Knowledge Sources for Constituent Parsing of German, a Morphologically Rich and Less-Configurational Language

Alexander Fraser*
Institute for NLP, University of Stuttgart

Helmut Schmid**
Institute for NLP, University of Stuttgart

Richárd Farkas†
Institute for NLP, University of Stuttgart

Renjing Wang‡
Institute for NLP, University of Stuttgart

Hinrich Schütze§
Institute for NLP, University of Stuttgart

We study constituent parsing of German, a morphologically rich and less-configurational language. We use a probabilistic context-free grammar treebank grammar that has been adapted to the morphologically rich properties of German by markovization and special features added to its productions. We evaluate the impact of adding lexical knowledge. Then we examine both monolingual and bilingual approaches to parse reranking. Our reranking parser is the new state of the art in constituency parsing of the TIGER Treebank. We perform an analysis, concluding with lessons learned, which apply to parsing other morphologically rich and less-configurational languages.

* Institute for Natural Language Processing, University of Stuttgart, Pfaffenwaldring 5b, 70569 Stuttgart, Germany. E-mail: fraser@ims.uni-stuttgart.de.
** Institute for Natural Language Processing, University of Stuttgart, Pfaffenwaldring 5b, 70569 Stuttgart, Germany. E-mail: schmid@ims.uni-stuttgart.de.
† Institute for Natural Language Processing, University of Stuttgart, Pfaffenwaldring 5b, 70569 Stuttgart, Germany. E-mail: farkas@ims.uni-stuttgart.de.
‡ Institute for Natural Language Processing, University of Stuttgart, Pfaffenwaldring 5b, 70569 Stuttgart, Germany.
§ Institute for Natural Language Processing, University of Stuttgart, Pfaffenwaldring 5b, 70569 Stuttgart, Germany.

Submission received: October 1, 2011; revised submission received: May 30, 2012; accepted for publication: August 3, 2012

1. Introduction

A large part of the methodology for parsing in natural language processing has been developed for English and a majority of publications on parsing are about parsing of English. English is a strongly configurational language. Nearly all of the syntactic information needed by any NLP application can be obtained by configurational analysis (e.g., by having a correct constituent parse).

Many other languages of the world are fundamentally different from English in this respect. At the other end of the configurational–nonconfigurational spectrum we find a language like Hungarian that has very little fixed structure on the level of the sentence. Leaving aside the issue of the internal structure of NPs, most sentence-level syntactic information in Hungarian is conveyed by morphology, not by configuration.

In this paper, we address German, a third type of language that is intermediate between English and Hungarian. German has strong configurational constraints (e.g., main clauses are verb-second) as well as rich derivational and inflectional morphology, all of which must be modeled for high-quality parsing. German's intermediate status raises a number of interesting issues in parsing that are of particular prominence for a mixed configurational/morphological language, but are—as we will argue—of general relevance for morphologically rich languages. Partly this is the case because there are few (if any) languages archetypical of being purely configurational and purely nonconfigurational (e.g., morphology is also important for English and even Hungarian has configurational constraints). For lack of a better term we refer to intermediate languages as typified by German as **MR&LC** for *morphologically rich and less-configurational*.

Part of the motivation for this special issue is that most work on parsing to date has been done on English, a morphologically simple language. As computational linguistics broadens its focus beyond English it becomes important to take a more general approach to parsing that can handle languages that are typologically very different from English. Rich morphology (RM) is one very salient characteristic of a language that affects the design of parsing methods. We argue that there are two other properties of languages that are relevant in a discussion of parsing RM languages: syncretism and configurationality. These two properties are correlated typologically with RM and should therefore be taken into account when we address parsing RM languages.[1]

We first define the three properties and explain their relevance for parsing. The large number of languages for which this correlation holds can be ordered along a single dimension that can be interpreted as degree of morphological complexity. We give examples for a number of languages that are positioned at different points on this scale. Finally, we argue that just as languages that are at the opposite end of the spectrum from English (prototypical examples of morphological richness like Hungarian) require parsing methods that can be quite different from those optimal for English, the same is true for a language like German that is in the middle of the spectrum—and what is required is in some respects different from what is optimal for one extreme (English) or the other (Hungarian).

The three correlated properties are rich morphology, syncretism, and configurationality. Morphological richness can be roughly measured by the number of different morphological forms a word of a particular syntactic category can have; for example,

1 We note, however, that this relationship is not a language universal. It is instead a frequently observed correlation; for Chinese, for instance, the correlation does not seem to hold as strongly.

a typical English noun has two forms (singular and plural), a typical German noun has eight forms (singular and plural in four different cases), and a typical Hungarian noun has several hundreds of forms. Syncretism refers to the fact that different morphological forms have identical surface realization; for example, the form *Mann* ('man' in German) can be the nominative, dative, or accusative singular of *Mann* depending on context. Configurationality refers to the degree to which the arrangement of words and phrases of a particular syntactic function in a sentence is fixed. English is highly configurational: it has limited flexibility in how the major phrases in a sentence (subject, verb, direct object, indirect object, etc.) can be ordered. Hungarian and Latin are highly flexible: Even though there are pragmatic constraints, in principle a large number of possible orderings are grammatical. German is less configurational. It has some strict constraints (verb second in main clauses, verb final in subordinate clauses), but also some properties of a nonconfigurational language; for example, ordering of phrases in the mittelfeld (the part of the main clause enclosed by the two parts of the verbal complex) is very flexible.

It is obvious why configurationality and rich morphology are typologically (negatively) correlated. Rich morphology specifies the syntactic role of a phrase in the sentence, so fixing a position is not required, and many morphologically rich languages therefore do not fix the position. Conversely, simple morphology gives little specific information about the role of words and phrases in the sentence. One device often used by morphologically simple languages to address this problem and reduce widespread ambiguity is to fix the order of words and phrases in the sentence.

Syncretism has an effect that is similar to simplification of complex morphology. Simple morphology is unspecific about grammatical function because it uses a small number of morphological categories. Syncretism is unspecific about grammatical function because it suffers from a high degree of ambiguity. Even though the number of different morphological categories is potentially large, syncretic forms conflate many of these categories, so that these forms are much less helpful in determining grammatical function than forms in a nonsyncretic language with the same number of categories. Again, to counteract the communicative difficulties that lack of morphological specificity would create, stricter constraints on ordering and configuration are often used by syncretic languages.

We have used English and Hungarian as examples for the extremes and German for the middle of the spectrum. We now give examples of other languages and their positions on the scale. Dutch is similar to German in that it also is verb second in main clauses and verb final in subordinate clauses. The order of arguments in the mittelfeld is much more restricted than in German, however. At the same time, Dutch morphology has been much more simplified in the last centuries than German morphology. This nicely confirms the correlation between RM and configurationality. Thus, Dutch is positioned between English and German on the scale.

Classical Arabic is somewhat similar to German: The number of different morphological forms is roughly comparable to German and it allows a number of different word orders. Modern Standard Arabic speakers rarely mark case, however, at least not in spontaneous speech. At the same time, Modern Standard Arabic speakers use SVO order much more frequently and consistently than is the case in Classical Arabic. Thus, Classical Arabic is roughly at the same position as German on the scale whereas spoken Modern Standard Arabic may be more comparable to Dutch.

Finally, Modern Greek is a language that is intermediate between German and Hungarian. It has richer morphology than German, but it has a fair amount of syncretism and therefore more morphological ambiguity than Hungarian. SVO is the

predominant word order in modern Greek, but other word orders can be used. The order within the noun phrase is more flexible than in German: Adjectives can precede or follow the noun.

In the examples we have given, the amount of information conveyed by a morphological form is negatively correlated with the amount of information conveyed by configuration. If morphology conveys a lot of information (due to a large number of distinctions and the lack of syncretism), then word order is freer and conveys less information. If morphology conveys less information (due to fewer distinctions or more syncretism), then configuration is fixed and provides more information to the speaker. This suggests that RM and configuration are important variables that should be taken into account in the design of parsing methods. In addition to looking at the extremes of the spectrum that are exemplified by English and Hungarian, we should also investigate the middle: morphologically (somewhat) rich languages that are less configurational. In this article, we look at the example of German.

One key question for MR&LC parsing is which type of parsing formalism to adopt, constituency or dependency. It is a widely held belief that dependency structures are better suited to represent syntactic analyses for morphologically rich languages because they allow non-projective structures (the equivalent of discontinuous constituents in constituency parsing). As Tsarfaty et al. (2010) point out, however, this is not the same as proving that dependency parsers function better than constituency parsers for parsing morphologically rich languages. In fact, most state-of-the-art dependency parsers (McDonald and Pereira 2006; Hall and Nivre 2008; Seeker et al. 2010a) generate purely projective dependency structures that are optionally transformed into non-projective structures in a post-processing step. Comparable post-processing techniques have been used in English constituency parsing (Gabbard, Marcus, and Kulick 2006; Schmid 2006; Cai, Chiang, and Goldberg 2011) to identify discontinuous constituents and might work for other languages, as well.

The overview paper of the Parsing German Shared Task (Kübler 2008) reports higher accuracies for detecting grammatical functions with dependency parsers than with constituent parsers, but the direct comparison is not fair as it required phrase boundaries to be correct on the constituent side while the tokens were the unit of evaluation on the dependency side.[2] How to carry out an absolutely fair comparison of the two representations is still an open research question.[3]

Constituent parses often provide more information than dependency parses. An example is the coordination ambiguity in *old men and women* versus *old men and children*. The correct constituent parse for the first expression contains a coordination at the noun level whereas the parse for the second expression coordinates at the level of NPs. The dependency structures of both expressions, on the other hand, are usually identical and thus unable to reflect the fact that *old* modifies *women* but not *children*. It is possible, in principle, to encode the difference in dependency trees (cf. Rambow 2010),

2 This is due to how the evalb tool used to calculate PARSEVAL works. If a constituent is not perfectly matched, the grammatical function is considered to be wrong, even if there was a partial match (at the token level). This is not a problem with dependency-based evaluation. For further discussion of the PARSEVAL metric and dependency-based evaluation see, for example, Rehbein and van Genabith (2007) and Tsarfaty, Nivre, and Andersson (2012).

3 Two possible solutions are to use TedEval (Tsarfaty, Nivre, and Andersson 2012), or to conduct an analysis of grammatical functions at the token level in a consistent fashion for both dependency and constituent parsers. In our case, the latter would require a high quality conversion from the Tiger constituency representation to a dependency representation, which we hope to implement in future work.

for example, by enriching the edge labels, but the constituent representation is simpler for this phenomenon.

Finally, there are some applications that need constituent parses rather than dependency parses. For instance, many hierarchical statistical machine translation systems use constituency parses, requiring the output of a dependency parser to be transformed into a constituent parse.[4] We conclude that there is no clear evidence for preferring dependency parsing over constituency parsing in analyzing languages with RM and instead argue that research in both frameworks is important.

