A Brief Introduction to the TIGER Treebank, Version 1

George Smith
Universität Potsdam

July 22, 2003

1 Introduction

This brief introduction provides a basic description of the structural format of Version 1 of the TIGER treebank, giving representative examples of the way linguistic phenomena are encoded. It is intended to guide the user through the first steps of becoming familiar with the treebank. The user is presumed to be a linguist specifically interested in German. A basic knowledge of the syntax and vocabulary of the German language is assumed.

The TIGER Treebank (Version1) is the product of an ongoing joint project at the the Department of Computational Linguistics and Phonetics in Saarbrücken,the Institute of Natural Language Processing (IMS) in Stuttgart, and the Institut für Germanistik in Potsdam[1]. Version 1 of the treebank consists of approximately 700,000 tokens (40,000 sentences) of German newspaper text, taken from the Frankfurter Rundschau.

The information encoded in the treebank can be most easily accessed via the TIGERSearch query tool (Lezius, 2002). Thus, for each phenomenon discussed here, sample queries in the TIGER query language are given which demonstrate how relevant sentences can be retrieved from the treebank via the query tool. These queries can be used as building blocks in the formulation of user queries. This document is not specifically intended as a tutorial for the query language, but it could provide a starting point for familiarizing oneself with the basic concepts of the language. The examples of queries given here progress from very simple to more complex and are accompanied by explanations. No prior knowledge of the query language is assumed. For introductory information regarding the TIGERSearch query tool and the TIGER language see König et al. (2003). For a formal definition of the language see König and Lezius (2003).

The annotation scheme used to encode grammatical information in the treebank builds on that developed for the NEGRA treebank (Skut et al., 1997, 1998; Brants et al., 1999). The corpus is divided into sentences. Each sentence has been given

[1]The TIGER project has been financed by the Deutsche Forschungsgemeinschaft since 1999

Nouns	N	Adverbs	ADV
Verbs	V	Conjunctions	KO
Articles	ART	Adpositions	AP
Adjectives	ADJ	Interjections	IT
Pronouns	P	Particles	PTK
Cardinal Numbers	CARD		

Table 1: Main STTS Categories

a structural description, henceforth annotation. The annotation can be divided into two parts: information specific to individual wordforms is encoded by tags associated with those forms; syntactic structure is encoded via a syntax graph, a tree-like structure with potentially crossing branches and labeled edges (König and Lezius, 2003). In this paper we will informally refer to trees. The wordforms of a sentence are tagged for part of speech[2]. A tree is built above the wordforms of a sentence. The nodes of the tree are labeled for their constituent category. The edges leading from one node to another are labeled according to the function of the child node in the constituent formed by the parent node. The treebank also contains secondary edges. Secondary edges are drawn between two nodes which are not in a relation of immediate dominance and indicate a syntactic relation between those nodes.

Searching the treebank using the TIGER query language involves describing nodes and relations between nodes via expressions in the language. These expressions can be combined using various Boolean operators. Basic node descriptions are introduced in section 2. Means of expressing relations between nodes are introduced in sections 3 and 4.

2 Part of Speech

The tagset for part of speech used in the TIGER Treebank is the Stuttgart-Tübinger Tagset (STTS), with minor variations. For a complete documentation of STTS see Schiller et al. (1999). A brief overview of the tagset is given in section A.1. Points where the tagset used in TIGER differs from Schiller et al. (1999) are given in section A.2. The STTS divides the wordforms of German into the eleven main categories shown in table 1. These main categories are then subject to varying degrees of further subdivision. A tag for a word form consists of the tag for the main word category followed by tags for the subcategories.

Table 2 uses the class of verbs as an example of the further subdivision of a main word category in STTS. It shows how verbs are further subdivided into full verbs, auxiliary verbs, and modal verbs, and how these are further subdivided into

[2]We are currently in the process of adding additional word-level tags. In version 2 of the treebank, inflected forms and certain other forms will be tagged with categories relevant for inflection, and each wordform will be tagged with its respective lemma.

Part of Speech	Verb Type	Finiteness	
V *Verb*	A *Auxiliar*	FIN	*finit*
		INF	*infinit*
		IMP	*imperativ*
		PP	*Partizip Perfekt*
	M *Modal*	FIN	*finit*
		INF	*infinit*
		PP	*Partizip Perfekt*
	V *Voll*	FIN	*finit*
		INF	*infinit*
		IZU	*Infinitiv mit zu*
		IMP	*imperativ*
		PP	*Partizip Perfekt*

Table 2: Subcategories of Verbs in STTS

finite forms and several categories of non-finite forms: infinitives, infinitives with *zu*, past participles and imperatives. The full label for a verb form is a concatenation of the abbreviation for the main category, verb type, and finiteness/infinitive-type.

