On Representing Dependency Relations – Insights from Converting the German TiGerDB

Adriane Boyd
Ohio State University
Dept. of Linguistics

Markus Dickinson
Indiana University
Dept. of Linguistics

Detmar Meurers
Ohio State University
Dept. of Linguistics

Abstract

Research in parser evaluation has led to the creation of dependency resources such as the TiGer Dependency Bank, a semi-automatic conversion of a subset of the TIGER Treebank. We explore the relationship between the TiGerDB representation and a more surface-oriented dependency analysis of German and describe how we mapped and recoded the TiGerDB into a format more closely linked to the original treebank data.

1 Introduction

Current research on parser evaluation across corpus annotation schemes has faced significant difficulties in finding comparable data (cf, e.g., Rehbein and van Genabith 2007, and references therein). The TiGer Dependency Bank (TiGerDB, Forst et al. 2004) is a valuable step in addressing this issue, in that it provides a dependency analysis based on sentences drawn from the newspaper corpus used in the TIGER Treebank, itself a combination of phrase structure and functional annotation. More than that, the goal of the TiGerDB is to be a theory-neutral dependency-based gold standard for evaluating German parsers. While it is related to the functional annotation in the TIGER Treebank, extensive manual and automatic conversion steps, such as disambiguating intermediate LFG f-structure representations using a broad-coverage LFG parser (Dipper 2003), were involved in developing a dependency annotation with that goal in mind. In sum, the TiGerDB as a dependency annotation of the TIGER sentences clearly is attractive both for serving as a target for evaluating parsers trained on a range of corpus annotation schemes and as a dependency representation in its own right, given the increased interest in dependency grammar and direct dependency parsing.

At the same time, there are aspects of the TiGerDB which arguably need to be revisited if it is to serve as a gold standard dependency resource. Its annotation is not directly aligned with the actual corpus tokens; it includes abstract nodes for analyzing coordination; and it introduces sublexical nodes for compounds, comparatives and superlatives, and contractions. Such nodes are not utilized in most

De Smedt, K., Hajič, J. and Kübler, S. (Eds.)
Proceedings of the Sixth International Workshop
on Treebanks and Linguistic Theories (2007)

dependency parsers (e.g., McDonald and Pereira 2006; Nivre and Hall 2005) or previous dependency banks. Finally, the TiGerDB includes dependencies with multiple heads, which most dependency encoding schemes do not allow.

This paper explores the relationship between the TiGerDB representation of German and a more traditional lexical dependency analysis and describes how we mapped and recoded the TiGerDB into a format which preserves the original distinctions, but more readily provides access to the key lexical information for data-driven parsing approaches. This also makes the dependency annotation more directly comparable to other dependency analyses of German, such as that of Foth (2006).

To support this process on the technical side, we define a treebank format in the form of an XML Schema Definition that supports the full range of dependency representations. On the one hand, this format is motivated by the need to represent the TiGerDB and all stages of the conversion process; on the other, we hope that it proves useful as a basis for representing and comparing a wider range of dependency representations.

2 TiGerDB and the desiderata to be addressed

The TiGerDB was developed based on the TIGER Treebank (Brants et al. 2002), a corpus text taken from the national German newspaper *Frankfurter Rundschau*. Its dependency annotation is based on the annotation scheme developed for the English PARC 700 Dependency Bank (King et al. 2003) and is designed to encode the "distinctions current deep parsers of German make" (Forst et al. 2004).

The dependency annotation scheme, aiming at a theory-neutral gold standard, differs significantly from the grammatical relations encoded in the original TIGER Treebank. Of the 53 types of dependency relations used, 11 are completely new, and 5 more have slightly different definitions. For example, the TIGER label *MO* (modifier) is split into *mo* (optional modifier) and *pd* (predicative argument) since these are distinctions often made by German parsers. In addition, the TIGER label *AMS* (measure argument/adjunct) is merged into the TiGerDB label *mo*.