We view the detailed description of a constituency parsing system for a morphologically rich language, a system that addresses the major problems that arise in constituency parsing for MR&LC, as one of our main contributions in this paper.

The first problem we address is the proliferation of phrase structure rules in MR&LC languages. For example, there are a large number of possible orderings of the phrases in the German mittelfeld, and many orderings are exceedingly rare. A standard constituency parser cannot estimate probabilities for the corresponding rules reliably.

The solution we adopt here is **markovization**—complex rules are decomposed into small unidirectional rules that can be modeled and estimated more reliably than complex rules. Although markovization in itself is not new, we stress its importance for MR&LC languages here and present a detailed, reproducible account of how we use it for German. Markovization combines the best of both worlds for MR&LC languages: Preferential configurational information can be formalized and exploited by the parser without incurring too large of a performance penalty due to sparse data problems.

The second problem that needs to be addressed in parsing many MR&LC languages is widespread syncretism. We mainly address syncretism by using a high performance finite-state automata-based morphological analyzer. Such an analyzer is of obvious importance for any morphologically rich language because the productivity of morphologically rich languages significantly increases the unknown-word rate in new text versus morphologically poor languages. So the parser cannot simply memorize the grammatical properties of words in the Treebank used for training. Instead we incorporate a complex guesser into our parser that, based on the input from the morphological analyzer, predicts the grammatical properties of new words and (equally important) unobserved grammatical properties of known words. With prevailing syncretism, this task is much more complex than in a language where case, gender, number, and so forth, can be deterministically deduced from morphology.

The morphological analyzer is based on (i) a finite state formalization of German morphology and (ii) a large lexicon of morphologically analyzed German words. We refer to these two components together as **lexical knowledge**. We show that lexical knowledge is beneficial for parsing performance for an MR&LC language like German.

In addition to lexical knowledge, there is a second important aspect of syncretism that needs to be addressed in MR&LC languages. Syntactic disambiguation in these languages must always involve both systems of grammatical encoding, morphology and configuration, acting together. The most natural way of doing this in a language like German is to perform this integration of the two knowledge sources directly as part of parsing. We do this by annotating constituent labels with grammatical function where appropriate. In contrast with syntactic parses of strongly configurational languages like English, syntactic parses of German are not useful for most tasks without having

4 We do note, however, that there are a few translation systems which use a dependency representation directly (e.g., Quirk, Menezes, and Cherry 2005; Shen, Xu, and Weischedel 2008; Tu et al. 2010).

grammatical functions indicated. It is not even possible to access the basic subcategorization of the verb (such as determining the subject) without grammatical functions. We argue that MR&LC languages like German should always be evaluated on labels-cum-grammatical-function.

Our last main contribution in this paper concerns the fact that we believe that MR&LC languages give rise to more ambiguity than languages that are predominantly configurational or morphological. As an example consider the German sentence "Die [the] Katze [cat] jagt [hunts] die [the] Schlange [snake]." In German either the cat or the snake can be the hunter. This type of ambiguity neither occurs in a strongly configurational language like English (where configuration determines grammatical function) nor in a morphologically rich language like Hungarian that has no or little syncretism (where morphology determines grammatical function). Although morphology and configuration in MR&LC languages often work hand in hand for complete disambiguation, there are also many sentences where neither of the two provides the necessary information for disambiguation. We believe that this distinguishing characteristic of MR&LC languages makes it necessary to tap additional knowledge sources. In this paper, we look at two such knowledge sources: monolingual reranking (which captures global properties of well-formed parses for additional disambiguation) and bilingual reranking (which exploits parallel text in a different language for disambiguation).

For monolingual reranking, we define a novel set of rich features based on subcategorization frames. We compare our compact feature set with a sparse feature set designed for German previously by Versley and Rehbein (2009). We show that the richer subcategorization-based framework for monolingual reranking is effective; it has comparable performance to the sparse feature set—moreover, they complement each other.

For bilingual reranking, we present our approach to bitext parsing, where a German parse is found that minimizes syntactic divergence with an automatically generated parse of its English translation. We pursue this approach for a number of reasons. First, one limiting factor for syntactic approaches to statistical machine translation is parse quality (Quirk and Corston-Oliver 2006). Improved parses of bitext should result in improved machine translation. Second, as more and more texts are available in several languages, it will be increasingly the case that a text to be parsed is itself part of a bitext. Third, we hope that the improved parses of bitext can serve as higher quality training data for improving monolingual parsing using a process similar to self-training (McClosky, Charniak, and Johnson 2006a).

We show that the three different knowledge sources we use in this paper (lexical knowledge, monolingual features, and bilingual features) are valuable separately. We also show that the gain of the two sets of reranking features (monolingual and bilingual) is additive, suggesting that they capture different types of information.

The resulting parser is currently the best constituent parser for German (with or without bilingual features). In particular, we show that the baseline parser without reranking is competitive with the previous state of the art (the Berkeley parser) and that the re-ranking can add an important gain.

2. Previous Work

Constituent parsing for English is well studied. The best generative constituent parsers are currently the Brown reranking parser (Charniak and Johnson 2005), the extension of this parser with self training by McClosky, Charniak, and Johnson (2006b), and the parser of Petrov and Klein (2007), which is an unlexicalized probabilistic

context-free grammar (PCFG) parser with latent feature annotations. Charniak and Johnson (2005) and Huang (2008) have introduced a significant improvement by feature-rich discriminative reranking as well.

The number of treebank constituent parsers for German is smaller. Dubey and Keller (2003) adapted Collins's (1997) lexicalized parser to German. An unlexicalized PCFG parser similar to our generative parser was presented by Schiehlen (2004). The best constituent parser participating in the ACL-08 Workshop on Parsing German (Kübler 2008) was the Berkeley parser (Petrov and Klein 2008). The Stanford parser was also adapted to German (Rafferty and Manning 2008). German dependency parsers have been developed by Menzel and Schröder (1998), Duchier and Debusmann (2001), Hall and Nivre (2008), Henderson et al. (2008), and Seeker et al. (2010a), to name a few.

There is also some previous work on German parse reranking. Forst (2007) presented a reranker for German LFG parsing, and Dreyer, Smith, and Smith (2006) applied reranking to German dependency parsing. Versley and Rehbein (2009) developed a reranking method for German constituent parsers. The work by Versley and Rehbein and by Schiehlen (2004) is closest to ours. Like them, we rerank the unlexicalized BitPar parser. We also refine treebank labels to increase parsing performance, but add more information and achieve a larger improvement. We use the monolingual feature set of Versley and Rehbein in our reranker, but add further monolingual features as well as bilingual features.

3. Generative Parsing Framework

Our generative parser is an unlexicalized PCFG parser which is based on the BitPar parser (Schmid 2004). BitPar uses a fast bitvector-based implementation of the well-known Cocke-Younger-Kasami algorithm and stores the chart as a large bit vector. This representation is memory efficient and allows full parsing (without search space pruning) with large treebank grammars. BitPar is also quite fast because the basic parsing operations are parallelized by means of (single-instruction) **and**-operations on bitvectors. BitPar can either be used to compute the most likely parse (Viterbi parse), or the full set of parses in the form of a parse forest, or the n-best parse trees.

3.1 Grammar

The grammar and lexicon used by our generative parser are extracted from the Tiger2 Treebank (Brants et al. 2002). Similar to Johnson (1998) and Klein and Manning (2003) we improve the accuracy of the unlexicalized parser by refining the non-terminal symbols of the grammar to encode relevant contextual information. This refinement weakens the strong independence assumptions of PCFGs and improves parsing accuracy. The extraction of the grammar and lexicon involves the following steps:

1. Discontinuous constituents are eliminated (Section 3.2).
2. Treebank annotations are transformed (Section 3.4) and augmented (Section 3.5).
3. Grammar rules, lexical rules, and their frequencies are extracted from the annotated parse trees.
4. The grammar is markovized (Section 3.6).

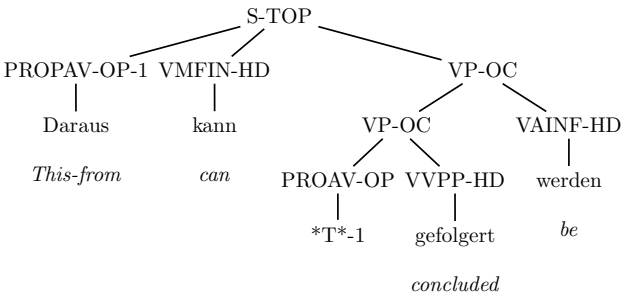

Figure 1
Projectivized parse tree for the sentence: *Daraus kann gefolgert werden* [From this can be concluded].

3.2 Raising for Non-Projectivity

The Tiger2 Treebank that we used in our experiments contains discontinuous constituents. As in other work on German parsing using the Tiger2 Treebank (Dubey and Keller 2003; Schiehlen 2004; Kübler, Hinrichs, and Maier 2006), we eliminated discontinuous constituents by raising those parts of the discontinuous constituent that do not contain the head to the child position of an ancestor node of the discontinuous constituent. Hsu (2010) compared three different Tiger2 conversion schemes and found raising to be the most effective. The projective parse tree in Figure 1, for instance, is obtained from a Tiger parse tree where the pronominal adverb *Daraus* was a discontinuous child of the lower VP-OC node.

The parse tree in Figure 1 shows a trace node and coreference indices (similar to the Penn Treebank annotation style for discontinuous constituents). If slash features are added to the nodes on the path between the PROAV node and its trace within the VP, it is possible to restore discontinuous constituents (Schmid 2006). Due to sparse data problems caused by the added slash features, however, the parsing accuracy drops by 1.5% compared with the version without slash features (when evaluated on projectivized parse trees). Traces are recognized with a precision of 53% and a recall of 33%. The correct antecedents are identified with a precision of 48% and a recall of 30%. These figures indicate that the identification of discontinuous constituents in Tiger parse trees is a harder task than in English Penn Treebank parses, considering the 84% F-score for the recognition of empty constituents and the 77% F-score for the identification of antecedents reported in Schmid (2006) for an analogous approach.

As the example in Figure 1 shows, the precise attachment point of constituents is often not required: We can simply assume that all constituents appearing at the S level are dependents of the main verb of the clause. Only for modifiers with scope ambiguities (e.g., negation particles) is it relevant whether they are attached at the S or VP level. These considerations suggest that it is better to recognize discontinuous constituents in a post-processing step as in Johnson (2001), Campbell (2004), and Levy and Manning (2004). In the rest of the paper, we will only work with parse trees from which coreference indices and trace nodes have been removed.