(1) a. gewesen
VAPP

b. könnte
VMFIN

c. abzugeben
VVIZU

A past participle of an auxiliary verb would then be given the tag in (1a) (*Verb, auxiliar, Partizip Perfekt*). A finite form of a modal verb would be given the tag in (1b) (*Verb, modal, finit*), and a single infinitive wordform containing *zu* would be given the tag in (1c) (*Verb, voll, infitiv, zu*). An example of part of speech tagging of a sentence is given in (2).

(2) Jetzt solle erneut ein Antrag gestellt werden .
ADV VMFIN ADJD ART NN VVPP VAINF $.

A search for wordforms belonging to a particular category is accomplished by using an expression in the TIGER language known as a *node description*. The simplest node description consists of an expression known as a *feature constraint*. The simplest feature constraint consists of a single feature-value pair. A feature and a value are separated by an equal sign.

(3) a. `[pos = "ART"]`

b. `[pos = "NN"]`

c. `[pos = "VVFIN"]`

The node description (3a) describes a node with the value `"ART"` (*Artikel*) for the feature `pos` (part of speech). In other words, this feature description can be used to access an article. Similarly, the node description (3b) would access a common noun (a node with the value `"NN"`, *Nomen, normal*, for the feature `pos`). And (3c) would access a finite form of a full verb (a node with the value `"VVFIN"`, *Verb, voll, finit*, for the feature `pos`). It is of course not always desirable to search solely for the most specific category. Less specific categories can be accessed by combining more specific categories. One way of accomplishing this is by using Boolean operators, another way is by using types.

Within the class of verbs, for example, one could search specifically for finite forms of full verbs, using the node description (4a). One could also search for finite forms of auxiliary verbs with the node description (4b), or for finite forms of modal verbs with the node description (4c). Or one could search for all finite verb forms, using the single feature constraint (4d) This node description uses the Boolean operator for disjunction `|`. Parentheses indicate grouping.

(4) a. `[pos = "VVFIN"]`

b. `[pos = "VAFIN"]`

c. `[pos = "VMFIN"]`

d. `[pos = ("VVFIN" | "VAFIN" | "VMFIN")]`

The double quotes in (4a–4c) signify constants, namely the strings `"VVFIN"`, `"VAFIN"` or `"VMFIN"`. These feature constraints accept nodes with the given string as a value for the feature `pos`. The constraint (4d) contains three constants and the Boolean operator for disjunction and accepts nodes with a value of either `"VVFIN"`, `"VAFIN"` or `"VMFIN"`. The Boolean expression describes a set of possible feature values. The parentheses indicate grouping.

(5) a. `finite := VAFIN, VVFIN, VMFIN;`

b. `[pos = finite]`

Another way of searching for finite verbs is by defining a type. A pre-defined type hierarchy for part of speech has been developed for use with the TIGER Treebank (Version 1). Users can also define their own type hierarchy (König et al., 2003). The type definition in (5a) defines a type for finite verb forms. The type `finite` can then be used in a node description such as (5b) to search for finite verb forms. A type, once defined, can be reused. It is ultimately simpler and more economical to use pre-defined types, rather than Boolean expressions for sets of feature values which are frequently searched for. The user can define a hierarchy of types for each feature. A type definition hierarchy is stored in a file and is thus specifically intended for reuse. Well-defined type hierarchies can simplify the con-

struction of queries considerably. In constructing a type hierarchy, a user can map the tagsets used in TIGER to a terminology of the users own choosing, making it easier to pose queries in an intuitive way.

3 Syntactic Structure

Syntactic structure is expressed by means of a graph structure. Constituent categories are encoded in node labels: non-terminal node labels represent phrasal categories; terminal nodes represent wordforms and are tagged as described in section 2. Syntactic functions are encoded in edge labels, discussed in section 4. A simple example from the treebank is given in figure 1.

Constituent structure trees in the TIGER Treebank tend to be flat; redundant branching is eliminated. The distinction between arguments and adjuncts is not made in constituent structure, but is expressed via syntactic functions expressed via edge labels (cf. section 4). In general, ergonomics has been an important design criterion. The elimination of redundant structures, which can be found in many linguistic descriptions but are not actually needed for processing queries, results in tree structures which are more easily viewed on screen. Two examples of this elimination of redundant information are the lack of non-branching NP nodes[3] and the encoding of prepositional phrases without a separate NP node for the noun phrase immediately dominated by the prepositional phrase[4]. In the former case, the category of the constituent formed by the wordform is derivable from the part of speech tag. In the latter case, the set of forms comprising the noun phrase within the prepositional phrase is completely predictable.

