DEPENDENCY-BASED SENTENCE SIMPLIFICATION FOR INCREASING DEEP LFG PARSING COVERAGE

Özlem Çetinoğlu, Sina Zarrieß and Jonas Kuhn
IMS, University of Stuttgart

Proceedings of the LFG13 Conference

Miriam Butt and Tracy Holloway King (Editors)

2013

CSLI Publications

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

Abstract

Large scale deep grammars can achieve high coverage of corpus data, yet cannot produce full-fledged solutions for each sentence. In this paper, we present a dependency-based sentence simplification approach to obtain full parses of simplified sentences that failed to have a complete analysis in their original form. In order to remove the erroneous parts that cause failure, we delete phrases from failed sentences by utilising their dependency structure, and reprocess the remaining shorter sentences with XLE to get full analyses. We ensure the grammaticality and preserve the core argument structure of simplified sentences by defining the deletion scheme only on a set of modifier phrases. We apply our approach on German data and retrieve full parses of simplified sentences for 52.37% of the failed TIGER sentences. With the combination of original and simplified sentences, the full XLE parses derived from the TIGER Treebank increases from 80.66% to 90.79%.

1 Introduction

Over the last two decades, the LFG community has witnessed the development of wide-coverage, deep, hand-crafted grammars for several languages (Butt et al., 2002). These grammars can typically parse over 90% of corpus data, yet cannot produce a full-fledged solution for each sentence. The difficulty of achieving 100% coverage on unrestricted text with a deep parsing approach lies in two aspects: missing lexical items, idiosyncrasies and rare constructions on one hand and ungrammatical material and spelling errors in real language use on the other. One natural solution to overcome such cases is to extend the grammar in question with manual rules for uncovered instances or to relax the existing rules to handle ungrammatical cases. However, such modifications on a large-scale, with an already high-coverage grammar is very labour-intensive and requires a high level of linguistic expertise. Moreover, the flexibility needed to handle erroneous inputs is hard to predict and employ during grammar development due the wide range of possible ungrammaticalities. In this work, we instead pursue an automatic way of dealing with failed sentences.

(1) exemplifies a TIGER sentence [1], a real corpus example that contains an agreement mistake. The article *des* 'of the' and the adjective *japanischen* 'Japanese', both in genitive case, give the hint that it is a genitive construction, but the noun *Außenministerium* 'foreign ministry' is in nominative case and therefore doesn't agree with the surrounding words.

(1)	*Ein*	*Sprecher*	***des***	***japanischen***	***Außenministerium***	*verkündete*	*daraufhin*
	A	speaker	of the	Japanese	foreign ministry	proclaimed	then
	,	*man*	*werde*	*Jelzins*	*Aussage*	*“*	*vorsichtig*	*analysieren*	*”*	*,*	*bevor*
	,	one	would	Yeltsin's	statement	“	carefully	analyze	”	,	before

[1] The TIGER Treebank (Brants et al., 2002) consists of over 50,000 sentences of German newspaper text. All sentences are syntactically annotated and each token contains lemma, POS tag, and morphological information.

man sie kommentiere , aber :
one it comment , but :

'A speaker (of the Japanese foreign ministry) then proclaimed that Yeltsin's statement would be " carefully analyzed " , before commenting on it , but :'

The German ParGram Grammar outputs only fragmented analyses for this sentence as given in Figure 1. Fragments, together with skimming, is a method XLE uses in dealing with robustness (Riezler et al., 2002). Fragmenting allows XLE to output chunk or partial analyses in cases where it cannot produce a complete parse. Skimming, on the other hand, handles time and memory problems. When parsing a sentence exceeds a certain time and memory threshhold XLE spends a limited amount of effort for the remaining constituents, which might lead to suboptimal or partial analyses. It is not possible to avoid such analyses unless we take into consideration parsing the closest well-formed sentence of the original problematic case.

A closer look at Figure 2 depicts the problems in the analysis of (1) more clearly. Both the c-structure and the f-structure of the partial phrase *Ein Sprecher des japanischen Außenministerium verkündete* 'A speaker of the Japanese foreign ministry proclaimed' are fragmented. In the c-structure, *Ein Sprecher des japanischen* 'a speaker of the Japanese' and *Außenministerium verkündete* 'foreign ministry proclaimed' form incorrect phrases. In the corresponding f-structure, *Sprecher* 'speaker' does not get any grammatical role and *Ministerium* 'ministry' is incorrectly analysed as the subject of *verkünden* 'proclaim'.

Such a fragmented analysis is not favorable for our research purposes. Full parses are crucial, for instance, in XLE parse disambiguation (Forst, 2007) and generation reranking (Cahill et al., 2007; Zarrieß et al., 2011) as well as deep syntactic analyses of raw text. This fact motivates investigations into how to gain the sentences we lose due to fragmented parses. For recovery, we have to locate and correct the problem.

The problematic part in (1) is the missing genitive marker 's' at the end of *Außenministerium* 'foreign ministry'. When this word is corrected to *Außenministeriums* the German ParGram Grammar outputs a fully connected c-structure and an f-structure with correct arguments. However automatically correcting this particular problem is not as easy as manual correction. Without 's' *Außenministerium* is a valid word in nominative case; there are no lexical level errors. The error emerges at the syntactic level where the nominative-marked noun prevents a genitive adjunct construction. Deciding how to correct an error is a much harder goal than locating it.