3.3 Morphological Features and Grammatical Functions

The Tiger2 Treebank annotates non-terminals not only with syntactic categories but also with grammatical function labels such as SB (subject), OA (accusative object), or

MO (modifier). These labels provide important information that is necessary in order to derive a semantic representation from a parse. It is not possible to infer the grammatical role of a constituent from its position in the parse tree alone (as can be done in English, for instance). Case information is needed in addition in order to help determine the correct grammatical role. The Tiger2 Treebank provides case, number, degree (positive, comparative, superlative), and gender information at the part-of-speech (POS) level.

Our parser concatenates the grammatical function labels as well as the case information of the POS tags to the base labels similarly to Dubey (2004) and Versley (2005). Our earlier experiments showed that adding case information increases F-score by 2.1% absolute. Further enriching the grammar with morphological features, however, hurts performance. Adding number features decreased F-score by 0.5%. Adding number, gender, and degree decreased F-score by 1.6%. When grammatical functions are taken into account in the evaluation, the performance drops by 1.5% when number, gender, and degree features are incorporated. It seems that the additional information supplied by the agreement features is not useful enough to outweigh sparse data problems caused by the more fine-grained label set. Therefore we only use case, but designing a smoothing procedure allowing us to use number, gender, and degree is interesting future work.

3.4 Tree Transformations

Similarly to Schiehlen (2004), we automatically augment the Tiger2 annotation with additional feature annotations. Our feature annotation set is larger than that of Schiehlen. In addition to making feature annotations, we also perform some tree transformations that reduce the complexity of the grammar. In all evaluations, we use the original (projectivized) Tiger parse trees as gold standard and convert the parse trees generated by our parser to the same format by undoing the transformations and removing the additional features. In the rest of this section, we explain the tree transformations that we used. The following section describes the feature annotations.[5]

Unary branching rules. The Tiger Treebank avoids unary branching nodes. NPs and other phrasal categories which dominate just a single node are usually omitted. The sentence *Sie zögern* [They hesitate], for instance, is analyzed as (S-TOP (PPER-SB Sie) (VVFIN-HD zögern)) without an explicit NP or VP. The lack of unary branching nodes increases the number of rules because now a rule S-TOP $\rightarrow$ PPER-SB VVFIN-HD is needed in addition to the rule S-TOP $\rightarrow$ NP-SB VVFIN-HD, for instance.

In order to reduce sparse-data problems, we insert unary branching nodes and transform this parse to (S-TOP (NP-SB (PPER-HD Sie)) (VVFIN-HD zögern)) by adding an NP node with the grammatical function (GF) of the pronoun. The GF of the pronoun, in turn, is replaced by HD (head). Such unary branching NPs are added on top of nouns (NN), pronouns (PPER, PDS, PIS, PRELS), cardinals (CARD), and complex proper names (PN) that are dominated by S, VP, TOP, or DL[6] nodes.[7] The transformation is reversible, which allows the original annotation to be restored.

5 Descriptions of the different symbols used in the Tiger annotation scheme are available at `http://www.ims.uni-stuttgart.de/tcl/RESOURCES/CL.html`.

6 DL is a discourse level constituent.

7 If a single proper name (NE) forms a noun phrase, we first add a PN node and then an NP node on top. If a simple noun (NN) with a GF other than NK appears inside of an NP, PP, CNP, CO, or AP, we also add an NP node on top of it. Similarly, we add a PN node on top of proper names (NE) in the same context.

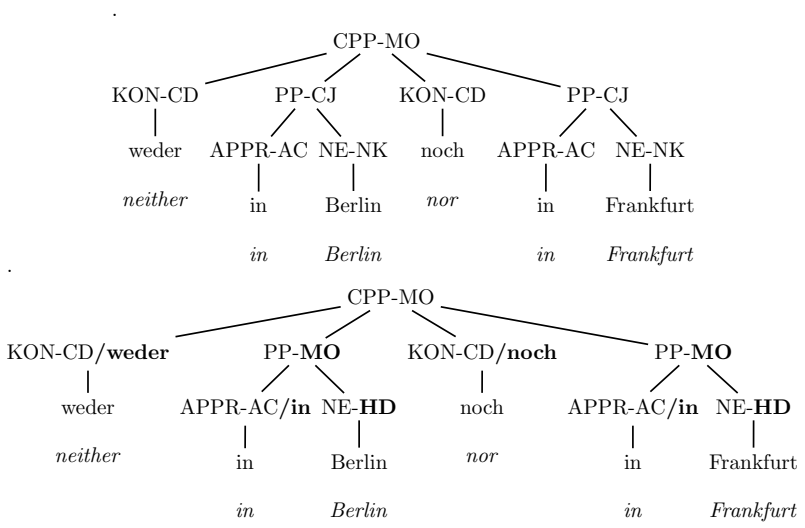

Figure 2
Parse of the phrase *weder in Berlin noch in Frankfurt* [neither in Berlin nor in Frankfurt] before and after selective lexicalization of prepositions and conjunctions. This example also shows the replacement of the grammatical function features CJ and NK discussed in the previous section. The modified parts are printed in **boldface**.

By adding a unary-branching NP-SB node, for instance, we introduce an additional independence assumption, namely, we assume that the expansion of the subject NP is independent of the other arguments and adjuncts of the verb (a plausible assumption that is confirmed by a performance improvement).

Elimination of NK. Tiger normally uses the grammatical function HD to mark the head of a phrase. In case of NPs and PPs, however, the GF of the head is NK (noun kernel). The same GF is also assigned to the adjectives and determiners of the noun phrase. We replace NK by HD in order to reduce the set of symbols.[8]

Elimination of CJ. Tiger annotates each conjunct in a coordination with the special grammatical function label CJ. We replace CJ by the grammatical function of the coordinated phrase. This transformation is also reversible.

3.5 Additional Feature Annotations

Selective lexicalization. We mark the POS tags of the frequent prepositions *in* [in], *von* [from, of], *auf* [on], *durch* [through, by means of], *unter* [under], *um* [around, at] and their variants regarding capitalization (e.g., *Unter*) and incorporation of articles (e.g., *unters*, *unterm*) with a feature which identifies the preposition. This can be seen as a restricted form of lexicalization. In the same way, we also "lexicalize" the coordinating conjunctions (KON-CD) *sowohl* [as well], *als* [as], *weder* [neither], *noch* [nor], *entweder* [either], and *oder* [or] if preceded by *entweder*. Figure 2 shows an example.

Sentence punctuation. If a clause node (S) has a sibling node labeled with POS tag "$." that dominates a question mark or exclamation mark, then the clause node and the POS tag are annotated with *quest* or *excl*, so the grammar models different clause types.

8 The original annotation can be restored: HD never occurs in NP or PP children in original Tiger parses.

.

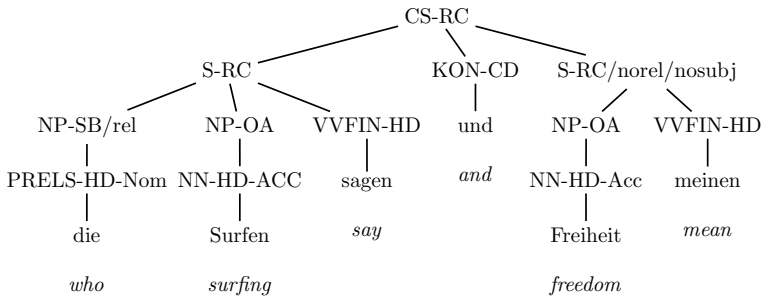

Figure 3
Parse of the phrase *die Surfen sagen und Freiheit meinen* [who say *surfing* and mean *freedom*] before and after annotation with relative clause features. This example also shows the *nosubj* feature, which will be discussed later.

Adjunct attachment. Adjuncts often differ with respect to their preferred attachment sites. Therefore, we annotate PPs and adverbials (AVP, ADV, ADJD) with one of the features N, V, or 0 which indicate a nominal parent (NP or PP), a verbal parent (VP, S), or anything else, respectively. In case of adverbial phrases (AVP), the label is propagated to the head child.

Relative clause features. In many relative clauses (S-RC), the relative pronoun (PRELS, PRELAT, PWAV, PWS) is embedded inside of another constituent. In this case, all nodes on the path between the pronoun and the clause node are marked with the feature *rel*. Furthermore, we add a feature *norel* to relative clauses if no relative pronoun is found. Figure 3 shows an example.

Wh features. Similar to the feature *rel* assigned to phrases that dominate a relative pronoun, we use a feature *wh* which is assigned to all NPs and PPs which immediately dominate a wh-pronoun (PWAT, PWS, PWAV). This feature better restricts the positions where such NPs and PPs can occur.

Noun sequence feature. If two nouns occur together within a German NP (as in *drei Liter Milch* [three liters (of) milk] or *Ende Januar* [end (of) January]), then the first noun is usually a kind of measure noun. We mark it with the feature *seq*.

Proper name chunks. Some noun phrases such as *Frankfurter Rundschau*, *Junge Union*, *Die Zeit* are used as proper names. In this case, the grammatical function of the NP is PNC. In order to restrict the nouns and adjectives that can occur inside of such proper name chunks, we mark their POS tags with the feature *name*.

Predicative APs. Complex adjectival phrases (AP) are either attributively used as noun modifiers inside of an NP or PP, or predicatively elsewhere. In order to better model the two types of APs, we mark APs that dominate a predicative adjective (ADJD) with the feature *pred*.[9]

Nominal heads of APs. Sometimes the head of an AP is a noun as in *(AP drei Millionen) Mark* [three million Marks] or *ein (AP politisch Verfolgter)* [a politically persecuted-person]. We mark these APs with the feature *nom*.

Year numbers. Years such as *1998* can appear in places where other numbers cannot. Therefore POS tags of numbers between 1900 and 2019 are marked with *year*.[10]

Clause type feature for conjunctions. The type of a subordinate clause and the subordinating conjunction are highly correlated. German object clauses (S-OC) usually

9 We also mark an AP parent of a node with the label AP-HD/pred in the same way.
10 For some texts, it might be advantageous to use a broader definition of year numbers.

start with *dass* [that] or *ob* [whether]; modifier clauses (S-MO) often start with *wenn* [if], *weil* [because], or *als* [when]. We mark subordinating conjunctions of argument clauses (S-OC), modifier clauses (S-MO), subject clauses (S-SB), and dislocated clauses (S-RE) with a feature (*OC*, *MO*, *SB*, or *RE*) identifying clause type. Without this feature, argument clauses of nouns, for instance, are often misanalyzed as modifiers of the main clause.

VP features. VPs that are headed by finite verbs, infinitives, past participles, imperatives, and *zu* infinitives are all used in different contexts. Therefore we mark object VPs (VP-OC) with a corresponding feature. When parsing the sentence *Alle Räume müssen mehrfach gesäubert und desinfiziert werden* [all rooms must multiply cleaned and disinfected be; all rooms must be ...], this feature allows the parser to correctly coordinate the two past participle VPs *mehrfach gesäubert* and *desinfiziert* instead of the past participle VP *mehrfach gesäubert* and the infinitival VP *desinfiziert werden*.

