# Making Ellipses Explicit in Dependency Conversion for a German Treebank

**Wolfgang Seeker and Jonas Kuhn**

Institut für Maschinelle Sprachverarbeitung
University of Stuttgart
Firstname.Lastname@ims.uni-stuttgart.de

**Abstract**

We present a carefully designed dependency conversion of the German phrase-structure treebank TiGer that explicitly represents verb ellipses by introducing empty nodes into the tree. Although the conversion process uses heuristics like many other conversion tools we designed them to fail if no reasonable solution can be found. The failing of the conversion process makes it possible to detect elliptical constructions where the head is missing, but it also allows us to find errors in the original annotation. We discuss the conversion process and the heuristics, and describe some design decisions and error corrections that we applied to the corpus. Since most of today's data-driven dependency parsers are not able to handle empty nodes directly during parsing, our conversion tool also derives a canonical dependency format without empty nodes. It is shown experimentally to be well suited for training statistical dependency parsers by comparing the performance of two parsers from different parsing paradigms on the data set of the CoNLL 2009 Shared Task data and our corpus.

**Keywords:** verb ellipses, dependency corpus, German TiGer treebank

## 1. Introduction

Dependency syntax (Mel'čuk, 1988) provides intuitive, light-weight descriptions especially suited to languages with free word order, since it directly represents the function of a word with respect to its head, and it can capture discontinuous structures like extraposition by non-projective trees (i. e. allowing edges to cross). However, unlike constituent-based approaches, it can run into problems for phenomena like head-ellipses, if the dependency representation is restricted to the surface of the sentence, as is for example assumed by all data-driven dependency parsers.[1] An elliptical structure where the head of a phrase is elided therefore causes problems for the representation, because it is not clear where the daughters of an 'empty head' should be attached.

In this work, we present a conversion of the German TiGer treebank (Brants et al., 2002) where we introduce empty nodes for verb ellipses if a phrase normally headed by a verb is lacking a head. Empty heads are thus explicitly represented in the syntactic structure, as it was done for example in the Hungarian corpus (Vincze et al., 2010). We restrict ourselves to annotating verb ellipsis because it is the most frequent ellipsis and it can be annotated automatically without risking to annotate too many incorrect ellipses. The conversion process is still heuristics driven as are most conversion tools, however, in our case, the head-finding heuristics for the TiGer treebank are constructed in a way such that only plausible heads are considered. For example, the head of a verb phrase in TiGer is explicitly marked by the function label *HD*. If no node is marked by *HD*, the conversion tool introduces an empty node instead of just choosing arbitrarily between the other nodes, which is the common way of handling these cases.

When a lot of statistical dependency parsers were developed in the last years, the need for training data led to the conversion of traditional phrase-structure treebanks to dependency format. But a rather rare phenomenon like ellipsis,[2] which in phrase-structure format can be easily represented by annotating an unheaded phrase node, would usually not receive special attention. Figure 1 shows the annotation of a sentence from three different dependency versions of TiGer. The top tree is from the corpus used in the PaGe Shared Task (Kübler, 2008), the middle tree is from the CoNLL 2009 Shared Task (Hajič et al., 2009), and the last tree was created by our conversion tool. In the original TiGer annotation, the sentence is analysed as two coordinated main clauses, where the first one is missing a passive auxiliary verb, while the second one is missing the finite auxiliary verb. This sentence is rather complex, since the two conjuncts are sharing some material. The missing word in the first conjunct does exist in the second conjunct, and vice versa.

In the top tree, the second conjunct is not represented, but all its parts are attached to the root node. Note that the tool by Versley (2005) that was used to derive this tree changes the function labels to comply with the grammar used by Foth et al. (2004). In the middle tree, the sentences are represented and the coordination is annotated (although due to the ellipsis, two words are coordinated that would not normally be coordinated). The passive constructions in the two sentences are structurally atypical since they are missing elements. In the bottom tree finally, our conversion tool introduced two empty nodes to represent the missing words in the incomplete passive constructions. Structurally, the sentences are now looking like other non-elliptical passive constructions.

The tool that we present in this work derives a dependency representation from the original TiGer corpus that explicitly represents empty verbal heads. The annotation scheme follows the annotation of the CoNLL 2009 Shared Task data set, but improves on it in a number of ways. By explicitly

[1] In dependency syntax theory for example, the ellipsis can be handled in a non-surface structure, cf. (Mel'čuk, 1988).

[2] It is rather rare in the Tiger corpus, which however does not mean that it would be rare in other language domains as well.