The branches of a tree may cross (cf. figure 2), allowing local and non-local dependencies to be encoded in a uniform way and eliminating the need for traces. This approach has significant advantages for non-configurational languages such as German, which exhibit a rich inventory of discontinuous constituency types and considerable freedom with respect to word order.

Searching for syntactic structures with the TIGERSearch software involves expressing relations between nodes. A number of predicates for expressing such relations are introduced in König et al. (2003). There are predicates for expressing relations of dominance (see examples 6–8) and predicates for expressing relations of linear precedence (see examples 9–11). In addition, there are predicates which directly express constraints on graphs (see example 12).

(6) a. `[cat = "NP"]`

b. `[cat = "NP"] > [pos = "NN"]`

[3] For an example of the lack of non-branching nodes, see the pronoun *sie* and the plural common nouns *Verpackungen* and *Etiketten* in figure 3

[4] For an example, see the two prepositional phrases in figure 1.

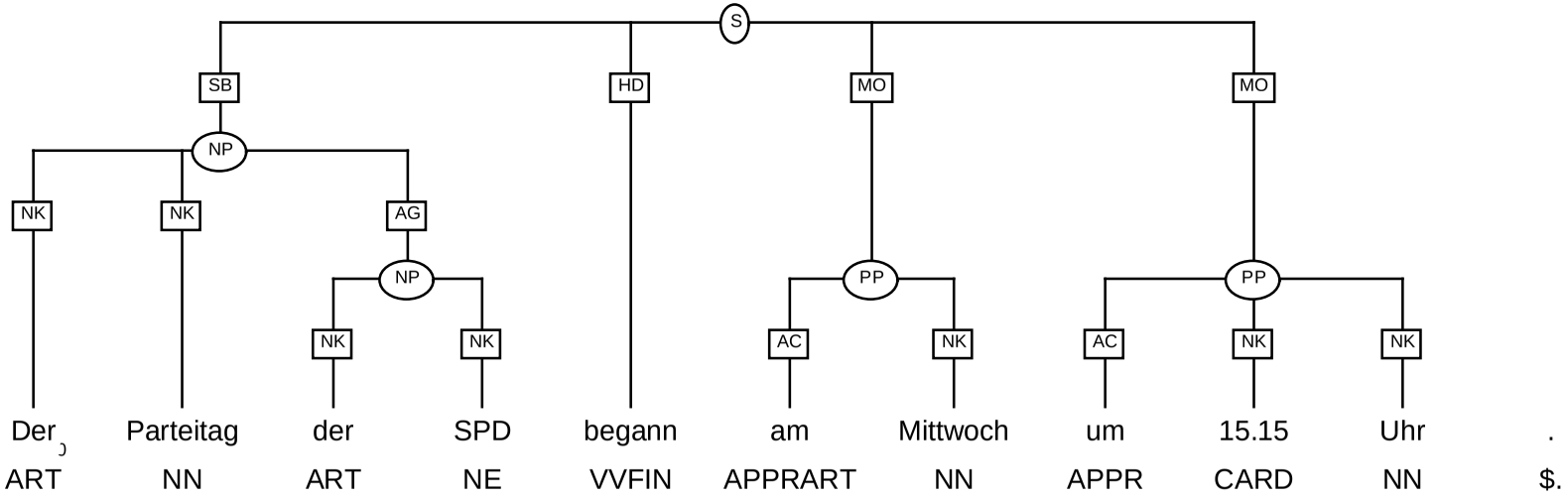

Figure 1: Basic Constituent Structure

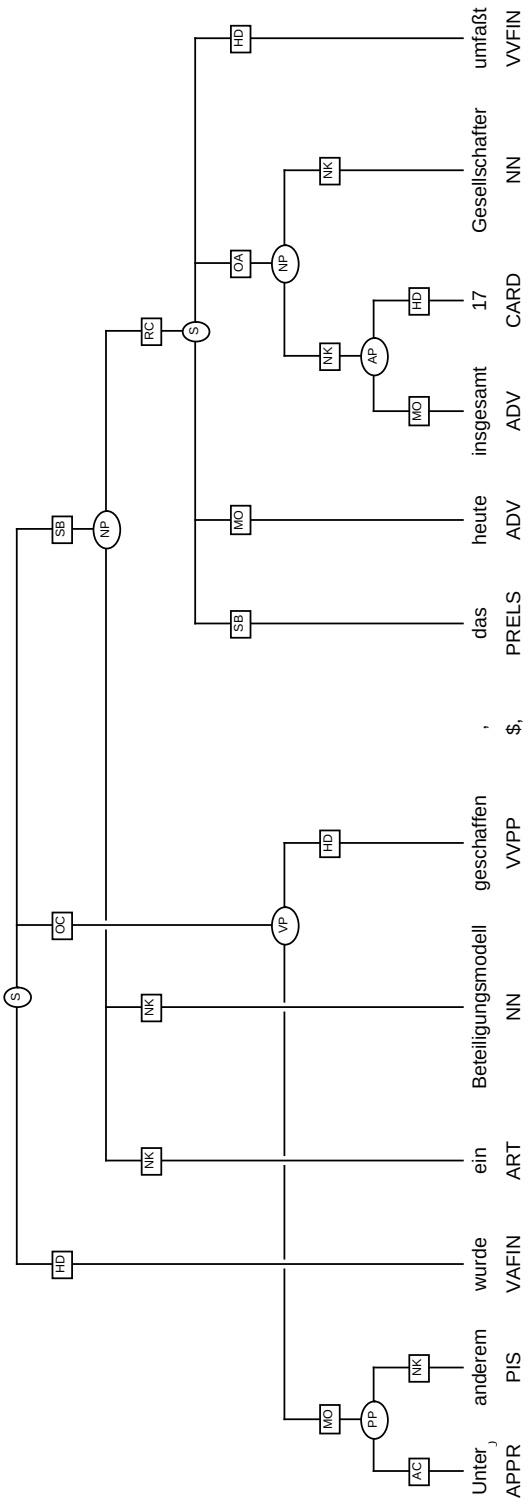

Figure 2: Discontinuous Constituents

The query in (6a) is matched by all noun phrases, that is, nodes with the value NP for the feature cat (category). The query in (6b) uses the operator for immediate dominance >. This operator expresses a relation of immediate dominance between two nodes. The query matches any noun phrase which immediately dominates a common noun. More precisely, it matches any structures consisting of a node with the value NP for the feature cat which immediately dominates a node with the value NN for the feature pos.

(7) a. `#xyz`

b. `#xyz:[cat = "NP"]`

c.
```
#np:[cat = "NP"] &
#np > [pos = "NN"]
```

When formulating a query, it is often necessary to refer to a particular node more than once. This is done by using variables (König et al., 2003). The symbol # indicates that what follows is a variable name. Example (7a) shows a valid variable name. It is also a simple query, matching any node, as the variable is not constrained. Variables may be constrained, as in example (7b), in which the variable xyz is constrained to match nodes of type [cat = "NP"]. Note the use of the symbol : to indicate that the variable is constrained. A simple example of the use of a variable is given in (7c). Here, a variable name is chosen which is mnemonically convenient, a practice which makes more complex queries easier to read. This query accesses the same set of nodes as the query in example (6b). It is not necessary to use a variable here; the example is intended to illustrate the syntax for using variables for node descriptions. The query in example (8), however, could not have been formulated without the use of a variable.