Phrases without a head. Some phrases in the Tiger corpus lack a head. This is frequent in coordinations. All phrases that do not have a child node with one of the grammatical functions HD, PNC, AC, AVC, NMC, PH, PD, ADC, UC, or DH are marked with the feature *nohead*.

Clauses without a subject. We also mark conjunct clauses with the feature *nosubj* if they are neither headed by an imperative nor contain a child node with the grammatical function SB (subject) or EP (expletive). This is useful in order to correctly parse coordinations where the subject is dropped in the second conjunct.

3.6 Markovization

The Tiger Treebank uses rather flat structures where nodes have up to 25 child nodes. This causes sparse data problems because only some of the possible rules of that length actually appear in the training corpus. The sparse data problem is solved by markovization (Collins 1997; Klein and Manning 2003), which splits long rules into a set of shorter rules. The shorter rules generate the child nodes of the original rule one by one. First, the left siblings of the head child of the rule are generated from left to right, then the right siblings are generated from right to left. Finally, the head is generated. Figure 4 shows the markovization of the rule NP → NM NN PP PP.

The auxiliary symbols that are used here encode information about the parent category, the head child, and previously generated children. Because all auxiliary symbols encode the head category, the head is already selected by the first rule, but only later actually generated by the last rule.

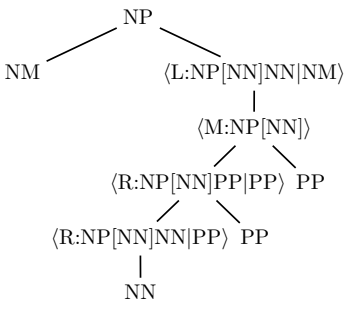

Figure 4
Markovization of the rule NP → NM NN PP PP.

The general form of the auxiliary symbols is ⟨*direction:parent[head]next|previous*⟩ where *direction* is either L, M, or R, *parent* is the symbol on the left hand side of the rule, *head* is the head on the right hand side of the rule, *next* is the symbol which will be generated next, and *previous* is the symbol that was generated before. Auxiliaries starting with L generate the children to the left of the head. Auxiliaries starting with R similarly generate the children to the right of the head and the head itself. The auxiliary starting with M is used to switch from generating left children to generating right children. Each rule contains information about the parent, the head, and (usually) three child symbols (which may include an imaginary boundary symbol). The first rule encodes the trigram *left-boundary NM NN*. The second rule is an exception which only encodes the bigram *NM NN*. The third rule encodes the trigram *PP PP right-boundary*. The last rule is an exception, again, and only encodes *NN PP*. There is no rule which covers the trigram consisting of the head and its two immediate neighbors.

Our markovization strategy only transforms rules that occur less than 10 times in the training data. If one of the auxiliary symbols introduced by the markovization (such as ⟨L:NP[NN]NN—NM⟩) is used less than 20 times (the values of the two thresholds were optimized on part of the development data) overall, it is replaced by a simpler symbol ⟨L:NP[NN]NN⟩ that encodes less context. In this way, we switch from a trigram model (where the next child depends on the two preceding children) to a bigram model (where it only depends on the preceding child) in order to avoid sparse data problems. The method is similar to the markovization strategy of Klein and Manning (2003) except that they markovize all rules. We simulated their strategy by raising the rule frequency threshold to a larger value, but obtained worse results. We also tried an alternative markovization strategy that generates all children left to right (the auxiliary symbols now lack the direction flag, and the rules cover all possible trigrams), but again obtained worse results. A disadvantage of our markovization method are spurious ambiguities. They arise because some of the rules which are not markovized are also covered by markovization rules.

3.7 Dealing with Unknown Words and Unseen POS Tags

BitPar includes a sophisticated POS guesser that uses several strategies to deal with unknown words and unseen POS tags of known words. Unknown words are divided into eight classes[11] based on regular expressions that are manually defined. These classes distinguish between lower-case words, capitalized words, all upper-case words, hyphenated words, numbers, and so forth. For each word class, BitPar builds a suffix tree (Weischedel et al. 1993; Schmid 1995; Brants 2000) from the suffixes of all words in the lexicon up to a length of 7. At each node of the suffix tree, it sums up the conditional POS probabilities (given the word) over all known words with that suffix. By summing POS probabilities rather than frequencies, all words have the same weight, which is appropriate here because we need to model the POS probabilities of infrequent words. BitPar computes POS probability estimates for each node using the sum of probabilities as a pseudo-frequency for each tag. The estimates are recursively smoothed with the Witten-Bell method using the smoothed POS probabilities of the parent node as a backoff probability distribution.[12] The suffix trees are pruned by recursively removing

11 We also experimented with more complex classifications, but they failed to improve the results.

12 The number of "observed" POS tags, which is needed by Witten-Bell smoothing, is defined as the number of POS tags with a pseudo-frequency larger than 0.5.

leaf nodes whose pseudo-frequency is below 5 or whose weighted information gain[13] is below a threshold of 1.

Whenever an unknown word is encountered during parsing, BitPar determines the word class and obtains the tag probability distribution from the corresponding suffix tree. BitPar assumes that function words are completely covered by the lexicon and never guesses function word POS classes for unknown words.

BitPar uses information from the unknown word POS guesser and (if available) information from an external lexicon (generated by a computational morphology, for instance, as we will discuss in Section 5.1) in order to predict unobserved POS tags for *known* words. First the external lexicon and the lexicon extracted from the training corpus are merged. Then smoothed probabilities are estimated using Witten-Bell smoothing with a backoff distribution. The backoff distribution is the average of:

(1) the probability distribution returned by the unknown word POS guesser if at least one possible POS tag of the word according to the lexicon is an open-class POS tag,

(2) the average POS probability distribution of all words with exactly the same set of possible POS tags as the given word[14] if at least one of the possible tags is unseen, and

(3) the prior POS probability distribution if no other word in the lexicon has the same set of possible POS tags and at least one of the word's possible POS tags is unseen.

4. Evaluation of the Generative Parser

As we present each knowledge source, we would like to evaluate it against manually annotated Treebanks. Our first evaluation shows that our generative parser introduced in the previous section is comparable with the Berkeley generative parser. Before we present this comparison in Section 4.1 we discuss evaluating parse accuracy.

In our evaluations, we use the Tiger Treebank (Brants et al. 2002) and a small Europarl Treebank (Padó and Lapata 2009). We take the first 40,474 sentences of the Tiger Treebank as training data (Tiger train), the next 5,000 sentences as development data (Tiger dev), and the last 5,000 sentences as test data (Tiger test). The Europarl data consists of 662 sentences[15] and are either completely used as test data and not divided up or we carried out seven-fold cross-validation experiments with our reranking models.

All parsers are evaluated on projectivized parse trees. This means that we apply step 1 of the grammar extraction process described in Section 3.1 to the test parses and use the result as the gold standard (except for the Padó set, which is already projectivized). The test sentences are parsed and the resulting parse trees are converted

13 The weighted information gain is the difference between the entropy of the parent node and the entropy of the current node, multiplied by the total frequency of the current node and divided by the number of "observed" POS tags of the current node.

14 A similar pooling of lexicon entries was previously used in the POS tagger of Cutting et al. (1992).

15 We use only the sentences in this set which had a single sentence as a translation, so that they could be used in bilingual reranking, which will be discussed later.

to the same format as the gold standard trees by undoing Steps 2, 3, and 4 of Section 3.1. This conversion involves four steps:

1. Demarkovization removes all the auxiliary nodes introduced by markovization and raises their children to the next non-auxiliary node.
2. The added unary-branching nodes are eliminated.
3. The original grammatical function labels NK inside of NPs and PPs, and CJ inside of coordinated phrases, are restored.
4. All feature annotations are deleted.

We use PARSEVAL scores (Black et al. 1991) and the standard evaluation tool *evalb*[16] to compare the converted parse trees with the gold standard parse trees using labeled F-score. We report accuracies for all test sentences and not just sentences of length up to 40. We do not evaluate parsers with gold standard POS tags, but instead automatically infer them. These considerations make our evaluation setting as close to the real-world setting as possible.

We report results for evaluations with and without grammatical functions. We report PARSEVAL scores with grammatical functions inside parentheses after the results using only basic constituent categories. We believe that grammatical functions are an important part of the syntactic analysis for any downstream applications in less-configurational languages such as German because crucial distinctions (e.g., the distinction between subject and object) are not feasible without them. We should mention that our results are not directly comparable to previously published results on the Tiger2 corpus (Kübler 2008; Versley and Rehbein 2009; Seeker et al. 2010b), because each of the previous studies used different portions of the corpus and there are differences in the evaluation metric as well. The transformed corpus (in our train, development, and test split format) and the evaluation scripts we used are available,[17] which we hope will enable direct comparison with our results.

4.1 Comparison of BitPar and Berkeley

The best constituent parser participating in the Parsing German Shared Task (Kübler 2008) was the Berkeley parser (Petrov and Klein 2008) and to the best of our knowledge it has achieved the best published accuracy for German constituency parsing so far. The Berkeley parser takes an automated approach, in which each constituent symbol is split into subsymbols applying an expectation-maximization method. We compare our manually enriched grammar to this automatic approach.

We trained the Berkeley parser on Tiger train using the basic constituent categories concatenated to the grammatical function labels as starting symbols. We found that it achieved the best PARSEVAL scores on Tiger dev after the fourth iteration. This model was used for parsing Tiger dev, Tiger test, and the Europarl corpus.

BitPar achieved 82.51 (72.46), 76.67 (65.61), and 77.13 (66.06), and the Berkeley parser achieved 82.76 (73.20), 76.37 (65.66), and 75.51 (63.3) on the three corpora, respectively. In general, these results indicate that these two parsers are competitive. On the other hand, the fact that the results of the Berkeley parser are much worse than

16 `http://nlp.cs.nyu.edu/evalb/`, 2008 version.
17 See `http://www.ims.uni-stuttgart.de/tcl/RESOURCES/CL.html`.

BitPar on the out-of-domain Europarl corpus indicates that it overfits to the domain of the training corpus (Tiger2). Following a reviewer suggestion, we looked at the sentences containing many words not occurring in the training data, and observed that our lexical resource is strongly helpful for these sentences. Another disadvantage of the automatic approach of the Berkeley parser is that the resulting subsymbols are not easily interpretable, which can hinder defining features for parse reranking using them. Based on these considerations, we decided to use BitPar in our reranking experiments. The combination of the two radically different approaches (linguistically motivated grammar extensions and automatic symbol splitting) is a rather promising area of research for improving parsing accuracy, which we plan to address in future work.