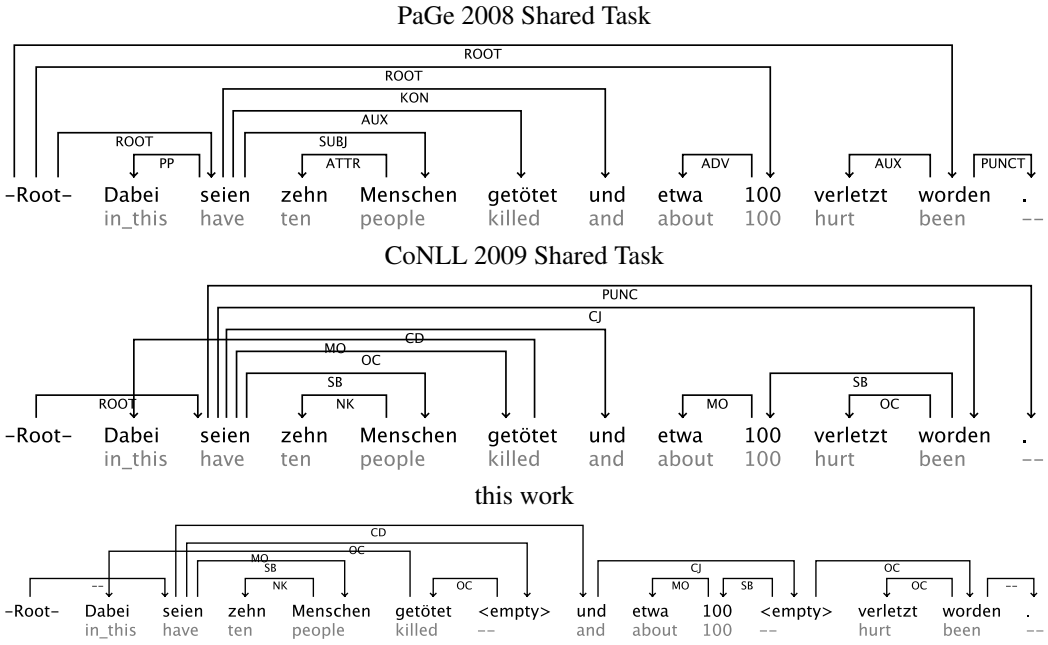



Figure 1: Three different annotations of an elliptical construction in German (TiGer no. 2,209).

representing empty nodes in the annotation, we make the information about ellipsis better accessible in the corpus. Furthermore, we make it possible to develop and evaluate dependency parsing methods that can handle elliptic constructions in German. In addition, the tool can also resolve the empty nodes by replacing them with one of their daughters in order to produce a canonical dependency representation. By comparing the performance of two statistical dependency parsers on the canonical version and the CoNLL 2009 Shared Task data, we show that our conversion is better suited to train dependency parsing models for German.

In the following, after discussing related work in Section 2., we first present the general conversion algorithm by defining the rules that our tool applies (Section 3.) and discuss some differences to the CoNLL 2009 Shared Task data set. We then show how verb ellipses are identified in the derivation process and give some statistics on the distribution of empty nodes (Section 4.). Section 5. describes error correction and problematic cases for the conversion, and in Section 6. we show for two representative parsers that the annotation of the new corpus is superior to the one present in the CoNLL 2009 Shared Task data. Section 7. concludes the paper.

## 2. Related Work

Since the rise of data-driven dependency parsing (Nivre et al., 2004; McDonald et al., 2005), traditional phrase-structure treebanks of many languages have been transformed to dependency format to fulfill the need for training data for statistical parsing models (among others, English (Magerman, 1994; Yamada and Matsumoto, 2003; Johansson and Nugues, 2007), German (Daum et al., 2004; Hajič et al., 2009), and Hungarian (Vincze et al., 2010)). While there have been grammar-based approaches (e. g. Bohnet (2003)), most automatically converted dependency treebanks have been converted using head-finding heuristics. These heuristics implement a set of descriptions how a (typical) head for a given phrase should look like (in terms of part-of-speech for example). The phrase-structure tree is then converted by choosing for every phrase one of the daughters as the head of the phrase.

German has two big syntactic treebanks, the TiGer treebank (Brants et al., 2002), and the TüBa-D/Z (Hinrichs et al., 2004). Both treebanks annotate phrase structure with modifications to account for German-specific phenomena. For example, because of the relatively free word order in German, all edges are labeled by functional labels specifying the function that the phrase bears to its mother node. Both treebanks have also been converted to dependency format several times (Daum et al., 2004; Versley, 2005; Hajič et al., 2009). While the versions by Daum et al. (2004) and Versley (2005) change the label set of the treebanks to comply with the grammar used by the parser by Foth et al. (2004), the data set derived from the TiGer treebank that was used in the CoNLL 2009 Shared Task (Hajič et al., 2009) stayed faithful to the original annotations. However, none of the German conversions pays special attention to the problem of ellipses, or rather to the problem of phrases where head-finding heuristics do not find a head.