Consequently, there is no direct mapping from TIGER to the TiGerDB. Creating the TiGerDB required significant manual work, aimed at obtaining a high-quality annotation reflecting the relevant distinctions for a gold standard parsing evaluation resource. At the same time, the current TiGerDB arguably is not in a format that can be readily used (cf., also Versley and Zinsmeister 2006). TiGerDB does not directly encode the original lexical items, using lemmas or abstract forms instead. For uses where the surface string is needed, the TiGerDB token numbers have to be used to align each dependency from TiGerDB with the surface string from the original TIGER Treebank. This alignment is complicated in several ways.

First, and most prominently, a clear mapping between the nodes in the TiGerDB and the words in TIGER does not always exist. On the one hand, in the TiGerDB, lemmas and abstract nodes with or without lexical counterparts are the elements

ISSN 1736-6305 Vol. 1
http://dspace.utlib.ee/dspace
/handle/10062/4476

involved in dependency relations. Instead of annotating conjuncts, coordination is annotated through the use of abstract coordination nodes, and subword nodes are used to annotate non-head elements of compounds, comparative and superlative adjectives, and contractions. On the other hand, some surface string tokens are not annotated as part of word-word dependencies at all: auxiliary verbs are annotated as tense and mood features and are thus left out of the dependency graph, and separable prefixes and the right parts of circumpositions also receive no annotation. Additionally, some multiword expressions are tokenized differently in the two resources.

Second, it can be difficult to align nodes with tokens due to simple inconsistencies. The TiGerDB token IDs pointing to the corresponding forms in TIGER contains errors, and thus one has to manually check and align cases.

Let us take a look at an example annotation from the TiGerDB to illustrate the basic setup and the features described above. The annotation of sentence (1) is shown right below it.

(1) Das gehört offenbar zum Spiel.
that belongs apparently to the game
'That apparently belongs to the game.'

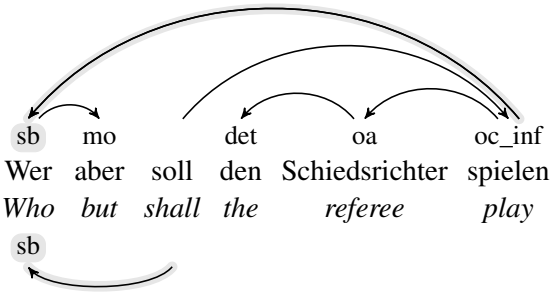
```
mo(gehören~0, offenbar~3)
mood(gehören~0, ind)
op(gehören~0, zu~4)
sb(gehören~0, pro~1)
tense(gehören~0, pres)
tiger_id(gehören~0, 2)
case(pro~1, nom)
gend(pro~1, neut)
num(pro~1, sg)
pers(pro~1, 3)
pron_type(pro~1, demon)
degree(offenbar~3, pos)
obj(zu~4, Spiel~5)
case(Spiel~5, dat)
det(Spiel~5, die~1012597)
gend(Spiel~5, neut)
num(Spiel~5, sg)
det_type(die~1012597, def)
```

The annotation consists of two types of expressions: dependency relations such as the first term establishing a `mo` relation between `gehören` ('to belong') and `offenbar` ('apparently'), and features specifications, such as the second to last term declaring the `num` of `Spiel` ('game') to be `sg`. Each of the terms of the TiGerDB annotation refers to a lemma or abstract node. For example, the term for the word *gehört* ('belongs') from the surface string refers to the lemma *gehören*, and the pronominal subject *das* ('that') appears as the special lemma `pro`, with `pron_type`, `case`, `num`, and `gend` specifying which kind of pronoun it is. The root *gehört* is given the ID 0 (notated after the ~), and the position of the token is encoded by the `tiger_id` feature, whereas usually the ID corresponds to the word's position in the sentence. Lastly, the definite article in the contraction *zum* (*zu dem*, 'to the') is annotated using the element `die` that does not have a one-to-one correspondence with any terminal in the sentence.