5. Impact of Our Lexical Resource

5.1 Integration of SMOR with BitPar

There are a large number of inflected word forms for many German lemmas. This causes sparse data problems if some forms are not observed in the training data. BitPar applies the heuristics described in Section 3.7 to obtain POS probabilities for unseen words. Although these heuristics seem to work quite well, we expect better results if the parser has access to information from a morphological analyzer.

We use the German finite-state morphology SMOR (Schmid, Fitschen, and Heid 2004) to provide sets of possible POS tags for all words. SMOR covers inflection, derivation, and compounding and achieves good coverage in combination with the stem lexicon IMSLex (Lezius, Dipper, and Fitschen 2000). SMOR is integrated into the parser in the following way. We create a combined word list from the training and testing data[18] and analyze it with SMOR. The SMOR analyses are then mapped to the POS tag set used by the parser, and supplied to BitPar as an external lexicon (see Section 3.7).

Consider the example word *erlischt* [goes out], which did not appear in the training corpus. SMOR produces the analysis *erlöschen.V.3.Sg.Pres.Ind*, which is mapped to *VVFIN-HD* and added to the lexicon. Using this entry, BitPar correctly parsed the sentence *Die Anzeige erlischt* [The display goes out]. Without using SMOR, the parser analysed *erlischt* as a past participle because *scht* is a frequent past participle ending.

5.2 Effect on In-Domain and Out-of-Domain Parsing

In order to measure the effect of the integration of a German morphology on parsing accuracy (see Section 5.1), we tested the BitPar parser on the Tiger data and on Europarl data. The results are summarized in Table 1. They show that the morphology helps on out-of-domain data (Europarl), but not so much on in-domain data (Tiger). The POS tagging accuracy, however, also increases on Tiger data by 0.13%. When grammatical functions are included in the evaluation, the performance improvement more than doubles on Europarl data. As a result, we decided to use the finite-state morphology in the rest of the experiments we conducted.

Table 1 also shows that the Tiger test data is harder to parse than the dev data. We examined the two subcorpora and found that the test data contains longer sentences

18 Because we are only using the words here, and not their POS labels, this approach is methodologically sound and could be applied to any unparsed data in the same way.

Table 1
Effect of using finite-state morphology on parsing accuracy. The values in parentheses are labeled F-scores from the evaluation with grammatical functions.

morphology	Tiger dev	Tiger test	Europarl
without	82.51 (72.46)	76.67 (65.61)	76.81 (65.31)
with	82.42 (72.36)	76.84 (65.91)	77.13 (66.06)
difference	–0.09 (–0.10)	+0.17 (+0.30)	+0.32 (+0.75)

(18.4 vs. 15.3 words on average) and that the ratio of unknown words is higher (10.0% vs. 7.6%).

6. Parse Reranking

The most successful supervised phrase-structure parsers are feature-rich discriminative parsers that heavily depend on an underlying PCFG grammar (Charniak and Johnson 2005; Huang 2008). These approaches consist of two stages. At the first stage they apply a PCFG grammar to extract possible parses. The full set of possible parses cannot be iterated through in practice, and is usually pruned as a consequence. The *n-best list parser*s keep just the 50–100 best parses according to the PCFG. Other methods remove nodes and edges from the packed parse forest whose posterior probability is under a pre-defined threshold (Charniak and Johnson 2005).

The task of the second stage is to select the best parse from the set of possible parses (i.e., rerank this set). These methods use a large feature set (usually a few million features) (Collins 2000; Charniak and Johnson 2005). The *n*-best list approaches can straightforwardly use local and non-local features as well because they decide at the sentence-level (Charniak and Johnson 2005). Involving non-local features is more complicated in the forest-based approaches. The conditional random field methods usually use only local features (Yusuke and Jun'ichi 2002; Finkel, Kleeman, and Manning 2008). Huang (2008) introduced a beam-search and average perceptron-based procedure incorporating non-local features in a forest-based approach. His empirical results show only a minor improvement from incorporating non-local features, however.

In this study, we experiment with *n*-best list reranking using a maximum entropy machine learning model for (re)ranking along with local and non-local features. Our reranking framework follows Charniak and Johnson (2005). At the first-stage of parsing, we extract the 100 best parses for a sentence according to BitPar's probability model. At parsing time, a weight vector w is given for the feature vectors (which numerically represent one possible parse) and we select the parse with the highest inner product of these two vectors. The goal of training is to adjust w. In the maximum entropy framework, this is achieved by solving the optimization problem of maximizing the posterior probability of the **oracle parse**—the parse with the highest F-score.[19] Our method aims to select the oracle, as the gold standard parse is often not present in the 100-best parses.[20] Our preliminary experiments showed that parse candidates close

19 Ties are broken using the PCFG probabilities of the parses.
20 The oracle F-score (i.e., the upper limit of 100-best reranking on the Tiger development corpus) is 90.17.

to the oracle confuse training. Hence during training, we removed all parses whose F-score is closer than 1.0 to the score of the oracle.[21]

As we discussed in Section 1, the parsing output of morphologically rich languages is useful only when it is additionally annotated with grammatical functions. The oracle parses often change if the grammatical function labels are also taken into consideration at the PARSEVAL score calculation. Hence slightly different objective functions are used in the two cases. We will report results achieved by reranking models where the oracle selection for training agrees with the evaluation metric utilized—that is, we trained different models (which differ in the oracle selection) for the basic constituent label evaluation and for the evaluation on grammatical functions.

During training we followed an eight-fold cross validation technique for candidate extraction (Collins 2000). Here, one-eighth of the training corpus was parsed with a PCFG extracted from seven-eighths of the data set. This provides realistic training examples for the reranker as these parses were not seen during grammar extraction. We used the ranking MaxEnt implementation of MALLET (McCallum 2002) with default parameters.

7. Monolingual Reranking

7.1 Subcategorization-Based Monolingual Reranking Features

We introduce here several novel subcategorization-based features for monolingual reranking. For this, we first describe our algorithm for extracting subcategorization (subcat) information. We use our enriched version of the Tiger2 training set. In order to extract verbal subcat frames we find all nodes labeled with the category **S** (clause) or **VP-MO** (modifying VP) and extract their arguments. Arguments[22] are nodes of the categories shown in Table 2. The arguments of nouns are obtained by looking for **NN** nodes which are either dominated by an **NP** or a **PP**, and which take a following node of category **PP**, **VP-OC**, or **S-OC** as argument.

The feature functions we present are mostly lexicalized. This means we need access to the head words of the arguments. The argument heads are extracted as follows: As NP head we take the last node whose function label is either **HD**, **NK**, or **PH**. If this node is of category **NP** or **PN**, we recursively select the head of that constituent. Similarly, the head of an **AP** is the last node with functional label **HD**. If it is an **AP**, the head is searched inside of it. In the case of **PP**s, we extract two heads, namely, the preposition (or postposition) as well as the nominal head of the **PP**, which is found using similar rules as for **NP**s. We also extract the case of the nominal head.

The extraction of verbal heads is somewhat more complicated. In order to obtain the correct verbal head of a clause irrespective of the verb position (verb-first, verb-second, verb-final), we extract all verbs that are dominated by the clause and a possibly empty sequence of **VP-OC** or **VP-PD** (statal passive) nodes and an optional **VZ-HD** node. Then we take the first non-finite verb, or alternatively the first finite verb if all verbs were finite. In order to avoid sparse data problems caused by the many different inflections of German verbs, we lemmatize the verbs.

21 In Fraser, Wang, and Schütze (2009) we used Minimum Error Rate Training. Once we made this change to maximum entropy the results on small feature sets became similar (details omitted).

22 An exception to this is that if a **PP** argument dominates a node of category **PROAV-PH**, it is considered a **PROAV-PH** argument. An example is the sentence *Er [he] wartet [waits] (PP-OP (PROAV-PH darauf [for this]), (S-RE dass [that] sie [she] kommt [comes]))*.

Table 2
Arguments used in extracted subcategorization frames.

NP-SB, PN-SB, CNP-SB, S-SB, VP-SB	subjects
NP-OA, PN-OA, CNP-OA	direct objects
NP-DA, PN-DA, CNP-DA	indirect objects
PRF-OA	reflexive direct objects
PRF-DA	reflexive indirect objects
NP-PD, CNP-PD	predicative NPs
ADJD-PD, AP-PD, CAP-PD	predicative adjectives
S-OC, CS-OC	argument clauses
PP-OP, CPP-OP	PP arguments
VP-OC/zu	infinitival complement clauses
PROAV-OP	pronominal adverbs serving as PP proxies such as daraus [out of this]
NP-EP	expletive subjects
VP-RE, NP-RE	VP/NP appearing in expletive constructions

In the case of coordinated phrases, we take the head of the first conjunct. Arguments are sorted to put them in a well-defined order. An example is that given the correct parse of the sentence *Statt [instead of] Details [details] zu [to] nennen [name], hat [has] er [he] unverdrossen [assiduously] die [the] "Erfolgsformel" [formula of success] wiederholt [repeated]*, meaning "instead of naming the details, he assiduously repeated the formula of success," we extract the two subcat frames:
VP-MO OBJ:Details VZ-HD:zu:nennen
S-TOP VP-MO SUBJ:er OBJ:Erfolgsformel VVPP-HD:wiederholt

We can now describe our features. The features focus on subcat frames taken from S nodes (VP-MO is treated as S), and on attachment of prepositions and conjunctions to nouns. We define conditional probability and mutual information (MI) features.

The two conditional probability features are **ProbPrepAttach** and **ProbAdverbAttach**, which calculate the probability for each preposition or adverb to be attached to its governor, given the label of the governor. We estimate this from the training data as follows, for the example of the PP feature. In the feature scoring, we give each preposition attachment a score which is the negative log10 of the probability $p(\text{lex\_prep}|\text{label\_governor}) = f(\text{lex\_prep}, \text{label\_governor})/f(\text{label\_governor})$ (with a cutoff of 5).

For all of our other monolingual features, we use (negative) pointwise mutual information: $-log_{10}(p(a,b)/p(a)p(b))$ (here we use cutoffs of 5 and -5).

MI_NounP and **MI_NounConj** give an assessment of a preposition or a conjunction being attached to a noun (given the lexicalized preposition and the lexicalized noun).

For the **MI_VSubcat** feature, we use as a the frame (without lexicalization), and as b the head verb. $p(a)$ is estimated as the relative frequency of this frame over all frames extracted from Tiger2 train. **MI_VSimpleSubcat** is a simpler version of **MI_VSubcat**. PP is excluded from frames because PP is often an adjunct rather than an argument.