Removing the erroneous part of the sentence instead of correcting it could be an acceptable trade-off for an automatic solution. Full parses including the core argument structure of a sentence is our main interest in parsing a sentence. If the problem is located in some modifier phrase its removal does not harm grammaticality and the full parse of the remaining part is still useful for our purposes. For instance, in parse disambiguation (Forst, 2007), the training data consists of

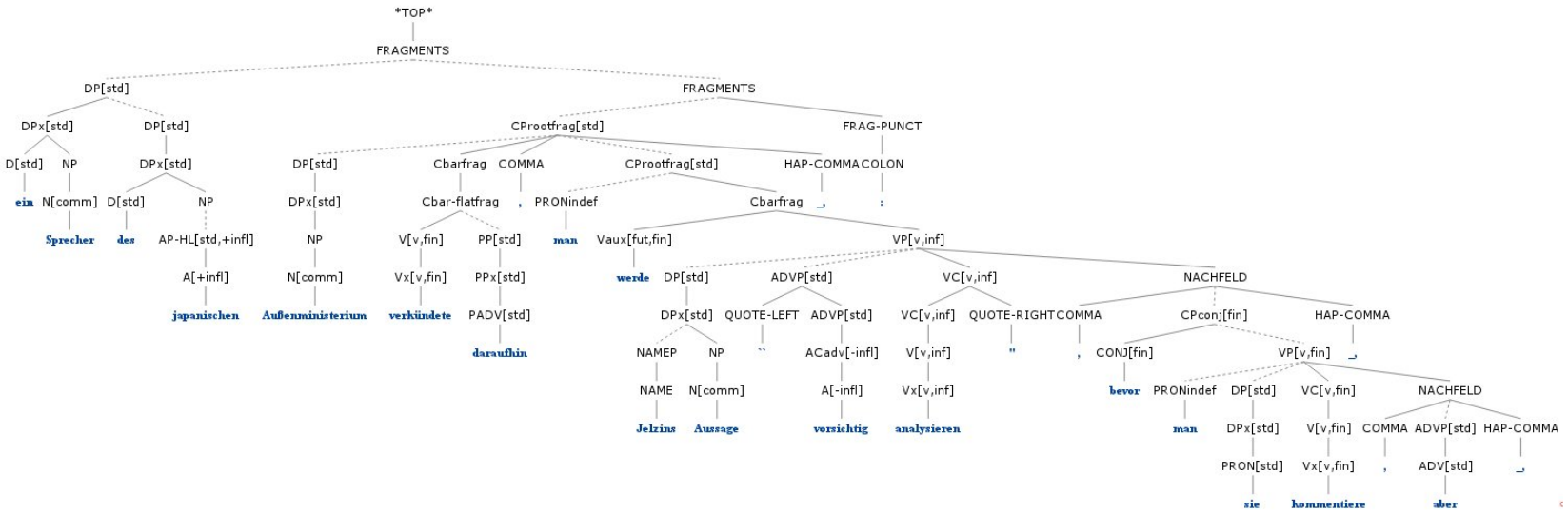

Figure 1: The c-structure of the sentence in (1), processed by the German ParGram Grammar.

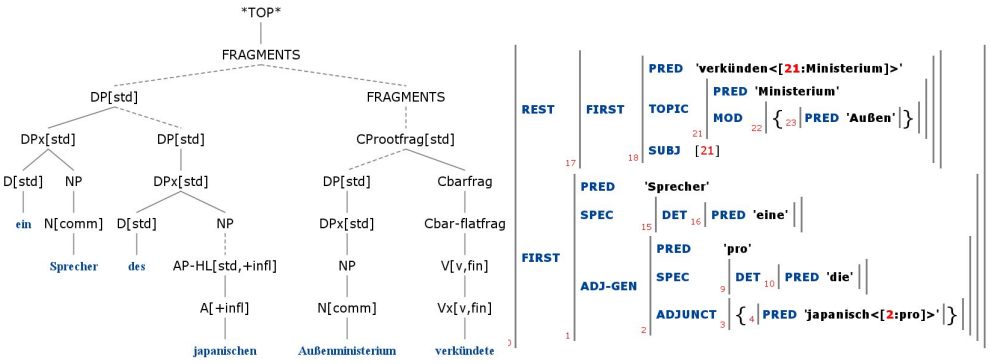

Figure 2: The c- and f-structure of the partial phrase *Ein Sprecher des japanischen Außenministerium verkündete* 'A speaker of the Japanese foreign ministry proclaimed', processed by the German ParGram Grammar.

sentence-full parse pairs. The gold LFG parse of a sentence is determined automatically by matching it to its gold TIGER representation.[2] When a modifier is removed from a sentence, its representation is removed from the gold TIGER tree. The LFG analyses of the simplified sentence are then matched against the simplified TIGER representation to find the gold LFG parse. The deleted parts or meaning changes do not cause the complete withdrawal of the simplified sentence as training data because our main purpose in this application is to identify which LFG parse is the gold one among alternatives for a given sentence.[3]

Employing such a simplification aproach reduces the task of recovering failed sentences to the automatic identification of modifiers. This brings us to the question of how to identify modifiers. Actually, the problematic modifier in sentence (1) can easily be identified by using a dependency tree. Figure 3 depicts the dependency representation of the first six tokens of our example sentence. *Außenministerium* is the head of the genitive adjunct (AG) with dependants *des* and *japanischen*.

We can get the dependency representations of sentences we want to simplify by using dependency parsers. They are easy and fast to train and parse with, and with

[2]Section 4 details the matching.

[3]Our goal is not so much to identify material that can be dropped without changing the meaning of a sentence, but rather material for which we can assume with great confidence that it does not affect the grammaticality of the material that we leave in: e.g., dropping the direct object of a transitive verb will in some cases be unproblematic (like in John sings a song → John sings), but in many cases it will. The simplified sentences can then be safely used as the skeleton to which a full LFG analysis can be attached – and of course, the fact that they are not original corpus material, but have undergone modification, has to be listed with them. The dependency tree information can even be used to indicate the missing material in the LFG structure (an aspect which we do not address in the present paper).