3 Representing dependencies: Malt-XML to Decca-XML

Before we can turn to the conversion process itself, we need to define a representation format that is flexible enough to encode the TiGerDB as well as a more traditional lexical dependency representation. Fortunately, an XML representation that satisfies many of the requirements has already been developed by Joakim Nivre and colleagues in the MALT project. We thus take the Malt-XML format (`http://w3.msi.vxu.se/~nivre/research/MaltXML.html`) as the starting point and focus on the necessary modifications.

One of the tenets of traditional dependency approaches that is relaxed in the TiGerDB is the single-head constraint (cf, e.g., Mel'čuk 1988, ch. 1). The TiGerDB annotation scheme in certain cases identifies a word as the dependent of two separate heads, for example, to explicitly encode all dependencies in control constructions. Sentence (2), translating as *'But who shall function as the referee?'*, illustrates this situation.



(2) Wer aber soll den Schiedsrichter spielen ?
Who but shall the referee play ?

In this sentence, the subject *wer* ('who') is analyzed as being dependent upon both the modal verb *soll* ('shall'), for which it is the syntactic subject, and the main verb *spielen* ('play'), for which it is a semantic argument. Thus, in order to represent the TiGerDB and, more generally, any dependency bank rejecting the single-head constraint, our encoding format must be flexible enough to encode multiple heads.

In the Malt-XML format, one can encode most of example (2), as in Figure 1.

```
<sentence id="8017">
  <word id="1" form="Wer" postag="PWS" head="6" deprel="sb" />
  <word id="2" form="aber" postag="ADV" head="1" deprel="mo" />
  <word id="3" form="soll" postag="VMFIN" head="0" deprel="root" />
  <word id="4" form="den" postag="ART" head="5" deprel="det"/>
  <word id="5" form="Schiedsrichter" postag="NN" head="6"
        deprel="oa" />
  <word id="6" form="spielen" postag="VVINF" head="3"
        deprel="oc_inf" />
  <word id="7" form="?" postag="$." />
</sentence>
```

Figure 1: Malt-XML encoding of most of example (2)

Each word is encoded as an element within a `<sentence>` structure. The dependencies are encoded in each dependent word using a `head` attribute that refers to

ISSN 1736-6305 Vol. 1
http://dspace.utlib.ee/dspace
/handle/10062/4476

the head via a unique ID introduced for each word. The deprel attribute encodes the nature of the dependency relation for each dependent.

The Malt-XML encoding in Figure 1 represents all dependency relations of (2) except for the subject relation between *soll* and *wer*, which cannot be directly encoded since multiple dependency heads cannot be expressed in Malt-XML.

For Decca-XML, we eliminate this restriction by replacing the head and deprel attributes with a new element type <head> as a child of <word>. The <head> element contains attributes for the id of the head and the type of relation (deprel). Figure 2 exemplifies how this makes it possible to encode the subject *wer* of example (2) as a dependent of two heads.

```
<sentence id="8017">
  <word id="1" form="Wer" postag="PWS">
    <head id="3" deprel="sb" />
    <head id="6" deprel="sb" />
  </word>
  ...
  <word id="3" form="soll" postag="VMFIN">
    <head id="0" deprel="root" />
  </word>
  ...
  <word id="6" form="spielen" postag="VVINF">
    <head id="3" deprel="oc_inf" />
  </word>
  ...
</sentence>
```

Figure 2: Decca-XML encoding of the multiple heads from example (2)

The second area where Decca-XML departs from the Malt-XML format arises from the use of abstract and sublexical nodes in the TiGerDB. We add two additional token types within the sentence, <abstract> and <subword>, to be used for tokens that do not directly correspond to terminals. For representing the TiGerDB, <abstract> is used for the coordination nodes and <subword> for the non-head elements of compound words, comparative/superlative adjectives, and contractions. Both <abstract> and <subword> can have multiple heads, just like <word>.

Finally, in order to transparently support referring back to the original annotation, Decca-XML includes an optional attribute externid for <sentence>, <word>, <abstract>, and <subword> elements that encodes the corresponding sentence or token ID from the original annotation.

4 The conversion process

The conversion process aligns the TiGerDB tokens with terminals in the TIGER Treebank and distinguishes the three types of tokens in the TiGerDB (word, abstract, subword) to create a graph of word-word dependencies for each sentence.