To represent empty heads in dependency treebanks, three solutions have been proposed: All the German dependency treebanks that we know simply choose one of the dependents of an unheaded phrase to become the head. This is a rather undesirable solution because it ignores the fact that there is something missing in the structure. In the Czech treebank (Böhmová et al., 2001), which is a genuine dependency treebank that was not derived from a phrase-structure treebank, a similar solution as for the German treebanks is used: the dependents of an empty node are attached to the head of the empty node but the thus suspended dependents

are marked by a different label (*ExD*). This is a cleaner solution since the information about ellipsis is still annotated. However, this annotation deletes any functional information by replacing the function labels with a uniform ellipsis marker, but more importantly, it still hides the structural fact that there is a node missing in the tree. The Hungarian dependency treebank (Vincze et al., 2010) finally explicitly represents the empty nodes in the trees by introducing phonetically empty elements that serve as attachment points to other tokens. This is the cleanest solution from a linguistic point of view, which we will adopt here as well, but it poses problems when parsing with data-driven dependency parsers, since most of today's statistical dependency parsers are not capable of handling empty nodes. Empty nodes create the problem that the number of nodes that the parser has to connect in order to arrive at a dependency structure is no longer determined by the number of tokens in the sentence. This is however one of the fundamental assumptions in dependency parsing, and the algorithms are built upon this.
Recently, a parser has been proposed by Dukes and Habash (2011) that extends the transition-based paradigm for dependency parsing by adding an additional move to the parser that introduces empty nodes into the tree. As far as we know, this is the only published dependency parser so far that can handle empty nodes directly during the parsing process. Unfortunately, Dukes and Habash (2011) do not provide evaluation on the performance on empty nodes.

## 3. Converting the Treebank

The TiGer treebank (Brants et al., 2002) that we use to derive the dependency corpus is a phrase-structure treebank that annotates discontinuous constituents and classifies every edge into functional categories. This is done to account for the relatively free word order of German. Release 2.1 from August 2006, on which this work is based, consists of 50,474 sentences of newspaper text from the Frankfurter Rundschau comprising 888,238 tokens, and it is annotated with lemma, part-of-speech, and morphological information on the token level. The part-of-speech annotation uses the Stuttgart-Tübingen Tag Set (STTS) (Schiller et al., 1999).
The standard conversion procedures derive a dependency tree by choosing one of the daughters of the current phrase as the head of the phrase. The choice is made by using head-finding heuristics that usually contain a part-of-speech specification and search direction (start from left/right) for each phrase category in the treebank. For converting the TiGer treebank, we can in addition use the function labels attached to each node to further specify the head. For some phrase categories, the function labels allow us to select the head without any uncertainty since the TiGer annotation scheme uses a special label *hd* to mark the head of a phrase.
Table 1 shows the head-finding rules that we use to derive the dependency corpus. The first column gives the phrase category for which we want to select the head, the second column gives the function label preference for the head, the third column gives the part-of-speech tag preference of the head, and the last column states from which direction we should search through the daughters. The head selection algorithm then loops over the label and the part-of-speech lists and, for each label – part-of-speech combination, searches through the daughters. As soon as it finds a daughter with the specified label and part-of-speech tag, it selects this daughter as the head of the whole phrase.
Note that for the categories that are marking their head with a special label (e. g. the *hd* label), the part-of-speech of the head and also the search directions are not taken into account. In those cases, we rely on the annotation of the corpus to determine the correct head.

| phrase | label | POS | dir |
|---|---|---|---|
| s | hd | * | * |
| vp | hd | * | * |
| vz | hd | * | * |
| avp | hd, ph | * | * |
| ap | hd, ph | * | * |
| dl | dh | * | * |
| aa | hd | * | * |
| isu | uc | * | * |
| pn | pnc | ne, nn, fm, trunc, appr, apprart, card, vvfin, vafin, adja, adjd, xy, <empty> | right |
| nm | nmc | nn, card | right |
| mta | adc | adja | right |
| avp | avc | adv | right |
| pp | ac, ph | apprart, appr, appo, proav, ne, apzr, pwav, trunc | right |
| pp | nk | proav | left |
| avp | avc | fm | left |
| ch | uc | fm, ne, xy, card | left |
| np | nk, ph | nn | left |
| np | nk, ph | ne, pper, pis, pds, prels, prf, pws, pposs, fm, trunc, adja, card, piat, pwav, proav, adjd, adv, apprart, pdat, <empty> | right |

Table 1: Head-finding rules for the dependency conversion. First column gives the phrase category for which the head is selected, second column gives the label preference of the head, third column gives the part-of-speech tag preference, and last column gives the search direction in which to search through the daughters of the phrase (left: start with left-most daughter, right: start with right-most daughter)

For some phrase categories, we have more than one rule. The rules are applied in the order that is shown in Table 1 which means that we can use more than one rule for a particular category and the first rule that finds a suitable dependent will win. Take for example the *pp*-rules (prepositional phrases): the first rule defines the general head selection that most of the time will select a preposition labeled with *ac*. The second rule is used for some rare cases where the preposition and the argument of the preposition have merged into one word. These words are called pronominal adverbs (proav) in the STTS. An example is the word *dafür* (for that). If these words are unmodified, they are not dominated by a *pp*-phrase node in the original TiGer corpus. However, they can be modified like prepositions with adverbs (e. g. *auch dafür* (also for that)), which will then

create prepositional phrases that do not contain a preposition because it is merged into the pronominal adverb. The second *pp*-rule catches these cases.