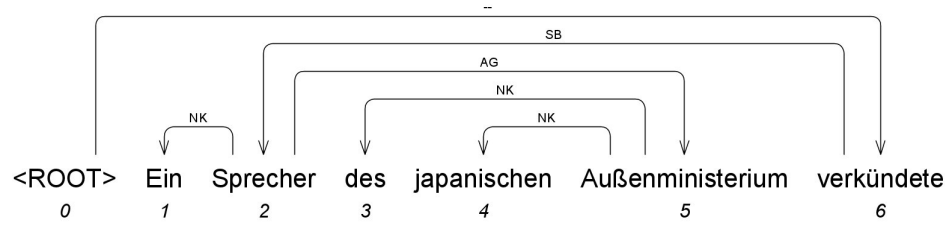

Figure 3: The dependency tree of the partial phrase *Ein Sprecher des japanischen Außenministerium verkündete* 'A speaker of the Japanese foreign ministry proclaimed'.

their robust nature they do not pose coverage problems. Moreover, they are less sensitive to the type of input errors in Figure 3. The dependency parser correctly identifies the phrases and assigns correct labels despite the missing 's' in *Außenministeriums*. When we remove the genitive adjunct (subtree with the head AG) from the dependency tree of the sentence in (1), the German ParGram Grammar fully and correctly parses the remaining sentence. Figure 4 displays the f-structure of the simplified sentence. *Sprecher* 'speaker' is correctly analysed as the subject of *verkünden* 'proclaim' and the complement clause with the head *analysieren* 'analyse' is correctly identified.

This approach can be generalised to a simplification scheme on all sentences an LFG grammar fails to output a full parse. The guiding idea is the following: a state-of-the-art statistical dependency parser is run on any sentence that doesn't receive a full analysis from XLE. Its coverage is 100%, and although the labelled dependency tree analysis that it produces will not be perfect, we can use it as a fairly reliable indication of "non-core" parts of the sentence: appositions, relative clauses, etc. We generate modified versions of the input string in which these parts are deleted and try to reparse them with the original LFG grammar. By using a conservative scheme of deletions, we try to ensure that grammaticality and the core argument structure of the sentences are preserved. Depending on the context of application, the resulting simplified f-structure can be used directly (e.g. as training data), or a synthesized analysis for the original string can be constructed (e.g. for feature extraction in a dependency parsing scenario).

We apply the scheme on German data and achieve full parses of simplified sentences on 52.37% of the failed TIGER sentences. This increases the full XLE parses derived from the TIGER Treebank from 80.66% to 90.79%. We evaluate the accuracy of the simplified f-structures by matching them against the gold TIGER trees and observe that the simplified f-structures match with the gold syntactic structure of the simplified input sentences for the 48.50% of the successfully reparsed cases. We also experiment with the simplification scheme in a non-gold

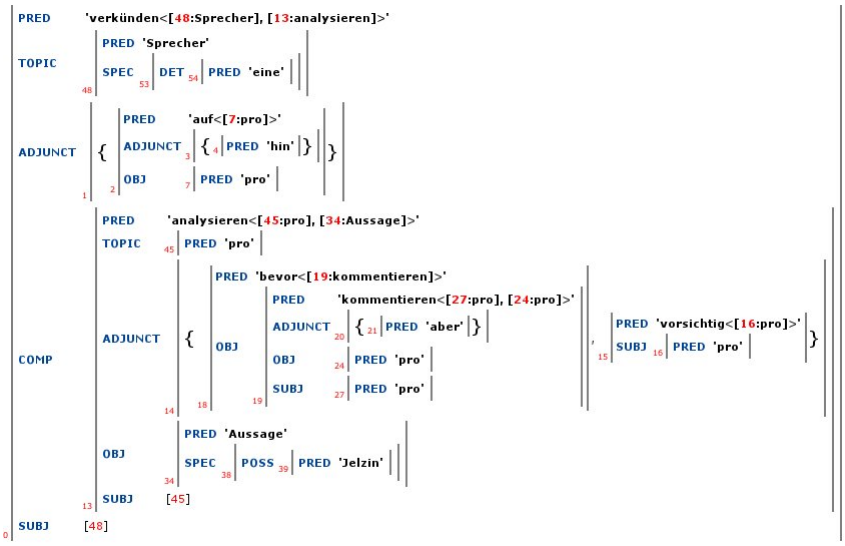

Figure 4: The f-structure of the simplified sentence *Ein Sprecher verkündete daraufhin , man werde Jelzins Aussage " vorsichtig analysieren " , bevor man sie kommentiere , aber :* 'A speaker then proclaimed that Yeltsin's statement would be " carefully analyzed " , before commenting on it , but :'

setting. Parsing results on raw data proves it is possible to apply the proposed system beyond the gold TIGER data.

The remainder of the paper is structured as follows: We look into related work in Section 2. We describe and experiment with sentence simplification in Section 3. We evaluate the accuracy of our system in Section 4 and extend it to parsing raw text in Section 5. An error analysis in Section 6 is followed by conclusion and future work in Section 7.

2 Related Work

Achieving high parsing coverage of LFG grammars has been in the scope of several researchers. Riezler et al. (2002) parse the Wall Street Journal with the English ParGram Grammar. They reach 100% coverage at the expense of fragmented and skimmed analyses. Rohrer and Forst (2006) also make use of skimming and fragmenting in parsing the TIGER Treebank with the German ParGram Grammar. Besides these standard mechanisms, they implement additional rules for linguistic phenomena that cause non-full analyses. Coordination, parentheticals, subject gaps, reported speech clauses are among the refined constructions. Dost and King

(2009) parse and analyse a large set of Wikipedia articles to determine lexical and syntactic gaps in the ParGram English grammar, and improve the grammar coverage by incorporating their findings.

An alternative approach for handling coverage problems is to generate f-structures by annotating statistical phrase-structure parser output with f-structure constraints and by solving those constraints (Cahill et al., 2004). After all, the statistical parsers they are based on are robust. Although the approach is applied to several languages, only the English system's output comes close to XLE f-structure duplicates, which also cannot reach 100% coverage (Hautli et al., 2010).

Sentence simplification in LFG systems has been studied before (Riezler et al., 2003; Crouch et al., 2004), in a different context than ours, paying attention to meaning preservation. Riezler et al. (2003) carry out sentence simplification on English computer news articles by converting parsed f-structures to reduced ones with transfer rules. They then disambiguate and generate from reduced f-structures to obtain shorter sentences. Crouch et al. (2004) further describes the type of rules used in the transfer system.