As we saw in section 2, the TiGerDB encodes a combination of binary relations between tokens (e.g., subject, accusative object) and features for individual tokens (e.g., case, number, tense). The focus here is on the dependency relations between pairs of tokens, with the features only being used to guide the conversion.[1]

4.1 Linguistic issues

4.1.1 Coordination

We start the discussion of linguistic issues arising in the conversion with two coordination examples. Example (3) shows an example including a conjunction (*and*), while example (4) does not include one.

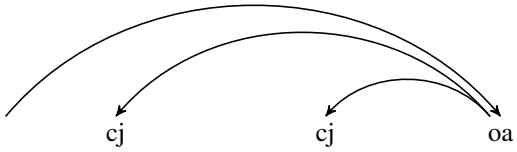
(3) Behörde und Begehrlichkeit
Authorities and Greed

```
cj(coord~0, Begehrlichkeit~3)     coord_form(coord~0, und)
cj(coord~0, Behörde~1)
```

(4) Siehe Kommentar S. 3, Berichte S. 4
See editorials p. 3, reports p. 4

```
cj(coord~521, Bericht~6)          oa(sehen~0, coord~521)
cj(coord~521, Kommentar~2)
```

As mentioned in section 2, in the TiGerDB coordination is analyzed with abstract `coord` tokens. If a conjunction is present, it is identified in the TiGerDB annotation by the feature `coord_form`, as exemplified by the third expression below (3). The relation between the `coord` token and its head encodes the function of the coordination as a whole in the sentence, e.g., in example (4), the accusative object (`oa`). The conjuncts are dependents of the `coord` token with the dependency relation `cj`, displayed graphically in (5).



(5) Siehe Kommentar S. 3, Berichte S. 4 `coord`

The use of abstract tokens allows all instances of coordination to be annotated in the same manner. The Decca-XML format can in principle represent the TiGerDB analysis using the `<abstract>` token type. In order to connect the dependency analysis to the TIGER Treebank terminals as directly as possible, however, we choose to use the conjunction as the head instead. In instances where there is no conjunction, there are several different approaches that could be taken.

[1]The `id` can be used to locate the morphological information in the TIGER Treebank, and the `externid` can be used to associate any additional TiGerDB features with a particular token.

ISSN 1736-6305 Vol. 1
http://dspace.utlib.ee/dspace
/handle/10062/4476

One option is to designate one of the conjuncts as the head, as illustrated in example (6). This is the approach used in for the Constraint Dependency Grammar analysis of German (Foth 2006). The first conjunct's dependency relation corresponds to the role of the entire phrase in the sentence and all other conjuncts are dependents below the first conjunct.

(6) Siehe Kommentar S. 3, Berichte S. 4

A second option is to use a special dependency label for conjuncts that includes information about the grammatical role and about the status of the phrase as a conjunct. Shown in example (7), the new label `oa_cj` is the dependency relation for both conjuncts. This is similar to the analysis of conjuncts in the Prague Dependency Treebank (Hajičová et al. 1998).

(7) Siehe Kommentar S. 3, Berichte S. 4

While the Decca-XML format could represent either of these, we choose to leave the abstract coordination as it stands when no conjunction is present. There is no TIGER Treebank terminal to associate with the abstract token, so the objective of aligning all TIGER terminals with nodes in the TiGerDB is satisfied.

For the most part, the conversion of coordination can proceed automatically. If a conjunction appears more than once in a sentence, the `coord_form` value in the TiGerDB does not provide enough information to determine which of the conjuncts to align with which coordination. In those cases, alignment is done by hand.

Finally, there is an additional TiGerDB feature that is relevant in this context. The feature `precoord_form` is used to identify any other token that should be associated with the conjunction, such as the first part of a composite coordinate form, as in *weder* ... *noch* ('neither ... nor'). In the conversion, we integrate these TIGER terminals into the annotation as dependencies by making them dependents of the conjunction with the new relation label `precoord`.