For the **MI_VArg** feature, we use as a the argument function and the head word of the argument (e.g., OBJ:Buch, which is "*book*" used as an object). As b we again use the head verb. The estimate of $p(a)$ is frequency(OBJ:Buch)/(total number of extracted frames).[23] In addition, this feature is refined into individual features for

23 We make the assumption that every frame has an object, but that this object can be NULL.

different kinds of arguments: **MI_VSubj**, **MI_VObj**, **MI_VIobj**, **MI_VPP**, **MI_VPRF**, **MI_VS_OC**, **MI_VVP**, and **MI_VerbPROAV**. As an example, the MI of "*lesen, OBJ:Buch*" (reading, object:Book) would be used for the MI_VArg features and for the MI_VObj feature. For functions such as **MI_VPP** which are headed by both a function word (here, a preposition) and a content word, only the function word is used (and no case).

The last MI feature is **MI_VParticle**. Some German verbs contain a separable particle, which can also be analyzed as an adverb but will then have a different meaning. For the sentence "*Und [and] Frau [Mrs.] Künast [(proper name)] bringt [brings] das [that] auch [also] nicht [not] rüber [across],*" if "*rüber*" is analyzed as an adverb, the verb means to carry/take/bring over [to another physical location], but if it is viewed as a particle, the sentence means Frau Künast is not able to explain this. The feature **MI_VParticle** helps with this kind of disambiguation.

7.2 The Versley and Rehbein Feature Set

We also carried out experiments with the feature set of Versley and Rehbein (2009), which is specially designed for German. It consists of features constructed from the lexicalized parse tree along with features based on external statistical information.

The features here are local in the sense that their values can be computed at the constituent in question, its daughters, and its spanning words. All features except the external statistical information are binary and indicate that a lexicalized pattern is present in the parse. They were originally designed for forest-based reranking (Versley and Rehbein 2009). Following Charniak and Johnson (2005) we sum up these local feature values in the parse tree. Thus our versions count the number of times that a particular pattern occurs in the entire parse tree.

The patterns used can be further subcategorized into three groups. The **wordform-based patterns** are token–POS (e.g., one pattern is "*lesen-VVINF*") and the word class of the token in question (word class comes from an automatic clustering of words based on contextual features). The **constituent-based patterns** are the size of the constituent, the constituent label, and the right-hand side of the derivational rule applied at the node in question. The last and biggest group of the pattern features is formed by the **bilexical dependencies**. They are based on the head word of the constituent node in question and its daughters. Versley and Rehbein (2009) have also introduced features that exploit **statistical information gathered from an external data set** and aim to resolve PP attachment ambiguity. Mutual information values were gathered on the association between nouns and immediately following prepositions, as well as between prepositions and closely following verbs on the DE-WaC corpus (Baroni and Kilgarriff 2006). These feature values were then used at NP→PP and VP→PP daughter attachments.

A total of 2.7 million features fired in the Tiger train. We ignored features firing in less than five sentences for computational efficiency, resulting in 117,000 extremely sparse features.

7.3 Monolingual Reranking Experiments

We rerank 100-best lists from BitPar (Schmid 2004), which uses the grammar extraction procedure and lexical resources introduced in Section 3. In each of the experiments we extracted the grammar from the Tiger train and used it to obtain the 100-best parses for the sentences of the evaluation corpus.

We trained reranking models on the Tiger train as described in Section 6 using our subcategorization-based features, the Versley09 feature set, and the union of these two

Table 3
The PARSEVAL score of **monolingual** features to rerank the parses of Europarl (seven-way cross-validation on 662 sentences) and Tiger2 (development and test sets).

	Tiger dev	Tiger test	Europarl CROSS	Europarl IN
Baseline	82.42 (72.36)	76.84 (65.91)	77.13 (66.06)	
subcat	83.19 (73.63)	77.65 (67.21)	77.23 (66.13)	77.73 (66.95)
Versley09	83.56 (73.89)	78.57 (68.42)	77.82 (66.87)	77.62 (66.05)
subcat+Versley09	84.19 (74.96)	78.86 (69.04)	77.76 (66.84)	77.93 (66.75)

sets. We evaluated the models on Tiger dev, Tiger test, and Europarl. As the domains of Tiger and Europarl are quite different, besides this cross-domain parser evaluation (CROSS) we carried out an in-domain (IN) evaluation as well. In the latter we followed the seven-fold cross-validation approach, that is, the reranking models were trained on six-sevenths of Europarl. The results are presented in Table 3.

The results presented in Table 3 show that the reranking models achieve an improvement over the baseline parser using both our and the Versley09 feature sets. The Versley09 feature set achieved better results than our monolingual features when a training dataset with sufficient size is given (Tiger). On the other hand using our 16 rich features (compared with 117,000 sparse features) is more suitable for the settings where only a limited amount of training instances are available (the training sets consist of 567 sentences of Europarl in seven-fold cross-validation). The reranking models using the union of the feature sets obtain close to the sum of the improvements of the two individual feature sets. The subcategorization features model rich non-local information, and the fine-grained features capture local distinctions well and the features based on the Web corpus access additional knowledge.

We performed an experiment adding one feature at a time, and found that the most effective features were **ProbAdverbAttach**, **MI_VPP**, **MI_VPRF**, **MI_VSubj**, and **MI_VArg**. After this the variation caused by numeric instability was too high to see a consistent incremental gain from the rest of the features. We conclude that these features can be robustly estimated and have more discriminative power than the others, but we emphasize that we used all features in our experiments.

Figure 5 shows a parse tree produced by the BitPar parser in which the noun phrase *diese Finanzierung* is incorrectly classified as an accusative object. The monolingual subcategorization features **MI_VSubcat**, **MI_VSimpleSubcat**, and **MI_VArg** enable the reranker to correctly analyze the noun phrase as a subject and to move it from the VP level to the S level.

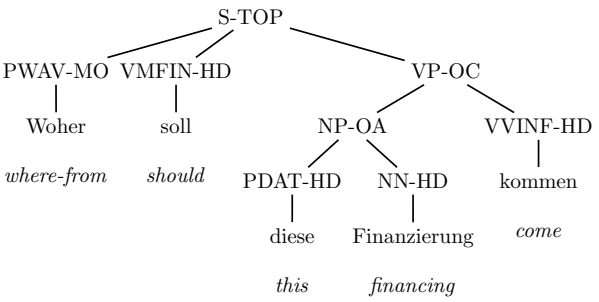

Figure 5
Erroneous parse produced by BitPar that is corrected by monolingual features.

8. Bilingual Reranking

We now present our bilingual reranking framework. This follows our previous work (Fraser, Wang, and Schütze 2009), which defined feature functions for reranking English parses, but now we will use these same feature functions (and three additional feature functions introduced to capture phenomena higher in the syntactic tree) to rerank German parses. The intuition for using this type of bitext projection feature is that ambiguous structures in one language often correspond to unambiguous structures in another. Our feature functions are functions on the hypothesized English parse e, the German parse g, and the word alignment a, and they assign a score (varying between 0 and infinity) that measures *syntactic divergence*. The alignment of a sentence pair is a function that, for each English word, returns a set of German words with which the English word is aligned. Feature function values are calculated either by taking the negative log of a probability, or by using a heuristic function which scales similarly.[24]

The bilingual feature functions we define are functions that measure different types of syntactic divergence between an English parse and a German parse. Charniak and Johnson (2005) defined the state of the art in discriminative n-best constituency parsing of English syntax (without the use of self-training). The n-best output of their generative parser is reranked discriminatively by a reranker. We call this CJRERANK. We will use an array of feature functions measuring the syntactic divergence of candidate German parses with the projection of the English parse obtained from CJRERANK.

In our experiments we use the English text of the parallel Treebank extracted from the Europarl corpus and annotated by Padó and Lapata (2009). There are 662 German sentences that are aligned to single English sentences; this is the set that we use. Due to the limited number of trees, we perform cross-validation to measure performance.

The basic idea behind our feature functions is that any constituent in a sentence should play approximately the same syntactic role and have a similar span as the corresponding constituent in a translation. If there is an obvious disagreement, it is probably caused by wrong attachment or other syntactic mistakes in parsing. Sometimes in translation the syntactic role of a given semantic constituent changes; we assume that our model penalizes all hypothesized parses equally in this case.

To determine which features to describe here we conducted a greedy feature addition experiment (adding one feature at a time), on top of our best monolingual system (combining both *subcat* and *Versley09* feature sets). All bilingual experiments use all of the features (not just the features we describe here). Definitions are available.[25]

BitParLogProb (the only monolingual feature used in the bilingual-only experiment) is the negative log probability assigned by BitPar to the German parse.

8.1 Count Feature Functions

Count feature functions *count* projection constraint violations.

Feature **CrdBin** counts binary events involving the heads of coordinated phrases. If we have a coordination where the English CC is aligned only with a German KON, and

24 A probability of 1 is a feature value of 0, whereas a low probability is a feature value which is $\gg 0$.
25 See `http://www.ims.uni-stuttgart.de/tcl/RESOURCES/CL.html`.

Table 4
Other projection features selected; see the previously mentioned Web page[25] for precise definitions.

POSParentPrjWordPerG2E	Computes the span difference between all the parent constituents of POS tags in a German parse and their respective coverage in the corresponding English parse, measured using percentage coverage of the sentence in words. The feature value is the sum of all the differences. The projection direction is from German to English.
AbovePOSPrjPer	Projection direction is from English to German, and measured in percentage sentence coverage using characters, not words. The feature value is calculated over all constituents above the POS level in the English tree.
AbovePOSPrjWord	Calculates a length-based difference using words.
POSPar2Prj	Only applies when the POS tag's parent has two children (the POS tag has only one sibling). Projects from English to German and calculates a length-based difference in characters.
POSPar2PrjPer	Calculates a percentage-based difference based on characters.
POSPar2PrjG2E	Like **POSPar2Prj** except projects from German to English.
POSPar2PrjWordG2E	Like **POSPar2PrjG2E** except uses word-based differences.

both have two siblings, then the value contributed to **CrdBin** is 1 (indicating a constraint violation) unless the head of the English left conjunct is aligned with the head of the German left conjunct and likewise the right conjuncts are aligned.

Feature **Q** simply captures a mismatch between questions and statements. If a German sentence is parsed as a question but the parallel English sentence is not, or vice versa, the feature value is 1; otherwise the value is 0.

Feature **S-OC** considers that a clausal object (OC) in a German parse should be projected to a simple declarative clause in English. This feature counts violations.

EngPPinSVP checks whether a PP inside of a S or VP in English attaches to the same (projected) constituent in German. If an English PP follows immediately a VP or a single verb, and the whole constituent is labeled "S" or "VP," then the PP should be identified as governed by the VP. In this case the corresponding German PP should attach as well to the German VP to which the English VP is projected (attachment in German can be to the left or to the right). If the governor in German does not turn out to be a VP or have a tag starting with "V," a value of 1 will be added to the feature for this German parse.

EngLeftSVP checks whether the left sibling of S or VP in English attaches to the same (projected) constituent in German (where attachment can be left or right). This feature counts violations.