Although utilising dependencies in sentence simplification is a novel approach in LFG research, dependency-based sentence simplification is an accepted method, often used to extract the important information out of the data in applications such as summarisation (Vanderwende et al., 2006) and spoken language understanding (Tür et al., 2011). Vanderwende et al. (2006) pays attention to grammaticality of simplified sentences whereas Tür et al. (2011) can ignore it, as the simplified dependency structures are used as features in a statistical classifier. Both systems work on English data.

On the German front, there has been research on deep parsing systems that utilise dependencies, mainly by developing hybrid approaches. Schiehlen (2003) parses the NEGRA Treebank with a combination of a shallow approach based on machine learning and a cascaded finite state parser. Frank et al. (2003) bring together a topological fields parser with a wide-coverage HPSG parser. Our system differs from those in making use of dependencies in sentence simplification decisions.

3 Sentence Simplification

We employ two different approaches to simplify sentences. The first one utilises the dependency representation of sentences to be simplified. We initially remove only one subtree at a time and then further simplify the trees with multiple subtree removal. As an alternative to dependency-based simplification we introduce an n-gram-based approach, which constitutes our baseline system.

3.1 Implementation Setup

We use the TIGER treebank (Brants et al., 2002) 2.1 as our data set in our experiments. We leave out sentences 8000 to 10000 as test and development sets following the TiGerDB split (Forst et al., 2004) and work on the training set which corresponds to the remaining 48471 sentences. We parse our corpus with a version of the German ParGram grammar (Rohrer and Forst, 2006) and use the dependency version of the TIGER Treebank which is converted by Seeker and Kuhn (2012). The edge labels used in the conversion are taken from the STTS tag set (Schiller et al., 1999).

3.2 Dependency-based Simplification

For our simplification process we manually define a set of deletable dependency subtrees that would not harm the grammaticality of a sentence and preserve the core argument structure. Table 1 gives the list of edge labels that correspond to the heads of deletable subtrees. In most of the cases, all instances of an edge label are deletable. However there are some conditional deletions. For instance a noun kernel (NK), which is an umbrella label for various relations inside of the nominal phrase including the head relations, is deletable only when it functions as an adjunct. Datives are reasonable candidates for deletion because they may be free datives, like *ihm* 'him' in the example *Sie backte ihm einen Kuchen* 'She baked him a cake'.[4]

The simplification script traverses the dependency tree of a sentence to be simplified and looks for a deletable subtree. Once a subtree is removed from the original tree, its yield is deleted from the original sentence. The shorter sentence we get out of the deletion might need punctuation adjustments. The TIGER Treebank does not necessarily attach all punctuation to a relevant node. The TIGER-to-dependency conversion tool assigns its previous token as the head of a punctuation mark. This approach might clash with the phrase boundaries, that is, there could be dangling punctuation tokens after simplification. Such cases are handled with a set of rules within the simplification script. The outcome of the script is a candidate sentence for reparsing with the German ParGram Grammar.

3.2.1 One Subtree Deletion

As our initial experiment, the simplification script goes through the list of deletable labels and removes only one deletable subtree at a time from a sentence. The number of candidates generated for each sentence varies depending on the number of deletable dependencies it contains. This procedure produces 52867 candidates, that is, 5.6 candidates per sentence on average. We process all the candidates with the German ParGram Grammar.

[4] In order to capture free datives, we deviate from our conservative deletion scheme that preserves core arguments. This rule might cause the deletion of datives that are part of a subcategorisation frame.

AG	genitive adjuncts
APP	appositions
JU	discourse marker-like
MNR	PP adjuncts (in noun phrases)
MO	modifiers
NG	negation
PAR	head of parenthesis
PG	possessive PP adjuncts
PH	placeholders (e.g. German Vorfeld *es*[5])
PNC	proper noun components
RC	relative clauses
RE	infinite clauses attached to nominals
SBP	PP subjects in passive
UC	inside foreign language phrases
VO	vocatives
NK	noun kernels
DA	datives

Table 1: Edge labels of deletable subtrees that are used in the dependency-based sentence simplification system.

3.2.2 Multiple Subtree Deletion

Failure in parsing a sentence might stem from problems in multiple subtrees. Hence, removing only one subtree is not sufficient to obtain a full parse in some cases. In order to simplify sentences even further, we let the simplification script generate all possible subtrees by deleting all combinations of deletable dependencies. This approach brings an overhead: the number of candidates grows very high for longer sentences. It can reach up to 608,255 candidates for a sentence and the average number of candidates per sentence is 924. Hence, the total number of candidates is not feasible for parsing. As a solution, we take only the 10 shortest candidates into account when a sentence has more than 10 candidates. We also remove the punctuation of the shortest candidate and include it as the 11th candidate. The average number of candidates per sentence drops down to 8.1 this way.

(2) marks all deletable subtrees for the sentence in (1). In the one subtree deletion setting, five candidates are produced out of this sentence. In the multiple subtree deletion setting, there are 63 alternatives; we only take the shortest 10. (3) shows the correctly parsed candidates among those 10 sentences. The deleted subtrees are represented with the edge labels of their heads.

[5]If a Vorfeld modifier is deleted, the sentence becomes ungrammatical due to word order. It is possible to reorder the sentence by using a lineariser (Bohnet et al., 2012) to achieve a grammatical order again.