4.1.2 Auxiliaries

When used as auxiliaries, forms of the verbs *haben*, *sein*, and *werden* are not represented as tokens in the TiGerDB annotation. Instead, tense and mood features are used on the non-finite verb to indicate the contribution of the auxiliary. We choose to make the analysis of auxiliaries parallel to the analysis of modal verbs, such as the modal verb *soll* we saw in example (2), where the modal verb is the head of the non-finite verb and the subject of the clause is both the subject of the modal verb and of the non-finite verb. In the conversion, the auxiliary becomes the

head of the non-finite verb using the existing dependency relation `oc_inf` (infinite clausal object) and the subject of the non-finite verb becomes the subject of both the non-finite verb and the auxiliary. This generalizes the double head analysis already used in the TiGerDB (cf. section 3) to the case of auxiliaries (and could easily be mapped to another representation given arguments for another analysis).

4.2 Data representation issues

4.2.1 Basic alignment with terminals from TIGER Treebank

The TiGerDB token IDs and token forms (e.g., `coord`) are used to identify the word tokens, abstract tokens, and sublexical tokens that the annotation refers to. The word token IDs are used to align the TiGerDB ID with the token in the TIGER Treebank in order to retrieve the inflected word form, and the associated lemma and part-of-speech information is taken from the TIGER Treebank for all word tokens. The root of a TiGerDB dependency graph bears ID 0, so for lexical roots the feature `tiger_id` is consulted to determine the associated word position in the TIGER Treebank.

4.2.2 Indirectly annotated tokens

In the TiGerDB, a few types of words are indirectly included in the annotation but are not indexed to the corresponding TIGER Treebank terminals. These include separable prefixes appearing separate from the verb stem and the right-most element in a circumposition. Both can be identified by unique parts of speech in the TIGER Treebank, and the conversion associates them with the appropriate verb stem or the left-most element of the circumposition, respectively.

4.2.3 Alternate tokenization

The TiGerDB does not always preserve the tokenization from the TIGER Treebank. Certain proper names, such as *Den Haag* ('The Hague') and *British Telecom*, are two tokens in the TIGER Treebank but are combined into a single token in the TiGerDB. Other sequences, such as telephone numbers containing spaces, are treated as multiple tokens in the TIGER Treebank but combined in the TiGerDB. In order to align the TiGerDB annotation with the TIGER Treebank tokens, it is necessary to segment the combined tokens back into the TIGER Treebank tokens. In the conversion, proper names are analyzed using the existing dependency relation `name_mod`. Other sequences are encoded using the more general relation `mod` also used for subword elements in compound nouns, which we now turn to.

4.2.4 Subword tokens

Subword tokens in the TiGerDB include non-head elements of compounds, comparative and superlative inflection for adjectives, and contractions such as preposi-

ISSN 1736-6305 Vol. 1
http://dspace.utlib.ee/dspace
/handle/10062/4476

tion + definite article contractions (e.g., *in* + *dem* → *im*, 'in the'). Because the internal analysis of compounds is not present in the TIGER Treebank, in our conversion the elements representing the non-head elements of compounds are converted to <subword> tokens. All subword information from the TiGerDB is preserved in the conversion, even where the TIGER Treebank already includes part-of-speech tags and morphological information which encode the adjective inflection information and distinguish contractions. For the use of the corpus as a resource for word-word dependency parsing, subword tokens can simply be ignored.

4.2.5 Hand correction

Most of the conversion described above proceeds automatically. A few modifications must be completed by hand: some token ID numbering errors in the TiGerDB need to be corrected; when there are multiple occurrences of the same conjunction in a sentence, each conjunct need to be associated with the correct occurrence in the sentence; and all tokenization mismatches need to be analyzed individually. Including all types of tokens, the subsection of the TIGER Treebank annotated in the TiGerDB includes about 36,000 tokens in 1,867 sentences. Approximately 4% of the token IDs were corrected, 300 conjunctions aligned, and 170 multi-word expressions segmented.

4.3 Detailed examples

We illustrate the conversion process with two TiGerDB sentences. Example (8) shows a sentence with multiple heads due to the analysis of predicative adjectives along with the sublexical analysis of a compound noun. The relevant relations from the TiGerDB analysis are shown below the example.