### 3.1. Special Cases

In addition to the rules in Table 1, the conversion procedure treats some phenomena in a special way to produce a structure that is a) linguistically more plausible and b) better suited for training statistical dependency parsers.

**Structured Prepositional Phrases.** The original annotation of prepositional phrases in the TiGer corpus does not introduce a noun phrase thus producing very flat structures. The CoNLL 2009 Shared Task data set adopted this scheme which leads to dependency structures like the upper structure in Figure 2. In these structures, all dependents of the noun are attached to the preposition which leads to different structures for noun phrases, namely noun phrases that are embedded under a preposition, and noun phrases that are not. As was done in the PaGe Shared Task data set (Kübler, 2008), we change the annotation to represent the noun phrase inside the prepositional phrase. By introducing additional structure into the annotation, as is shown in the bottom structure of Figure 2, we unify these two noun phrase annotations, which is desirable from a linguistic point of view, because then all noun phrases are annotated in the same way. It is also beneficial to statistical parsing, as is shown in Seeker et al. (2010), because the statistical model can treat both types of noun phrase structures the same way regardless of whether they are embedded under a preposition or not.

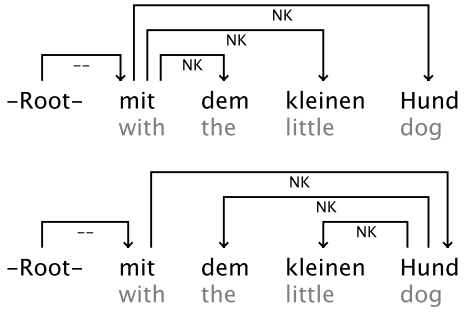



Figure 2: A prepositional phrase in the CoNLL 2009 data (top) and in our version (bottom).

The actual conversion happens by applying the *np*-head rule to the daughters of a prepositional phrase in order to choose the head of the embedded noun phrase. After that, all dependents that are not marked by the labels *ac, mo, ph, cm* are attached to the selected noun phrase head. The other dependents include either the head of the prepositional phrase or modifiers of it, which we do not want to attach to the noun phrase head. The normal conversion procedure is then continued.

**Coordination.** There are in principle three possibilities, how coordination can be annotated in dependency structures: one can make the coordinating conjunction the head of the structure and attach everything to it, one can make the first conjunct the head and attach everything else there, or one can make the first conjunct the head and then chaining conjuncts and coordinating conjunctions. Figure 3 shows schematics of the three possibilities. The first option was used for example in the Prague Dependency Treebank of Czech (Böhmová et al., 2001). From a linguistic point of view, this is the most informative scheme, because it is the only scheme that allows a distinction between dependents that affect the entire coordination, and dependents that affect only a particular conjunct. However, it is also a problematic scheme, if no coordinating conjunction is present. In these cases, the Czech corpus uses punctuation as the head of the coordination, which however restricts this annotation scheme to edited text. The CoNLL 2009 Shared Task data set for German adopted the second strategy where everything is attached to the first conjunct. This avoids the problem when there is no coordinating conjunction present. However, Nilsson et al. (2006) present experimental results on the Prague Dependency Treebank that the third option is best suited to statistical dependency parsing. Our conversion tool therefore annotates coordination as a chain starting with the first conjunct. There is one exception to that rule: if the first conjunct turns out to be a truncated element, as *Bus-* in *Bus- und Bahnverkehr* (roughly: bus and train traffic), then the first non-truncated element is made the head of the coordination.

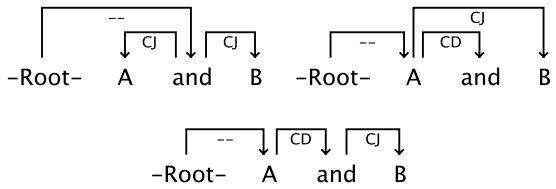



Figure 3: Three different annotation options for coordination structures.

**Noun Phrases with Multiple Nouns.** Table 1 shows two rules for noun phrases (np), the first of which selects the left-most noun as the head, while the second searches for a head from the right if no noun was found. In the prototypical noun phrase of German, the head is on the right of the phrase, and we could just always select the right-most noun as well. However, in certain cases the TiGer corpus has noun phrases that contain more than one noun. This happens for measurement constructions and close apposition as shown in Example 1 and 2 respectively. Because the original TiGer does not introduce unary branchings, the second noun *Tee/Arbeitslosigkeit* is annotated as sister to the first noun *Tasse/Thema*. In most of these cases, the other dependents morphologically agree with the first noun,[3] which motivates our choice to always select the left-most of several nouns to be the head of the whole phrase. For all other cases where the noun phrase does not contain a noun, we apply the second rule to find the head (e. g. proper nouns, pronouns etc.) on the right of the phrase.

[3]These constructions are much more complex than we can discuss here. There are e. g. some exceptions to the agreement phenomenon, often due to diachronic reinterpretation. See e. g. Spranger (2005) for an overview of cases that occur in German.