(2) *Ein Sprecher [$_{AG}$ des japanischen Außenministerium] verkündete [$_{MO}$*
A speaker of the Japanese foreign ministry proclaimed
daraufhin] , man werde Jelzins Aussage “ [$_{MO}$ vorsichtig]
then , one would Yeltsin's statement “ carefully
analysieren ” , [$_{MO}$ bevor man sie kommentiere ,] [$_{MO}$ aber] :
analyze ” , before one it comment , but :
‘A speaker [of the Japanese foreign ministry] [then] proclaimed that Yeltsin's statement would be “ [carefully] analyzed ” , [before commenting on it ,] [but] :’

(3) a. Ein Sprecher [$_{AG}$] verkündete [$_{MO}$], man werde Jelzins Aussage “ [$_{MO}$] analysieren ” [$_{MO}$] [$_{MO}$]:

b. Ein Sprecher [$_{AG}$] verkündete daraufhin , man werde Jelzins Aussage “ [$_{MO}$] analysieren ” [$_{MO}$] [$_{MO}$]:

c. Ein Sprecher [$_{AG}$] verkündete [$_{MO}$], man werde Jelzins Aussage “ vorsichtig analysieren ” [$_{MO}$] [$_{MO}$]:

3.3 N-gram Based Sentence Simplification

One might argue that a dependency-based simplification system is costly due to parsing times and resources to train parsers. We pursue the question if it is possible to benefit from a simpler simplification system in regaining failed sentences. Our baseline simplification technique is based on n-grams. We utilise van Noord's (2004) parsability metric, which is mainly designed for error mining purposes. Parsability of a word is defined as the ratio of a word's occurrence in successful parses $C(w|OK)$ to its occurrence in all sentences $C(w)$:

$$P(w) = \frac{C(w|OK)}{C(w)}$$

The metric is extendable to word sequences. When the sequence is represented as $w_i \dots w_j$, the parsability of a word sequence is:

$$P(w_i \dots w_j) = \frac{C(w_i \dots w_j|OK)}{C(w_i \dots w_j)}$$

In order to incorporate the parsability concept into our sentence simplification system, we extract the n-grams (n=1,2,3) of failed sentences.[6] We then calculate the number of occurrences of those n-grams in failed sentences and in the whole treebank, hence their parsability scores. We delete the n-grams with zero parsability to obtain the simplified candidates. If there are no n-grams with zero parsability scores in the sentence we delete the n-gram with the lowest parsability. We apply a set of punctuation correction rules to this approach too. We achieve a set of 26822 candidates after the simplification and we reparse all these sentence with the German ParGram Grammar. Note that this approach does not ensure the grammaticality of a simplified sentence or the preservation of argument structure.

[6] We limit ourselves to small numbers of n due to our small corpus size.

For the sentence in (1), the simplified sentences are given in (4), showing the deletion of unigrams, bigrams, and trigrams respectively. None of the simplifications in this example leads to a grammatical sentence, therefore no full analyses are obtained after reprocessing. Note that *Außenministerium* that causes the parse failure is not among the deleted n-grams, since its occurrences as a nominative noun have full analyses that increase its overall parsability score. No full analyses and no correct identification of the problem in the given sentence indicate the n-gram based model can fail in cases the dependency-based model achieves success.

(4)
a. Ein Sprecher des japanischen Außenministerium verkündete daraufhin , man werde Jelzins Aussage " vorsichtig analysieren " , bevor man sie ~~kommentiere~~ , aber :
b. Ein Sprecher des japanischen Außenministerium verkündete daraufhin , man werde Jelzins Aussage " vorsichtig ~~analysieren "~~ , bevor man sie kommentiere , aber :
c. Ein Sprecher des ~~japanischen Außenministerium verkündete daraufhin ,~~ man ~~werde Jelzins Aussage " vorsichtig analysieren " , bevor man sie kommentiere ,~~ aber :

3.3.1 Coverage Results

Table 2 gives the overview of coverage statistics. 80.66% of the TIGER training set has full XLE parses. The remaining 9373 sentences constitute the set to be simplified.

Our baseline n-gram system can generate fully parsed simplified sentences for 2893 (30.87%) of the cases. When only one subtree is deleted, 3367 (35.92%) sentences have at least one simplified form with a full parse. When all possible candidates are created and 10 shortest are chosen, the number of sentences with at least one simplified full parse increases to 4607 (49.83%). We also create a combination of full parses where all full parses with one subtree deletion are taken and the set of candidates that one subtree deletion failed but multiple subtree deletion succeeded to produce full parses are added. This combination increases the number of full parses to 4909 with a coverage of 52.37%.

Note that the upper limit of simplified sentences with a full parse is 8462 (90.28%) because 911 sentences are not simplifiable at all. 58.02% of the simplifiable sentences achieve a full parse in the combination system.

4 Accuracy of the Simplification System

The full f-structure output of a simplified sentence exhibits valid syntactic structures for that sentence. But it is possible that the syntactic structures the grammar produces do not match the underlying syntactic structure of the corresponding simplified sentence.

In evaluating the parses we achieve, we can take advantage of the gold TIGER Treebank trees. They represent the gold syntactic structure of a given sentence

System	sent.	full parses
TIGER Training	48471	39098 (80.66%)
n-gram deletion	9373	2893 (30.87%)
1 subtree shorter	9373	3367 (35.92%)
10 shortest	9373	4607 (49.83%)
1 subtree shorter + 10 shortest	9373	4909 (52.37%)

Table 2: Full parse statistics when the original training sentences are used, n-gram simplification is used in failed sentences, only one subtree is deleted in simplification, and 10 shortest candidates are parsed in the multiple subtree simplification.

in TIGER XML representation; we can compare the XLE parses with these gold structures. However, this comparison is not straightforward. LFG and TIGER represent the same sentence in different structures. The correspondence neither between functions nor between features is one-to-one. There are ambiguous mappings such as modifiers (MO) in the TIGER Treebank. In LFG, they could be realised as an ADJUNCT, a directional oblique (OBL-DIR), a locative oblique (OBL-LOC), or a manner oblique (OBL-MANNER), depending on the context. There are also structural dissimilarities, for instance, the treatment of auxiliary verbs. In TIGER trees the main verb is dependent on the auxiliary as its clausal object (OC), whereas in LFG the main verb is the head and the auxiliary contributes to tense and aspect features.

Forst (2007) introduces a matching system that compares XLE parses with TIGER trees by converting the TIGER XML representation to f-structures through XLE's tranfer rules. We apply his approach to simplified sentences and check if XLE parses are compatible with TIGER trees as our accuracy evaluation. This comparison, at the same time, produces a set of XLE parses that have gold f-structures. Later these so called TIGER-compatible f-structures are used in parse disambiguation and generation reranking.