Span Projection Feature Functions. Span projection features calculate an absolute or percentage difference between a constituent's span and the span of its projection. Span size is measured in characters or words. To project a constituent in a parse, we use the word alignment to project all word positions covered by the constituent and then look for the smallest covering constituent in the parse of the parallel sentence.

PPParentPrjWord checks the correctness of PP attachment. It projects all the parents of PP constituents in an English parse to German, and sums all the span differences. It is measured in words. In addition to **PPParentPrjWord** we implement two bonus features, **NonPPWord** and **NonPPPer**. The former simply calculates the number of words that

do not belong to PP phrases in the sentence, and the latter computes the non-PP proportion in a character-based fashion. These can be thought of as tunable parameters which adjust **PPParentPrjWord** to not disfavor large PPs. The other selected projection features are described in Table 4.

Probabilistic Feature Functions. We use Europarl (Koehn 2005), from which we extract a parallel corpus of approximately 1.22 million sentence pairs, to estimate the probabilistic feature functions described in this section.

We describe the feature **PTag**, despite the fact that it was not selected by the feature analysis, because several variations (described next) were selected. **PTag** measures tagging inconsistency based on estimating the probability for each English word that it has a particular POS tag, given the aligned German word's POS tag. To avoid noisy feature values due to outliers and parse errors, we bound the value of **PTag** at 5.[26] We use relative frequency to estimate this feature. When an English word is aligned with two words, estimation is more complex. We heuristically give each English and German pair one count. The value calculated by the feature function is the geometric mean[27] of the pairwise probabilities.

The feature **PTagEParent** measures tagging inconsistency based on estimating the probability that the parent of the English word at position i has a particular tag, given the aligned German word's POS label. **PTagBiGLeft** measures tagging inconsistency based on estimating the probability for each English word that it has a particular POS tag, given the aligned German word's label and the word to the left of the aligned German word's label. **PTagBiGParent** measures tagging inconsistency based on estimating the probability for each English word that it has a particular POS tag, given the aligned German word's label and the German word's parent's label.

8.2 Bilingual Reranking Experiments

We performed experiments looking at bilingual reranking performance. To train the parameters of the probabilistic feature functions, we use 1-best parses of the large Europarl parallel corpus (from CJRERANK and BitPar). We work on the same 100-best list (of the German sentences in the small Padó set) as was used in the previous section. We parse the English sentences of the small Europarl set with CJRERANK; this parse is used as our bilingual knowledge source. Finally we rerank using the bilingual features (results in the first row of Table 5).

We then combine the monolingual features with the bilingual features. We rerank using both the monolingual and the bilingual features together, and the results are presented in Table 5. The bilingual feature-based reranker achieved 1 percentage point improvement over the baseline. This advantage was just slightly decreased when monolingual features are also present. This indicates again that the monolingual and bilingual features can capture different linguistic phenomena and their information content is rather different. As in the Europarl IN setting, using the large sparse Versley09 feature set the reranker could not learn a meaningful model from a moderate-sized training data set.

26 Throughout this paper, assume $\log(0) = -\infty$.

27 Each English word has the same weight regardless of whether it was aligned with one or with more German words.

Table 5
PARSEVAL scores of **bi+monolingual** features to rerank the parses of Europarl (seven-way cross-validation) and the added value of bilingual features over the results achieved by the corresponding monolingual feature set.

Mono features	without bilingual	with bilingual	added value
NONE	77.13 (66.06)	78.10 (67.12)	+0.97 (+1.06)
subcat	77.73 (66.95)	78.54 (67.95)	+0.78 (+1.00)
Versley09	77.62 (66.05)	77.71 (66.06)	+0.09 (+0.01)
subcat+Versley09	77.93 (66.75)	78.70 (67.45)	+0.78 (+0.70)

The parse tree in Figure 6 demonstrates the value of bilingual features. It was produced by the monolingual reranker and it incorrectly combines the two adverbs *aber* and *ebenso* into an adverbial phrase and places this under the VP. The bilingual reranker instead attaches the two adverbs separately at the S level. The attachment to the S node indicates that the two adverbs modify the modal verb *kann* and not the full verb *sagen*. This is triggered by the feature **POSPar2Prj**.

8.3 Previous Work on Bitext Parsing

Bitext parsing was also addressed by Burkett and Klein (2008). In that work, they use feature functions defined on triples of (English parse tree, Chinese parse tree, alignment) which are combined in a log-linear model, much as we do. In later work (Burkett, Blitzer, and Klein 2010), they developed a unified joint model for solving the same problem using a weakly synchronized grammar. To train these models they use a small parallel Treebank that contains gold standard trees for parallel sentences in Chinese and English, whereas we only require gold standard trees for the language we are reranking. Another important difference is that Burkett and Klein (2008) use a large number of automatically generated features (defined in terms of feature generation templates) whereas we use a small number of carefully designed features that we found by linguistic analysis of parallel corpora. Burkett, Blitzer, and Klein (2010) use a subset of the features of Burkett and Klein (2008) for synchronization, along with monolingual parsing and alignment based features. Finally, *self-training* (McClosky, Charniak, and Johnson 2006b) is another differentiator of our work. We use probabilities estimated from aligned English CJRERANK parses and German BitPar parses of the large Europarl corpus in our bilingual feature functions. These feature functions are used to

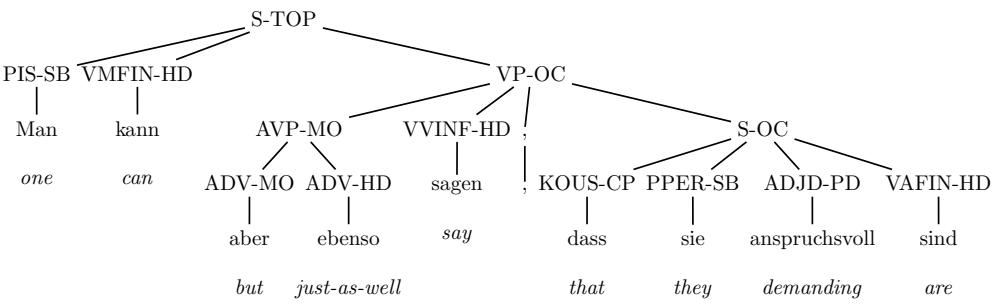

Figure 6
Erroneous parse produced by the reranker using only monolingual features, which is corrected by bilingual features. The sentence means *One can, however, just as well say that they are demanding*.

improve ranking of German BitPar parses in the held-out test sets, which is a form of self-training.

Two other interesting studies in this area are those of Fossum and Knight (2008) and of Huang, Jiang, and Liu (2009). They improve English prepositional phrase attachment using features from a Chinese sentence. Unlike our approach, however, they do not require a Chinese syntactic parse as the word order in Chinese is sufficient to unambiguously determine the correct attachment point of the prepositional phrase in the English sentence without using a Chinese syntactic parse.

We know of no other work that has investigated to what extent monolingual and bilingual features in parse reranking are complementary. In particular, the work on bitext parsing by Burkett and Klein (2008) does not address the question as to whether the effect of monolingual and bilingual features in parse reranking is (partially) additive.

We demonstrate bilingual improvement for a strong parser of German. Previously, we showed bilingual improvement for parsing English with an unlexicalized parser (Fraser, Wang, and Schütze 2009), using 34 of the 37 bilingual feature functions we use in this work.

9. Conclusion

In this paper, we have focused on MR&LC languages like German—languages that are morphologically rich, but also have a strong configurational component. We have argued that constituency parsing is, perhaps contrary to conventional wisdom, an appropriate parsing formalism for MR&LC because constituents capture configurational constraints in a transparent way and because for many applications constituency parsing is preferable to dependency parsing. Our detailed description of a constituency parsing system for a morphologically rich language, a system that addresses the major problems that arise in constituency parsing for MR&LC, is one main contribution of this paper. Two of these problems are rule proliferation and syncretism. We have addressed rule proliferation by markovization and syncretism by (i) deploying a high performance finite-state-based morphological analyzer that is based on rich lexical knowledge and (ii) encoding grammatical functions directly as part of the phrase labels. This direct encoding allows us to directly combine morphological and configurational information in parsing and arrive at a maximally disambiguated parse. We argued that this is the right setup for MR&LC languages because applications must have access to grammatical functions.

A large part of this paper was concerned with making available and evaluating additional knowledge sources for improved parsing of the MR&LC language German. Our motivation was that (as we argued) MR&LC languages have in general higher ambiguity than purely configurational and purely morphological languages, in particular with respect to grammatical functions. Apart from the lexical knowledge embedded in the morphological analyzer, we presented work on two other knowledge sources to address this type of additional ambiguity: monolingual reranking (which looks at global sentence-wide constraints for disambiguation) and bitext reranking (which exploits parallel text for disambiguation). We were able to improve the performance of a strong baseline parser using these three knowledge sources and we showed that they are largely complementary: Performance improvements were additive when we used them together. The resulting parser is currently the best constituent parser for German (with or without bilingual features).

New languages and even new domains can require new treebanks. To create such a treebank for a MR&LC language, we would first annotate a small number of gold

standard trees, using parallel text with English or another language if such text is available. Next, we would consider how to quickly differentiate constituents of the same type using constituent labels plus grammatical functions, as we outlined in Section 3. Following this, we would use BitPar to build a parser in the same way as we presented here, and to determine the optimal level of markovization, which we assume would be very high with a small number of gold standard training trees. Next, as more trees are annotated in an active learning framework, we would begin to develop morphological analysis. We would implement the bilingual framework following this (if we have access to bitext). Then we would implement basic subcategorization extraction and add monolingual features. Finally, as more gold standard trees are annotated, the reranking framework should be constantly retrained. In particular, we expect that the effect of the knowledge sources we have presented will be much stronger when starting with less training data.

Our work in this paper will be of use to developers of German syntactic parsers as we have state-of-the-art performance using linguistically motivated features that are easy to understand. We also hope that our work can serve as a cookbook of ideas to try for others working on parsers for other morphologically rich languages.

Acknowledgments

We would like to thank Sandra Kübler and Yannick Versley. We gratefully acknowledge Deutsche Forschungsgemeinschaft (DFG) for funding this work (grants SCHU 2246/6-1 *Morphosyntax for MT* and SFB 732 D4 *Modular lexicalization of PCFGs*). This work was supported in part by the IST Programme of the European Community, under the PASCAL2 Network of Excellence, IST-2007-216886. This publication only reflects the authors' views.

References

Baroni, Marco and Adam Kilgarriff. 2006. Large linguistically processed Web corpora for multiple languages. In *EACL: Posters & Demonstrations*, pages 87–90, Trento.