(1) *eine* *große* *Tasse* *Tee*
fem.sg fem.sg fem.sg masc.sg
a big cup of tea

(2) *das* *alte* *Thema* *Arbeitslosigkeit*
neut.sg neut.sg neut.sg fem.sg
the old topic unemployment

## 4. Marking Verbal Ellipses by Introducing Empty Nodes

In order to find elliptical constructions, we exploit the fact that the rules that we use to derive the dependency structure do not cover every eventuality (Table 1). The rule set contains one rule for verb phrases (vp) and sentences (s). The differences between these two categories in the TiGer corpus is that sentences are headed by finite verbs and verb phrases are headed by non-finite verbs. Our conversion tool only introduces empty nodes for these two phrase categories and each empty node does have at least one dependent, i. e. the empty nodes are always structural heads in the dependency structure. Since we annotate empty nodes automatically, this is the safest way to avoid the incorrect introduction of superfluous empty nodes.

The actual introduction of the empty nodes is an easy process. Before the actual conversion, we check all verb phrases and sentences in the phrase-structure corpus if they contain a daughter marked with the *hd* label (see top two rules in Table 1). If we do not find any such daughter, we attach an empty node as head daughter. A more difficult problem is then to position the empty node in the linear string of the sentence. From a linguistic perspective, this might not be a necessary measure to be taken since the dependency structure does not change with respect to the position of the empty node although a reasonable position might better support corpus search. However, from a dependency parsing perspective, the position of the empty node has a big impact on the projectivity of the dependency structure. If we placed e. g. all empty nodes at the end of a sentence, it would introduce a lot of non-projective edges, i. e. crossing edges, which would make it harder for most statistical dependency parsing algorithms to predict the structure.[4] Since the empty nodes function as phonetically empty verbal heads, we therefore apply two heuristics in order to position the empty nodes in the places where the overt verb would go as well. In verb phrases, the empty node is positioned as the last daughter but before any trailing sentential or punctuation dependents. This is in accordance to the verb final nature of German. We exclude sentential dependents because German tends to extrapose sentential dependents to a position after the verb. For sentence phrases (s), the heuristic is a bit more difficult: the s-category in the TiGer corpus marks sentences headed by finite verbs. This targets all matrix clauses but also relative clauses and a number of subordinate clauses. In German, the former type has the finite verb in the second position preceded by exactly one constituent, whereas the latter two have the verb at the end. So the heuristic positions the empty node at the end of the phrase (but again before any punctuation or sentential dependents), if the sentence node is labeled with *rc* (relative clause), or if it has a dependent marked by *cp* (complementizer) or *cm* (comparative marker). In all other cases, the empty node is placed as the second daughter. These heuristics approximate the verb position of German for the empty nodes.


*'Denmark's second biggest insurance company in danger.'*

Figure 4: Example of a headline (TiGer no. 1,275)

Having the empty nodes in place, we now give some statistics about the corpus. The current corpus contains 3,595 empty nodes, which gives 3,035 sentences that contain at least one empty node. 704 of these sentences do not contain any verb at all, 962 do not contain any finite verb. Most of these are head lines of articles, as shown in Figure 4. Table 2 shows the edge labels with which the empty nodes are labeled.

| label | freq | explanation |
|---|---|---|
| – – | 1444 | root |
| cj | 1046 | coordination |
| par | 417 | parenthesis |
| oc | 365 | clausal object |
| mo | 129 | modifier |
| rc | 42 | relative clause |
| re | 36 | repeated element |
| nk | 30 | noun kernel element |
| cc | 27 | comparative constr. |
| sb | 20 | subject |
| mnr | 13 | noun modifier |
| rs | 13 | reported speech |
| pd | 8 | predicate |
| app | 4 | apposition |
| da | 1 | dative |

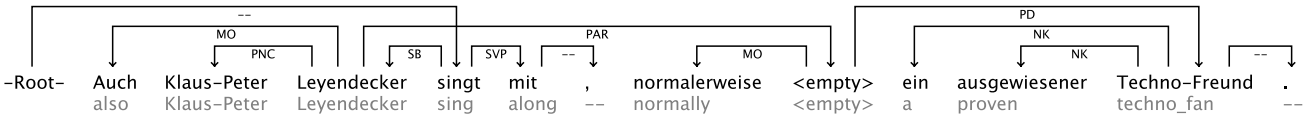
Table 2: Frequency of edge labels that are marking empty nodes

The most frequent case is the empty node being the root node of the whole sentence, which is the case e. g. for all the head lines. The second most frequent case is that the empty node heads a conjunct in a coordination. We show an example for that in Figure 1. Figure 5 gives an example for an empty node heading a parenthesis, and an example for empty nodes heading a clausal object is again shown in Figure 1. These four edge labels constitute 91% of the empty nodes in the corpus.