The TIGER-compatible f-structures are given in Table 3. In the TIGER training set 11,931 (30.53%) sentences have a gold f-structure. When one subtree is removed in the simplification of failed sentences, 665 out of 3367 sentences get a TIGER-compatible f-structure. The accuracy increases to 50.90% with a set of 2345 sentences in the multiple subtree simplification. When one and multiple subtree simplification are combined, the number of full parses increases to 4909 from 4607. But not as many of the additional sentences have TIGER-compatible parses, hence the accuracy of the system drops to 48.50%.

System	sent.	full parses	TIGER-compatible
TIGER Training	48471	39098	11931 (30.53%)
1 subtree shorter	9373	3367	665 (19.75%)
10 shortest	9373	4607	2345 (50.90%)
1 subtree shorter + 10 shortest	9373	4909	2381 (48.50%)

Table 3: TIGER-compatible f-structure statistics. The percentages show the ratio of TIGER-compatible parses to full parses.

5 Getting Dependencies from Raw Text

Our experiments so far are conducted on the gold TIGER trees. Working on gold trees is crucial to obtain training material for the disambiguation/reranking systems. Another reason we want to improve the coverage of LFG parsing is to utilise the features derived from deep analyses in dependency parsing. There could be two approaches to follow in feature integration: The LFG features of each input sentence are used by the dependency parser during parsing time or the LFG features can be used as hard or soft constraints during training time. In either case the accuracy and high coverage of LFG parses facilitate dependency parsing.

Approaches on integrating deep LFG analyses into dependency parsing are realistic only when we do not limit ourselves to gold representations. To see if our system is extendable to real-world scenarios, we simulate such a system in a predicted setting. We use predicted lemma, POS, and morphological features[7] and parse the TIGER sentences with a statistical dependency parser (Bohnet, 2010). All systems are trained on the TIGER training data by using 10-fold cross-validation. The results of the predicted setting are given in Table 4. It can be observed that for all systems the difference between the predicted and gold setting is quite low (1.66%, 3.46%, 1.82% absolute for 1 subtree shorter, 10 shortest, and the combination system respectively). This proves that the simplification approach we employ can easily be applied to realistic, non-gold scenarios too.

6 A Closer Look into Failed Sentences

Our work mainly focuses on ways of recovering the sentences lost during parsing. As a byproduct it gives us insight into the weak and strong points of the modules we employ. The parsability metric we used in n-gram-based simplification also lists the word sequences that cannot be parsed, as originally used in error detection. Table 5 lists the 10 most frequent n-grams with zero parsability. The

[7] all processed with mate lemmatiser, POS tagger, and morphological analyser: http://code.google.com/p/mate-tools/.

Data	**System**	sent.	full parses
Predicted	1 subtree shorter	9373	3211 (34.26%)
	10 shortest	9373	4346 (46.37%)
	1 subtree shorter + 10 shortest	9373	4738 (50.55%)
Gold	1 subtree shorter	9373	3367 (35.92%)
	10 shortest	9373	4607 (49.83%)
	1 subtree shorter + 10 shortest	9373	4909 (52.37%)

Table 4: TIGER coverage statistics for predicted and gold systems.

sign denotes sentence boundaries. It can be observed that lowercase abbreviations of news agencies such as AFP, DPA, and RTR are among the most frequent sources of error. These abbreviations are often used in a template `venue, date (agencies)`, e.g., *AUCKLAND, 9. November (dpa/rtr)*, which is not handled well by the German ParGram Grammar. *ski* is a one-word sentence, tagged as a non-word (XY) in the TIGER Treebank. *90/Die* comes from the political party *Bündnis 90/Die Grünen* 'Allience 90/The Greens', and similarly *CDU/CSU* is referred to as the union of two political parties. Those tokens have high parsability on their own but when combined into a coordination with a slash as the conjunction, the German ParGram Grammar cannot parse them. *Befreiungstiger von Tamil* is part of a longer multiword *Befreiungstiger von Tamil Eelam (LTTE)* 'Liberation Tigers of Tamil Eelam'. Actually *Befreiungstiger von Tamil* 'Liberation Tigers of Tamil' can be correctly parsed by the German ParGram Grammar. What it fails is to combine *Tamil* and *Eelam* as a phrase. As a consequence, the analysis of the whole phrase consists of two fragments *Befreiungstiger von Tamil* and *Eelam (LTTE)*.

Parsability	Count	n-gram
0.000	11	Befreiungstiger von Tamil
0.000	11	CDU / CSU
0.000	17	# ski #
0.000	29	90 / Die
0.000	31	/ dpa /
0.000	34	afp / dpa
0.000	38	# (...
0.000	40	# (rtr
0.000	41	dpa / rtr
0.000	95	# (dpa

Table 5: 10 most frequent n-grams with zero parsability, derived from 9373 failed sentences.

Another way of analysing problematic cases is looking into the phrases that enabled full parses after their deletion. The most frequent such phrases are given in Table 6 in the one subtree simplification setting. The most frequent phrase *sich* 'oneself' is used in reflexive verbs. Most German reflexive verbs have a non-reflexive version too, and it seems when *sich* is dropped from the sentence the German ParGram grammar can parse the sentences in the non-reflexive form. The second most frequent phrase that deletion helps is the negation adjunct *nicht* 'not', to our surprise. Negation is well handled in a hand-crafted large-scale grammar after all. An investigation into sentences fully parsed after removing *nicht* reveals that the word order of *nicht* is not canonical in such cases. (5) exemplifies such a sentence with *nicht* intervening between the predicative adjective and the finite copula. *ihm* is the third person masculine or neutral personal pronoun in dative case, and its deletion helps due to free datives (cf. section 3.2). The remaining frequent phrases that prevent full parses in the original sentences are adverbs *so* 'so', *auch* 'too', *immer* 'always', *rund* 'around'[8], *nur* 'only', and *aber* 'but'.

Count	Phrase
189	sich
82	nicht
40	so
34	auch
22	ihm
18	Immer
16	rund
15	nur
15	aber

Table 6: The most frequent (count $>$ 15) deleted phrases that enable their simplified sentences to have full parses.