Black, E., S. Abney, S. Flickenger, C. Gdaniec, C. Grishman, P. Harrison, D. Hindle, R. Ingria, F. Jelinek, J. Klavans, M. Liberman, M. Marcus, S. Roukos, B. Santorini, and T. Strzalkowski. 1991. Procedure for quantitatively comparing the syntactic coverage of English grammars. In *Proceedings of the Workshop on Speech and Natural Language*, HLT '91, pages 306–311, Pacific Grove, CA.

Brants, Sabine, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George Smith. 2002. The TIGER Treebank. In *Proceedings of the Workshop on Treebanks and Linguistic Theories*, pages 24–41, Sozopol.

Brants, Thorsten. 2000. TnT—a statistical part-of-speech tagger. In *ANLP*, pages 224–231, Seattle, WA.

Burkett, David, John Blitzer, and Dan Klein. 2010. Joint parsing and alignment with weakly synchronized grammars. In *HLT-NAACL*, pages 127–135, Los Angeles, CA.

Burkett, David and Dan Klein. 2008. Two languages are better than one (for syntactic parsing). In *EMNLP*, pages 877–886, Honolulu, HI.

Cai, Shu, David Chiang, and Yoav Goldberg. 2011. Language-independent parsing with empty elements. In *ACL*, pages 212–216, Portland, OR.

Campbell, Richard. 2004. Using linguistic principles to recover empty categories. In *ACL*, pages 645–652, Barcelona.

Charniak, Eugene and Mark Johnson. 2005. Coarse-to-fine *n*-best parsing and MaxEnt discriminative reranking. In *ACL*, pages 173–180, Ann Arbor, MI.

Collins, Michael. 1997. Three generative, lexicalized models for statistical parsing. In *ACL*, pages 16–23, Madrid.

Collins, Michael. 2000. Discriminative reranking for natural language parsing. In *ICML*, pages 25–70, Stanford, CA.

Cutting, Doug, Julian Kupiec, Jan Pedersen, and Penelope Sibun. 1992. A practical part-of-speech tagger. In *ANLP*, pages 133–140, Trento.

Dreyer, Markus, David A. Smith, and Noah A. Smith. 2006. Vine parsing and minimum risk reranking for speed and precision. In *CoNLL*, pages 201–205, New York, NY.

Dubey, Amit. 2004. *Statistical Parsing for German: Modeling Syntactic Properties and Annotation Differences*. Ph.D. thesis, Saarland University.
Dubey, Amit and Frank Keller. 2003. Probabilistic parsing for German using sister-head dependencies. In *ACL*, pages 96–103, Sapporo.
Duchier, Denys and Ralph Debusmann. 2001. Topological dependency trees: a constraint-based account of linear precedence. In *ACL*, pages 180–187, Toulouse.
Finkel, Jenny Rose, Alex Kleeman, and Christopher D. Manning. 2008. Efficient, feature-based, conditional random field parsing. In *ACL*, pages 959–967, Columbus, OH.
Forst, Martin. 2007. Filling statistics with linguistics—property design for the disambiguation of German LFG parses. In *Proceedings of the ACL Workshop on Deep Linguistic Processing*, pages 17–24, Prague.
Fossum, Victoria and Kevin Knight. 2008. Using bilingual Chinese–English word alignments to resolve PP-attachment ambiguity in English. In *AMTA*, pages 48–53, Honolulu, HI.
Fraser, Alexander, Renjing Wang, and Hinrich Schütze. 2009. Rich bitext projection features for parse reranking. In *EACL*, pages 282–290, Athens.
Gabbard, Ryan, Mitchell Marcus, and Seth Kulick. 2006. Fully parsing the Penn Treebank. In *HLT-NAACL*, pages 184–191, Morristown, NJ.
Hall, Johan and Joakim Nivre. 2008. A dependency-driven parser for German dependency and constituency representations. In *Proceedings of the Workshop on Parsing German*, pages 47–54, Columbus, OH.
Henderson, James, Paola Merlo, Gabriele Musillo, and Ivan Titov. 2008. A latent variable model of synchronous parsing for syntactic and semantic dependencies. In *CoNLL*, pages 178–182, Manchester.
Hsu, Yu-Yin. 2010. Comparing conversions of discontinuity in PCFG parsing. In *TLT*, pages 103–113, Tartu.
Huang, Liang. 2008. Forest reranking: Discriminative parsing with non-local features. In *ACL*, pages 586–594, Columbus, OH.
Huang, Liang, Wenbin Jiang, and Qun Liu. 2009. Bilingually constrained (monolingual) shift-reduce parsing. In *EMNLP*, pages 1,222–1,231, Singapore.
Johnson, Mark. 1998. PCFG models of linguistic tree representations. *Computational Linguistics*, 24(4):613–632.
Johnson, Mark. 2001. A simple pattern-matching algorithm for recovering empty nodes and their antecedents. In *ACL*, pages 136–143, Philadelphia, PA.
Klein, Dan and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *ACL*, pages 423–430, Sapporo.
Koehn, Philipp. 2005. Europarl: a parallel corpus for statistical machine translation. In *MT Summit X*, pages 79–86, Phuket.
Kübler, Sandra. 2008. The PaGe 2008 shared task on parsing German. In *Proceedings of the Workshop on Parsing German*, pages 55–63, Columbus, OH.
Kübler, Sandra, Erhard W. Hinrichs, and Wolfgang Maier. 2006. Is it really that difficult to parse German? In *EMNLP*, pages 111–119, Sydney.
Levy, Roger and Christopher D. Manning. 2004. Deep dependencies from context-free statistical parsers: Correcting the surface dependency approximation. In *ACL*, pages 327–334, Barcelona.
Lezius, Wolfgang, Stefanie Dipper, and Arne Fitschen. 2000. IMSLex—representing morphological and syntactical information in a relational database. In *EURALEX*, pages 133–139, Stuttgart.
McCallum, Andrew Kachites. 2002. Mallet: A machine learning for language toolkit. `http://mallet.cs.umass.edu`.
McClosky, David, Eugene Charniak, and Mark Johnson. 2006a. Effective self-training for parsing. In *HLT-NAACL*, pages 152–159, Morristown, NJ.
McClosky, David, Eugene Charniak, and Mark Johnson. 2006b. Reranking and self-training for parser adaptation. In *COLING-ACL*, pages 337–344, Sydney.
McDonald, Ryan and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *EACL*, pages 81–88, Trento.
Menzel, Wolfgang and Ingo Schröder. 1998. Decision procedures for dependency parsing using graded constraints. In *COLING-ACL Workshop on Processing of Dependency-Based Grammars*, pages 78–87, Montreal.
Padó, Sebastian and Mirella Lapata. 2009. Cross-lingual annotation projection of semantic roles. *Journal of Artificial Intelligence Research*, 36:307–340.

Petrov, Slav and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *HLT-NAACL*, pages 404–411, Rochester, NY.

Petrov, Slav and Dan Klein. 2008. Parsing German with latent variable grammars. In *Proceedings of the Workshop on Parsing German*, pages 33–39, Columbus, OH.

Quirk, Chris and Simon Corston-Oliver. 2006. The impact of parse quality on syntactically-informed statistical machine translation. In *EMNLP*, pages 62–69, Sydney.

Quirk, Chris, Arul Menezes, and Colin Cherry. 2005. Dependency treelet translation: Syntactically informed phrasal SMT. In *ACL*, pages 271–279, Oxford.

Rafferty, Anna and Christopher D. Manning. 2008. Parsing three German Treebanks: Lexicalized and unlexicalized baselines. In *Proceedings of the Workshop on Parsing German*, pages 40–46, Columbus, OH.

Rambow, Owen. 2010. The simple truth about dependency and phrase structure representations: an opinion piece. In *HLT-NAACL*, pages 337–340, Los Angeles, CA.

Rehbein, Ines and Josef van Genabith. 2007. Evaluating evaluation measures. In *NODALIDA*, pages 372–379, Tartu.

Schiehlen, Michael. 2004. Annotation strategies for probabilistic parsing in German. In *COLING*, pages 390–396, Geneva.

Schmid, Helmut. 1995. Improvements in part-of-speech tagging with an application to German. In *Proceedings of the ACL SIGDAT-Workshop*, pages 47–50, Dublin.

Schmid, Helmut. 2004. Efficient parsing of highly ambiguous context-free grammars with bit vectors. In *COLING*, pages 162–168, Geneva.

Schmid, Helmut. 2006. Trace prediction and recovery with unlexicalized PCFGs and slash features. In *COLING-ACL*, pages 177–184, Sydney.

Schmid, Helmut, Arne Fitschen, and Ulrich Heid. 2004. SMOR: A German computational morphology covering derivation, composition and inflection. In *LREC*, pages 1,263–1,266, Lisbon.

Seeker, Wolfgang, Bernd Bohnet, Lilja Øvrelid, and Jonas Kuhn. 2010a. Informed ways of improving data-driven dependency parsing for German. In *COLING: Posters*, pages 1,122–1,130, Beijing.

Seeker, Wolfgang, Ines Rehbein, Jonas Kuhn, and Josef Van Genabith. 2010b. Hard constraints for grammatical function labelling. In *ACL*, pages 1,087–1,097, Uppsala.

Shen, Libin, Jinxi Xu, and Ralph Weischedel. 2008. A new string-to-dependency machine translation algorithm with a target dependency language model. In *ACL-HLT*, pages 577–585, Columbus, OH.

Tsarfaty, Reut, Joakim Nivre, and Evelina Andersson. 2012. Cross-framework evaluation for statistical parsing. In *EACL*, pages 44–54, Avignon.

Tsarfaty, Reut, Djamé Seddah, Yoav Goldberg, Sandra Kuebler, Yannick Versley, Marie Candito, Jennifer Foster, Ines Rehbein, and Lamia Tounsi. 2010. Statistical parsing of morphologically rich languages (SPMRL) what, how and whither. In *Proceedings of the NAACL HLT 2010 First Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 1–12, Los Angeles, CA.

Tu, Zhaopeng, Yang Liu, Young-Sook Hwang, Qun Liu, and Shouxun Lin. 2010. Dependency forest for statistical machine translation. In *COLING*, pages 1,092–1,100, Beijing.

Versley, Yannick. 2005. Parser evaluation across text types. In *Fourth Workshop on Treebanks and Linguistic Theories (TLT)*, pages 209–220, Barcelona.

Versley, Yannick and Ines Rehbein. 2009. Scalable discriminative parsing for German. In *IWPT*, pages 134–137, Paris.

Weischedel, Ralph, Marie Meteer, Richard Schwartz, Lance Ramshaw, and Jeff Palmucci. 1993. Coping with ambiguity and unknown words through probabilistic models. *Computational Linguistics*, 19(2):359–382.

Yusuke, Miyao and Tsujii Jun'ichi. 2002. Maximum entropy estimation for feature forests. In *HLT*, pages 292–297, San Diego, CA.