### 4.1. Resorting to Canonical Dependency Structures

At the moment, there is only one statistical parser that we know of that is capable of handling empty elements directly during parsing, which is the parser described by Dukes and

[4] Admittedly, as far as we know, there is only one statistical dependency parser so far that is actually capable of handling empty nodes directly, namely the parser by Dukes and Habash (2011). Nonetheless we think it is worth producing a structure as structurally simple and canonical as possible.




*'Even Klaus-Peter Leyendecker sings along, normally a proven techno fan.'*

Figure 5: Example of a parenthesis (PAR) headed by an empty node (TiGer no. 18,574)

Habash (2011). Their parser is a shift-reduce parser (Nivre et al., 2004) that has an additional move that allows it to introduce empty nodes into the structure. Unfortunately, they do not provide an evaluation of their parser specifically targeting the empty node performance. However, since there are no other such parsers at the moment, we also implemented the standard fallback routine that is usually used to deal with empty heads in dependency corpora: one of the daughters is chosen to become the head of the phrase. This does not lead to linguistically plausible structures but it allows statistical dependency parsers to process these trees. For the sake of completeness, we show the list of part-of-speech tags in Figure 6 that we consider from left to right when substituting the empty node with one of its daughters. This list is not very much motivated, especially at the end, except that it should cover all cases that occur in the treebank.

vvfin, vafin, vmfin, nn, ne, apprart, appr, appo, ptkneg, vvpp, vapp, adjd, proav, pper, vvinf, vainf, vminf, adja, card, kous, pwav, ptkant, pis, pds, pws, adv, trunc, fm

Figure 6: List of part-of-speech tags that can be substituted for an empty node, searching daughters from left to right.

## 5. Error Correction and Manual Head Selection

The rules shown in Table 1 and the introduction of empty nodes alone will not fully convert all trees in the TiGer treebank because there are errors in the annotation that block the application of an otherwise applicable rule. As we stated in the previous section, we deliberately did not design the rules to cover every possible situation in the corpus in order to find those constructions which genuinely lack something. Although this enables us to find verb ellipses, it also means that annotation errors will lead to partially derived trees. E. g. there are prepositional phrases that do contain a preposition but this preposition is not labeled by *ac* or *ph* (see first rule for pp). We therefore spent some time to correct several types of errors in the original treebank in order to arrive at a more consistently annotated data set. We will describe briefly, what we did, but please note that there are of course still errors left in the annotation that we did not find. Note also that the decision if something is an error was made by the authors by consulting the annotation guidelines and doing manual consistency checks.

All error corrections that we apply are integrated into the conversion tool in terms of modification statements. When the tool converts the TiGer treebank, it first applies the modifications to the original corpus before it starts converting the trees to dependency format. This way, anyone can download the standard TiGer release and apply our tool without needing access to our corrected version. Note that this also means that this tool only works with TiGer 2.1 Release August 2006.

The errors we corrected can be classified by whether they affect part-of-speech tags or phrase categories, morphology, function labels, or the syntactic structure. Since these errors interact sometimes, we give some orientation figures in Table 3 on the atomic changes that the conversion tool makes to the treebank before the conversion.

| | |
|---|---|
| part-of-speech tags and phrase category labels | 1,642 |
| edge labels | 681 |
| morphological annotation | 305 |
| reattachment of edges | 291 |
| additional nodes | 80 |
| deleted nodes | 41 |

Table 3: Number of atomic changes that the tool makes in order to correct annotation errors.

To find errors in the treebank, we used an idea by Bouma (2010). We loaded the whole corpus in Prolog by representing every node in the corpus as a single Prolog statement. Thus we can use Prolog as a querying language to query the corpus. We looked for patterns that would violate the annotation guidelines. For example, we looked for verb phrases that contained words that were tagged with verbal part-of-speech tags but did not contain a daughter with an *hd* label (which marks the head of a verb phrase) and vice versa. Another pattern that uses part-of-speech tags would be to look for prepositional phrases that do not contain a preposition. One can also crosscompare edge labels and morphology, e. g. we looked for words that had edge label *sb* (subject) but were not marked for nominative case. The general idea is to come up with patterns that you would not expect to occur either because of a linguistic insight into the language or because of the annotation guidelines. Manning (2011) uses this approach to find and correct errors in the Penn-Treebank part-of-speech annotation. When we found these patterns, we manually checked the results and added a modification statement in case it turned out to be an annotation error.

With error correction applied, we can convert almost the complete corpus. However, a small number of cases still remained that could not be derived but are not annotation errors either. In addition to that, we found some phenomena, where we found it easier to treat them manually instead of creating very complex derivation rules. For these, we manually selected the head. One such case is shown in Example 3. It is a company name that ends on the abbreviation *GmbH* which states the legal status of the company (limited liability company). Linguistically, the complete expression is used like a compound noun, and agreement

features indicate that the head of the whole expression is the last noun. Since this breaks our noun phrase head selection rule, which would select *Club* (Robinson is tagged as proper noun) as the head, we manually override this by marking *GmbH* as the correct head.