(5) *Daß es so einfach* ***nicht*** *ist , weiß natürlich auch der*
That it so easy **not** is , knows naturally also the
CDU-Politiker .
CDU politician .
'Of course the CDU politician also knows it is **not** so easy.'

The top 21 entries of the deleted phrases list is one-token phrases. The highest ranked multiple-token phrase is *von Bündnis 90/Die Grünen* 'from Allience 90/The Greens' with 10 occurrences, which comfirms our findings in Table 5. The length of deleted phrases goes up to 91 tokens. Figure 5 displays the distribution of phrase lengths in the scenario where deleting one subtree solves the failure problem and the shorter sentence achieves a full parse. Deleting only one token is enough for

[8]Modifying numerals in these occurrences.

a full parse in 1732 of the cases. Deleting 2, 3, 4 tokens helps 770, 525, 371 sentences respectively. For 2205 of the sentences, omitting phrases of 5 or more tokens are necessary for a full parse.

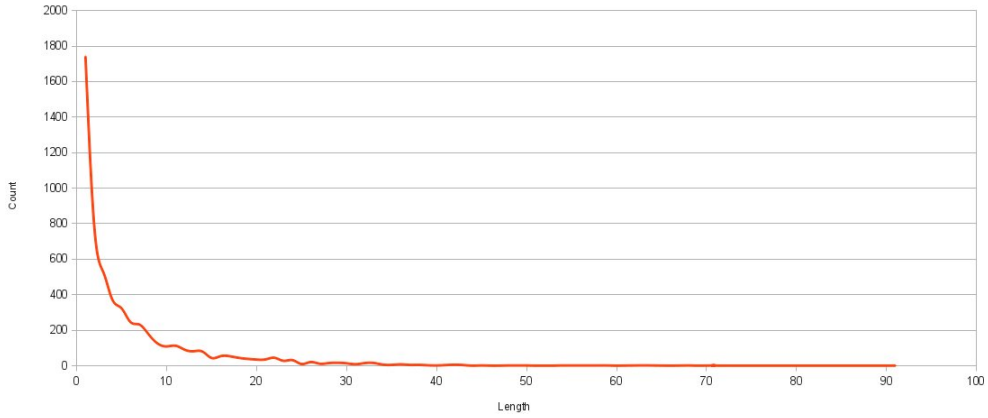

Figure 5: The distribution of phrase lengths that are deleted in the one subtree simplification and enabled full parses in the simplified sentence.

7 Conclusion and Future Work

We have presented a dependency-based simplification approach that increases the coverage of full LFG parses of the TIGER Treebank for German. Our approach is based on the idea of removing the modifiers from a sentence with no full analyses and reparsing the shorter sentence with the German ParGram Grammar. If the problematic part that caused a parse failure is in the removed modifier, we achieve a full parse of the simplified sentence. This method ensures grammaticality while it preserves the core parts. We utilise the dependency representations of sentences to identify modifiers. A simplification script traverses the dependency tree of a sentence and deletes subtrees based on a manually collected list of deletable functions. The outcome is a set of candidate sentences to be reparsed.

Experiments on the TIGER Treebank show 52.37% of the failed sentences achieve at least one full parse in their simplified form. And among those simplified full sentences 48.50% of the analyses are compatible with the underlying analyses of simplified versions of the original sentences. As a comparison we experiment with an n-gram-based simplification system and observe that the dependency-based system clearly outperforms the n-gram one, in addition to its superiority in grammatical simplifications. In order to test the reliability of our system in a real-world scenario we repeat the simplification experiments in predicted settings. The results we achieve are comparable to the gold settings, showing that we can apply simplification on dependency parser outputs too.

An error analysis on the simplified sentences shows there are some systematic failures in the German ParGram Grammar due to domain-specific tokens or constructions (e.g., news agencies, venue-date boilerplates). Some commonly used adverbs also prevent the sentences from having full parses when they are used in non-canonical ways.

The simplification method we proposed in this paper can be a basis to future research in several directions. XLE parse disambiguation (Forst, 2007) and generation reranking (Cahill et al., 2007; Zarrieß et al., 2011) are among our initially motivating applications. We plan to revisit both systems by investigating the effect of using the extended set of TIGER-compatible f-structures. An additional application of sentence simplification which we can pursue is to simplify fully parsed sentences that TIGER-compatibles f-structures cannot be extracted from. This could provide extra training material for parse disambiguation and generation reranking.

Another natural extension to the current work is incorporating the deep syntactic representations XLE grammars output as features of a dependency parser. Øvrelid et al. (2009) converts the LFG output into dependencies to allow parser stacking. Experiments show features extracted from deep LFG structures help improve parser accuracies both for English and for German. As our future work, we aim at investigating more ways of integration.

Both systems using the f-structures directly and systems using them as features would benefit from even more improvements in coverage. We aim to advance the simplification system both in terms of coverage of failed sentences and in terms of the content of the full reparses.

0.97% of the failed TIGER sentences are not simplifiable with the existing simplification scheme. It means the errors that prevent full parses reside in the core arguments of those sentences and it is not possible to delete them. In such cases a paraphrasing approach can replace a simplification approach. For instance, nominal phrases can be replaced with easier-to-parse nominals, e.g. pronouns, or coordinations can be replaced with one of their conjucts. Obviously, this paraphrasing technique can be used within the set of the simplifiable sentences too.

When there are too many candidates to parse in the multiple subtree simplification setting, we choose 10 shortest candidates. That simple criterion prevents us from finding longer fully parsed candidates. We can improve the criterion by paying attention to parsability scores of subtrees to be parsed. If we eliminate the subtrees including word sequences with zero or low parsability, we reduce the number of candidates in an earlier step, and further include longer sentences into the candidate list.

The simplification technique we propose is language independent except for the manual selection of deletable subtrees. Therefore, it is possible to apply it to several existing LFG grammars with relatively little effort. Especially for grammars under development, simplification both handles coverage issues and points out areas to improve. We are specifically interested in Turkish, as we possess the requirements for its application: native speaker knowledge for deletable subtree identification, a

ParGram grammar (Sulger et al., 2013), and a state-of-the-art dependency parsing system (Çetinoğlu and Kuhn, 2013).