(8) Die Personenführung ist exzellent.
the guiding of people is excellent
'The way people are guided is excellent.'

```
pd(sein~0, exzellent~4)          mod(Führung~2, Personen~2001)
sb(sein~0, Führung~2)            sb(exzellent~4, Führung~2)
tiger_id(sein~0, 3)
```

The verb *ist* ('is', lemma: *sein*) bears TiGerDB token ID 0 as the root of the sentence. The subword token *Personen* bears ID 2001. The remaining token IDs correspond to the original TIGER Treebank terminals. After locating the token position of the root using the feature `tiger_id`, the word tokens are aligned with the original sentence, and the lemmas and part-of-speech tags are extracted from the TIGER Treebank. The subword token *Personen* is identified by its token ID and its binary dependency relations is mapped. The external id pointers (`externid`) map each token back to the TiGerDB token ID so that additional features could be added as needed. The resulting Decca-XML output is shown in Figure 3.

```
<sentence id="9595" externid="tiger-db-9595.fdsc">
  <word id="1" form="Die" lemma="der" postag="ART" externid="1">
    <head id="2" deprel="det"/>
  </word>
  <word id="2" form="Personenführung" lemma="Personenführung"
   postag="NN" externid="2">
    <head id="3" deprel="sb"/>
    <head id="4" deprel="sb"/>
  </word>
  <word id="3" form="ist" lemma="sein" postag="VAFIN" externid="0">
    <head id="0" deprel="ROOT"/>
  </word>
  <word id="4" form="exzellent" lemma="exzellent" postag="ADJD"
   externid="4">
    <head id="3" deprel="pd"/>
  </word>
  <word id="5" form="." lemma="--" postag="$." externid="">
  </word>
  <subword id="2001" form="Personen" externid="2001">
    <head id="2" deprel="mod"/>
  </subword>
</sentence>
```

Figure 3: Decca-XML annotation for example (8)

Example (9) demonstrates the conversion of an abstract `coord` token when a conjunction is present. As shown in Figure 4, the abstract `coord` is replaced by the overt conjunction.

(9) Verkehrschaos durch Eisregen und Schnee
traffic chaos through freezing rain and snow
'Traffic chaos caused by freezing rain and snow'

```
mo(Chaos~0, durch~2)              mod(Regen~3, Eis~3001)
mod(Chaos~0, Verkehrs~1001)       cj(coord~501, Regen~3)
tiger_id(Chaos~0, 1)              cj(coord~501, Schnee~5)
obj(durch~2, coord~501)           coord_form(coord~501, und)
```

5 Conclusion and Outlook

We have shown how we performed a conversion of a dependency resource for German, the TiGerDB, into a more readily usable dependency annotation of the surface string, thus promoting work in parser evaluation. In performing the conversion, we accounted for abstract nodes, sublexical nodes, and word alignment issues, all of which makes the treebank more directly connected to the original data. Given the direct connection of the TiGerDB to the TIGER Treebank, the new resource can also contribute to a comparison of constituency-based and dependency-based corpus annotation.

ISSN 1736-6305 Vol. 1
http://dspace.utlib.ee/dspace
/handle/10062/4476

```
<sentence id="8810" externid="tiger-db-8810.fdsc">
  <word id="1" form="Verkehrschaos" lemma="Verkehrschaos" postag="NN"
        externid="0">
    <head id="0" deprel="root" />
  </word>
  <word id="2" form="durch" lemma="durch" postag="APPR" externid="2">
    <head id="1" deprel="mo" />
  </word>
  <word id="3" form="Eisregen" lemma="Eisregen" postag="NN" externid="3">
    <head id="4" deprel="cj" />
  </word>
  <word id="4" form="und" lemma="und" postag="KON" externid="501">
    <head id="2" deprel="obj" />
  </word>
  <word id="5" form="Schnee" lemma="Schnee" postag="NN" externid="5">
    <head id="4" deprel="cj" />
  </word>
  <subword id="1001" form="Verkehrs" postag="" externid="1001">
    <head id="1" deprel="mod" />
  </subword>
  <subword id="3001" form="Eis" postag="" externid="3001">
    <head id="3" deprel="mod" />
  </subword>
</sentence>
```