(3) *die* *Robinson* *Club* *GmbH*
fem.sg masc.sg fem.sg
the Robinson Club GmbH

## 6. Quality Evaluation for Dependency Parsing

In order to evaluate the quality of the annotation scheme, we train two data-driven parsers, one for each of the major dependency parsing paradigms. For the graph-based paradigm (McDonald et al., 2005), we use the parser described in Bohnet (2010),[5] and for the transition-based paradigm (Nivre et al., 2004), we used the transition-based parser described in Bohnet (2011).

As our baseline, we use the original CoNLL 2009 Shared Task data set for German (Hajič et al., 2009) because it uses the same edge label set. Both data sets have been derived from the TiGer corpus. However, because the CoNLL data were derived from the SALSA Corpus (Burchardt et al., 2006), it is based on an older release of TiGer. We therefore changed our version of the corpus manually in order to have the exact same sentences in training, development, and test data for this evaluation. In order to provide an equal preprocessing for both data sets, they were annotated by the same tools for lemmata, part-of-speech, and morphological information using a ten-fold cross-annotation on the training set. The development and the test data were then annotated with models trained on the training data.

Table 4 shows the labeled and unlabeled attachment scores (no punctuation) for both parsers when trained and tested on each corpus version.

| | graph | | trans | |
|---|---|---|---|---|
| | LAS | UAS | LAS | UAS |
| CoNLL '09 dev | 87.71 | 90.40 | 86.48 | 89.48 |
| this work dev | 89.84 | 92.24 | 88.17 | 90.76 |
| CoNLL '09 test | 88.50 | 90.96 | 87.31 | 89.96 |
| this work test | 90.30 | 92.56 | 88.68 | 91.15 |

Table 4: LAS (labeled attachment score) and UAS (unlabeled attachment score) on both corpora (no punctuation)

Since both corpora use different annotation schemes and thus different gold standards, a direct comparison of the parsing performance will not tell us if one of the corpora is better than the other. We therefore use the tedeval procedure[6] presented in Tsarfaty et al. (2011) that computes a generalized gold standard from both corpora and evaluates both parsers against this shared gold standard using a tree edit distance measure (TED). This generalized gold standard only contains information that is present in both single gold standards, meaning that both parsers are then compared against the same structures. If a parser performs better on the generalized gold standard, it means that it is better modeling the linguistic content that is shared between both annotation schemes. Since the tedeval software currently only supports projective trees, we projected all our trees before the experiment. We also excluded all punctuation during the evaluation.[7]

| | graph | trans |
|---|---|---|
| CoNLL '09 dev | 94.15 | 93.98 |
| this work dev | 95.21 | 94.21 |
| CoNLL '09 test | 94.66 | 94.24 |
| this work test | 95.25 | 94.64 |

Table 5: Labeled micro TED scores (no punctuation). Pairwise differences are statistically significant using the shuffling test provided with the tedeval software.

Table 5 shows the labeled micro TED scores when comparing the parsers on both corpora against a common gold standard. All pairwise differences are statistically significant indicating that our conversion is better suited to train parsing models with the two parsers that we use in the experiment. Note also that the differences between the two corpora are much smaller than the differences in Table 4. This effect is in line with the findings in the experiments in Tsarfaty et al. (2011) demonstrating that direct comparison of parser performance on different annotation schemes can be easily misleading.

## 7. Conclusions and Future Work

We presented a tool to derive a dependency version from the German TiGer treebank that explicitly introduces empty nodes to represent verb ellipses. The derivation process uses head-finding heuristics to determine the head of a phrase but, different to previous work, these heuristics do not trade precision for coverage. Instead, the fact that sometimes no head can be found for a phrase is used to detect elliptical constructions, which are then annotated explicitly using phonetically empty elements. The tool can also derive a canonical version of the corpus without empty nodes, which is shown by experiment to be better suited for training statistical dependency parsers than the data used in the CoNLL 2009 Shared Task. In the future, we would like to spend some effort into manually annotating the kind of verbal ellipses that we found in the corpus. The tool will be made freely available for academic purposes on the authors' webpages.

## Acknowledgements

This work was funded by the Deutsche Forschungsgemeinschaft (DFG) via Project D8 of the SFB 732 "Incremental Specification in Context". We would also like to thank Dr. Bernd Bohnet for providing his parsers and Anders Björkelund for providing his projection software.

## 8. References

A. Böhmová, Jan Hajič, Eva Hajičová, and B. Hladká. 2001. The prague dependency treebank: Three-level

[5] downloadable from code.google.com/p/mate-tools, you can also find the preprocessing tools here

[6] http://stp.lingfil.uu.se/ tsarfaty/unipar/index.html

[7] We count all tokens as punctuation whose gold standard part-of-speech tag starts with $.

annotation scenario. In A. Abeillé, editor, *Treebanks: Building and using syntactically annotated corpora. Kluwer Academic Publishers*, chapter 1, pages 103–127. Kluwer Academic Publishers, Amsterdam.