Acknowledgements

This work is funded by the project D2 of the Collaborative Research Centre (SFB 732) “Incremental Specication in Context”.

References

Bohnet, Bernd. 2010. Top Accuracy and Fast Dependency Parsing is not a Contradiction. In *Proc. of COLING*, pages 89–97, Beijing, China.

Bohnet, Bernd, Björkelund, Anders, Kuhn, Jonas, Seeker, Wolfgang and Zarriess, Sina. 2012. Generating Non-Projective Word Order in Statistical Linearization. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 928–939, Jeju Island, Korea: Association for Computational Linguistics.

Brants, Sabine, Dipper, Stefanie, Hansen, Silvia, Lezius, Wolfgang and Smith, George. 2002. The TIGER Treebank.

Butt, Miriam, King, Tracy Holloway, Masuichi, Hiroshi and Rohrer, Christian. 2002. The Parallel Grammar Project. In N.Oostijk J. Carroll and R. Sutcliffe (eds.), *Proceedings of the Workshop on Grammar Engineering and Evaluation*, pages 1–7, cOLING02.

Cahill, Aoife, Burke, Michael, O’Donovan, Ruth, Van Genabith, Josef and Way, Andy. 2004. Long-Distance Dependency Resolution in Automatically Acquired Wide-Coverage PCFG-Based LFG Approximations. In *(ACL’04)*.

Cahill, Aoife, Forst, Martin and Rohrer, Christian. 2007. Stochastic realisation ranking for a free word order language. In *ENLG ’07*.

Çetinoğlu, Özlem and Kuhn, Jonas. 2013. Towards Joint Morphological Analysis and Dependency Parsing of Turkish. In *Proceedings of the Second International Conference on Dependency Linguistics (DepLing 2013)*, pages 23–32, Prague, Czech Republic: Charles University in Prague, Matfyzpress, Prague, Czech Republic.

Crouch, Richard, King, Tracy Holloway, Maxwell III, John T., Riezler, Stefan and Zaenen, Annie. 2004. Exploiting F-structure Input for Sentence Condensation. In Miriam Butt and Tracy Holloway King (eds.), *Proceedings of LFG Conference*, Christchurch, New Zealand: CSLI Publications.

Dost, Ascander and King, Tracy Holloway. 2009. Using Large-scale Parser Output to Guide Grammar Development. In *Proceedings of the 2009 Workshop on Grammar Engineering Across Frameworks (GEAF 2009)*, pages 63–70, Suntec, Singapore: Association for Computational Linguistics.

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

Forst, Martin, Bertomeu, Núria, Crysmann, Berthold, Fouvry, Frederik, Hansen-Schirra, Silvia and Kordoni, Valia. 2004. Towards a dependency-based gold standard for German parsers – The TiGer Dependency Bank. In *LINC '04*.

Frank, Anette, Becker, Markus, , Markus Becker, Crysmann, Berthold, Kiefer, Bernd, Schfer, Ulrich and Of, Dfki Gmbh School. 2003. Integrated Shallow and Deep Parsing: TopP meets HPSG. In *ACL 2003*.

Hautli, Annette, Çetinoğlu, Özlem and van Genabith, Josef. 2010. Closing the Gap Between Stochastic and Hand-crafted LFG Grammars. In Miriam Butt and Tracy Holloway King (eds.), *Proceedings of the LFG Conference*.

Øvrelid, Lilja, Kuhn, Jonas and Spreyer, Kathrin. 2009. Improving data-driven dependency parsing using large-scale LFG grammars. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 37–40, Suntec, Singapore: Association for Computational Linguistics.

Riezler, Stefan, King, Tracy H., Crouch, Richard and Zaenen, Annie. 2003. Statistical Sentence Condensation using Ambiguity Packing and Stochastic Disambiguation Methods For LFG. In *HLT-NAACL*.

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

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

Schiehlen, Michael. 2003. Combining Deep and Shallow Approaches in Parsing German. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 112–119, Sapporo, Japan: Association for Computational Linguistics.

Schiller, Anne, Teufel, Simone and Stöckert, Christine. 1999. Guidelines für das Tagging deutscher Textcorpora mit STTS. Technical Report, Technical Report, University of Stuttgart, 1999.

Seeker, Wolfgang and Kuhn, Jonas. 2012. Making Ellipses Explicit in Dependency Conversion for a German Treebank. In *Proceedings of the 8th International*

Conference on Language Resources and Evaluation, pages 3132–3139, Istanbul, Turkey: European Language Resources Association (ELRA).

Sulger, Sebastian, Butt, Miriam, King, Tracy Holloway, Meurer, Paul, Laczkó, Tibor, Rákosi, György, Dione, Cheikh Bamba, Dyvik, Helge, Rosén, Victoria, De Smedt, Koenraad, Patejuk, Agnieszka, Çetinoğlu, Özlem, Arka, I Wayan and Mistica, Meladel. 2013. ParGramBank: The ParGram Parallel Treebank. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 550–560, Sofia, Bulgaria: Association for Computational Linguistics.

Tür, Gökhan, Hakkani-Tür, Dilek, Heck, Larry and Parthasarathy, S. 2011. Sentence Simplification for Spoken Language Understanding. In *IEEE ICASSP*.

van Noord, Gertjan. 2004. Error Mining for Wide-Coverage Grammar Engineering. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 446–453, Barcelona, Spain.

Vanderwende, L., Suzuki, H. and Brockett, C. 2006. Microsoft Research at DUC 2006: Task-Focused Summarization with Sentence Simplification and Lexical Expansion. In *Document Understanding Workshop, HLT-NAACL 2006*.

Zarrieß, Sina, Cahill, Aoife and Kuhn, Jonas. 2011. Underspecifying and Predicting Voice for Surface Realisation Ranking. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1007–1017, Portland, Oregon, USA: Association for Computational Linguistics.