Figure 4: Decca-XML annotation for example (9)

An interesting opportunity for research and discussion in our opinion also arises from the fact that in addition to the TiGerDB and the dependency annotation we have derived from it, there also are two dependency representations which were directly extracted from the TIGER Treebank, one by Amit Dubey for the CoNLL-X Shared Task and another by Manuel Kountz and Martin Forst (Kountz 2006), which are both available in conjunction with the TIGER Treebank. Together with related work on closely related data (Daum et al. 2004), this should support a detailed exploration of the differences between dependency annotation schemes, both regarding the different dependency relations assumed and regarding the analysis of various phenomena, such as coordination, control, or auxiliaries as discussed in this paper. A discussion of the merits of each choice could also contribute to a standardization of dependency analysis.

On the practical side, we developed a flexible XML encoding format for dependency treebanks. The Decca-XML Schema Definition is freely available at `http://decca.osu.edu`. We will also provide the dependency resource resulting from the conversion to the TIGER Project, so that it can be made available in conjunction with the TIGER Treebank and the TiGerDB.

Acknowledgements This paper is based upon work supported by the National Science Foundation under Grant No. IIS-0623837. We would also like to thank Martin Forst for his kind assistance and the anonymous TLT reviewers for their useful comments.

References

Brants, S., S. Dipper, S. Hansen, W. Lezius and G. Smith (2002). The TIGER Treebank. In *Proceedings of TLT-02*. Sozopol, Bulgaria.

Daum, M., K. Foth and W. Menzel (2004). Automatic transformation of phrase treebanks to dependency trees. In *Proceedings LREC-04*. Lisbon, Portugal.

Dipper, S. (2003). *Implementing and Documenting Large-Scale Grammars — German LFG*. Ph.D. thesis. IMS, University of Stuttgart. AIMS Vol. 9(1).

Forst, M., N. Bertomeu, B. Crysmann, F. Fouvry, S. Hansen-Schirra and V. Kordoni (2004). Towards a Dependency-Based Gold Standard for German Parsers. The TIGER Dependency Bank. In *Proceedings of LINC-04*. Geneva, Switzerland.

Foth, K. (2006). *Eine umfassende Constraint-Dependenz-Grammatik des Deutschen*. Tech. rep., Universität Hamburg.

Hajičová, E., J. Panevova and P. Sgall (1998). Language resources need annotations to make them reusable: The Prague Dependency Treebank. In *Proceedings of LREC-98*. Granada, Spain, pp. 713–718.

King, T. H., R. Crouch, S. Riezler, M. Dalrymple and R. M. Kaplan (2003). The PARC 700 Dependency Bank. In *Proceedings of LINC-03*. Budapest, Hungary.

Kountz, M. (2006). *Extraktion von Dependenztripeln aus der TIGER-Baumbank*. Studienarbeit Nr. 54, IMS, Universität Stuttgart.

McDonald, R. and F. Pereira (2006). Online learning of approximate dependency parsing algorithms. In *Proceedings of EACL-06*. Trento, Italy.

Mel'čuk, I. A. (1988). *Dependency syntax: theory and practice*. SUNY series in linguistics. Albany, NY: State University Press of New York.

Nivre, J. and J. Hall (2005). MaltParser: A Language-Independent System for Data-Driven Dependency Parsing. In *Proceedings of TLT-05*. Barcelona, Spain.

Rehbein, I. and J. van Genabith (2007). Treebank Annotation Schemes and Parser Evaluation for German. In *Proceedings of EMNLP-CoNLL-07*. Prague.

Versley, Y. and H. Zinsmeister (2006). From Surface Dependencies towards Deeper Semantic Representations. In *Proceedings of TLT-06)*. Prague, Czech Republic.

ISSN 1736-6305 Vol. 1
http://dspace.utlib.ee/dspace
/handle/10062/4476