Bernd Bohnet. 2003. Mapping Phrase Structures to Dependency Structures in the Case of ( Partially ) Free Word Order Languages. In *International Conference on Meaning-Text Theory*, pages 239–249, Paris, France.

Bernd Bohnet. 2010. Very high accuracy and fast dependency parsing is not a contradiction. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 89–97, Beijing, China. Association for Computational Linguistics.

Bernd Bohnet. 2011. Comparing Advanced Graph-based and Transition-based Dependency. In *Proceedings of Depling 2011*, pages 282–289, Barcelona, Spain.

Gerlof Bouma. 2010. Syntactic tree queries in Prolog. In *Proceedings of the Fourth Linguistic Annotation Workshop, ACL 2010*, pages 212–216, Uppsala, Sweden. Association for Computational Linguistics.

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

Aljoscha Burchardt, Katrin Erk, Anette Frank, Andrea Kowalski, Sebastian Padó, and Manfred Pinkal. 2006. The SALSA corpus: a German corpus resource for lexical semantics. In *LREC 2006*, pages 969–974, Genoa, Italy.

Michael Daum, Kilian Foth, and Wolfgang Menzel. 2004. Automatic transformation of phrase treebanks to dependency trees. In *Proc. 4th Int. Conf. on Language Resources*.

Kais Dukes and Nizar Habash. 2011. One-Step Statistical Parsing of Hybrid Dependency-Constituency Syntactic Representations. In *International Conference on Parsing Technology (IWPT 2011)*, pages 92–104, Dublin, Ireland. Association for Computational Linguistics.

Kilian Foth, Michael Daum, and Wolfgang Menzel. 2004. A broad-coverage parser for German based on defeasible constraints. In *Proceedings Constraint Solving and Language Processing*.

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Stepánek, Pavel Stranák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang. 2009. The CoNLL-2009 shared task: Syntactic and Semantic dependencies in multiple languages. In *Proceedings of the 13th CoNLL Shared Task*, pages 1–18, Boulder, Colorado.

Erhard Hinrichs, Sandra Kübler, Karin Naumann, Heike Telljohann, and Julia Trushkina. 2004. Recent developments in linguistic annotations of the TüBa-D/Z treebank. In *Proceedings of the Third Workshop on Treebanks and Linguistic Theories*, pages 51–62, Tübingen, Germany.

Richard Johansson and Pierre Nugues. 2007. Extended constituent-to-dependency conversion for English. In *Proc. of the 16th Nordic Conference on Computational Linguistics (NODALIDA)*, pages 105–112. Citeseer.

Sandra Kübler. 2008. The PaGe 2008 shared task on parsing German. In *Proceedings of the Workshop on Parsing German*, pages 55–63, Morristown, NJ, USA. Association for Computational Linguistics.

David M. Magerman. 1994. *Natural language parsing as statistical pattern recognition*. Ph.D. thesis, Stanford.

Christopher D. Manning. 2011. Part-of-speech tagging from 97% to 100%: is it time for some linguistics? In *Computational Linguistics and Intelligent Text Processing*, pages 171–189. Springer.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005. Non-projective dependency parsing using spanning tree algorithms. *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing - HLT '05*, pages 523–530.

Igor Mel'čuk. 1988. *Dependency Syntax: Theory and Practice*. State University Press of New York, suny serie edition.

Jens Nilsson, Joakim Nivre, and Johan Hall. 2006. Graph transformations in data-driven dependency parsing. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL - ACL '06*, pages 257–264, Morristown, NJ, USA. Association for Computational Linguistics.

Joakim Nivre, Johan Hall, and Jens Nilsson. 2004. Memory-based dependency parsing. In *Proceedings of CoNLL 2004*, pages 49–56, Boston, Massachusetts.

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

Wolfgang Seeker, Bernd Bohnet, Lilja Ø vrelid, and Jonas Kuhn. 2010. Informed ways of improving data-driven dependency parsing for German. In *Proceedings of Coling 2010*, Beijing.

Kristina Spranger. 2005. Some remarks on the Annotation of Quantifying Noun Groups in Treebanks. In *Proceedings of the 6th International Workshop on Linguistically Interpreted Corpora (LINC-2005)*, pages 81–90.

Reut Tsarfaty, Joakim Nivre, and Evelina Andersson. 2011. Evaluating Dependency Parsing: Robust and Heuristics-Free Cross-Annotation Evaluation. In *Empirical Methods of Natural Language Processing (EMNLP)*, Edinburgh.

Yannick Versley. 2005. Parser evaluation across text types. In *Proceedings of the Fourth Workshop on Treebanks and Linguistic Theories (TLT 2005)*.

Veronika Vincze, Dóra Szauter, Attila Almási, György Móra, Zoltán Alexin, and János Csirik. 2010. Hungarian Dependency Treebank. In *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC 2010)*, pages 1855–1862, Valletta, Malta.

Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical Dependency Analysis with Support Vector Machines. In *Proceedings of IWPT*, pages 195–206.