 2002. The parallel grammar project. In *COLING-02 on Grammar engineering and evaluation-Volume 15*, volume pages, page 7. Association for Computational Linguistics.

Aoife Cahill, Martin Forst, Mairead McCarthy, Ruth ODonovan, Christian Rohrer, Josef van Genabith, and Andy Way. 2003. Treebank-based multilingual unification-grammar development. In *Proceedings of the Workshop on Ideas and Strategies for Multilingual Grammar Development at the 15th ESSLLI*, page 1724.

Aoife Cahill, Michael Burke, Ruth O'Donovan, Josef van Genabith, and Andy Way. 2004. Long-distance dependency resolution in automatically acquired wide-coverage PCFG-based LFG approximations. *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics - ACL '04*, pages 319–es.

Aoife Cahill, Michael Burke, Ruth O'Donovan, Stefan Riezler, Josef van Genabith, and Andy Way. 2008. Wide-Coverage Deep Statistical Parsing Using Automatic Dependency Structure Annotation. *Computational Linguistics*, 34(1):81–124, März.

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

Grzegorz Chrupała and Josef Van Genabith. 2006. Using machine-learning to assign function labels to parser output for Spanish. In *Proceedings of the COLING/ACL main conference poster session*, page 136143, Sydney. Association for Computational Linguistics.

Stephen Clark and Judith Hockenmaier. 2002. Evaluating a wide-coverage CCG parser. In *Proceedings of the LREC 2002*, pages 60–66.

James Clarke and Mirella Lapata. 2008. Global inference for sentence compression an integer linear programming approach. *Journal of Artificial Intelligence Research*, 31:399–429.

Richard Crouch, Ronald M. Kaplan, Tracy Halloway King, and Stefan Riezler. 2002. A comparison of evaluation metrics for a broad-coverage stochastic parser. In *Proceedings of LREC 2002 Workshop*, pages 67–74, Las Palmas, Canary Islands, Spain.

Peter Eisenberg. 2006. *Grundriss der deutschen Grammatik: Das Wort*. J.B. Metzler, Stuttgart, 3 edition.

Martin Forst, Núria Bertomeu, Berthold Crysmann, Frederik Fouvry, Silvia Hansen-Shirra, and Valia Kordoni. 2004. Towards a dependency-based gold standard for German parsers The TiGer Dependency Bank. In *Proceedings of the COLING Workshop on Linguistically Interpreted Corpora (LINC '04)*, Geneva, Switzerland.

Martin Forst. 2007. Filling Statistics with Linguistics Property Design for the Disambiguation of German LFG Parses. In *Proceedings of ACL 2007*. Association for Computational Linguistics.

Jun'Ichi Kazama and Jun'Ichi Tsujii. 2005. Maximum entropy models with inequality constraints: A case study on text categorization. *Machine Learning*, 60(1):159194.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of ACL 2003*, pages 423–430, Morristown, NJ, USA. Association for Computational Linguistics.

Manfred Klenner. 2005. Extracting Predicate Structures from Parse Trees. In *Proceedings of the RANLP 2005*.

Manfred Klenner. 2007. Shallow dependency labeling. In *Proceedings of the ACL 2007 Demo and Poster Sessions*, page 201204, Prague. Association for Computational Linguistics.

Terry Koo and Michael Collins. 2005. Hidden-variable models for discriminative reranking. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing - HLT '05*, pages 507–514, Morristown, NJ, USA. Association for Computational Linguistics.

Sandra Kübler. 2005. How Do Treebank Annotation Schemes Influence Parsing Results? Or How Not to Compare Apples And Oranges. In *Proceedings of RANLP 2005*, Borovets, Bulgaria.

David M. Magerman. 1995. Statistical decision-tree models for parsing. In *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*, page 276283, Morristown, NJ, USA. Association for Computational Linguistics Morristown, NJ, USA.

André F. T. Martins, Noah A. Smith, and Eric P. Xing. 2009. Concise integer linear programming formulations for dependency parsing. In *Proceedings of ACL 2009*.

Ryan McDonald and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proceedings of EACL*, volume 6.

Yusuke Miyao, Takashi Ninomiya, and Jun'ichi Tsujii. 2003. Probabilistic modeling of argument structures including non-local dependencies. In *Proceedings of the Conference on Recent Advances in Natural Language Processing RANLP 2003*, volume 2.

Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Gülsen Eryigit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. 2007. MaltParser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(2):95–135, Januar.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL - ACL '06*, pages 433–440, Morristown, NJ, USA. Association for Computational Linguistics.

Vasin Punyakanok, Wen-Tau Yih, Dan Roth, and Dav Zimak. 2004. Semantic role labeling via integer linear programming inference. In *Proceedings of the 20th international conference on Computational Linguistics - COLING '04*, Morristown, NJ, USA. Association for Computational Linguistics.

Vasin Punyakanok, Dan Roth, and Wen-tau Yih. 2008. The Importance of Syntactic Parsing and Inference in Semantic Role Labeling. *Computational Linguistics*, 34(2):257–287, Juni.

Adwait Ratnaparkhi. 1998. *Maximum Entropy Models for Natural Language Ambiguity Resolution*. Ph.D. thesis, University of Pennsylvania.

Ines Rehbein and Josef van Genabith. 2009. Automatic Acquisition of LFG Resources for German-As Good as it gets. In Miriam Butt and Tracy Holloway King, editors, *Proceedings of LFG Conference 2009*. CSLI Publications.

Ines Rehbein. 2009. *Treebank-based grammar acquisition for German*. Ph.D. thesis, Dublin City University.

Dan Roth and Wen-Tau Yih. 2004. A linear programming formulation for global inference in natural language tasks. In *Proceedings of CoNNL 2004*.

Anne Schiller, Simone Teufel, and Christine Stöckert. 1999. Guidelines für das Tagging deutscher Textcorpora mit STTS (Kleines und großes Tagset). Technical Report August, Universität Stuttgart.

Anne Schiller. 1994. Dmor - user's guide. Technical report, University of Stuttgart.

Helmut Schmid. 1994. Probabilistic Part-of-Speech Tagging Using Decision Trees. In *Proceedings of International Conference on New Methods in Language Processing*, volume 12. Manchester, UK.

Reut Tsarfaty and Khalil Sima'an. 2008. Relational-realizational parsing. In *Proceedings of the 22nd International Conference on Computational Linguistics - COLING '08*, pages 889–896, Morristown, NJ, USA. Association for Computational Linguistics.