(8)
```
#np:[cat = "NP"] &
#np > [pos = "ART"] &
#np > [pos = "NE"]
```

In example (8), we see three lines, each containing an expression representing a condition on syntactic structure. The expressions in these lines are joined by the Boolean operator for conjunction (&). In the first line, we see a constrained variable: there is a node #np with the value "NP" for the feature cat. In the second line of the query the condition is expressed that the node referred to by the variable #np dominates a node with the value ART for the feature pos. In the third line, the condition is expressed that the node #np also dominates a node with the value NE for the feature pos. The query is thus matched by any noun phrase which immediately dominates both an article (ART) and a proper noun (NE).

(9) `[word = "noch"] . [word = "nicht"]`

The queries in examples (9–11) demonstrate the use of two operators for relations of precedence: . and .*. The operator . indicates immediate precedence. The query in example (9) would access all instances in which the wordform *noch* immediately precedes the wordform *nicht*.

(10)
```
#s:[cat = "S"] &
#n:[cat = "NP"] &
#v:[pos = "VAFIN"] &
#s >SB #n &
#s > #v &
#v .* #n
```

A more complex example of the use of a predicate expressing a relation of linear precedence is given in (10). The query would access all sentences in which a noun phrase functioning as subject is preceded by a finite form of an auxiliary verb, such as the sentence in figure 2. The operator `.*` denotes precedence which may be, but need not be, immediate. The first three lines of the query contain expressions constraining the three variables: `#s`, `#n` and `#v`. In the fourth line, a relation of labeled immediate dominance is expressed (see section 4). In other words, `#n` is the subject (`SB`) of `#s`. The fifth line contains an expression indicating that `#v` is immediately dominated by `#s`. The last line contains an expression indicating that `#v` occurs at some point in the sentence before `#n`.

(11)
```
#s:[cat = "S"] >SB #n:[cat = "NP"] &
#s > #v:[pos = "VAFIN"] &
#v .* #n
```

The query (10) can be reformulated in a more compact way. In (10), the first three lines simply constrain the three variables used. This makes the query easy to read. It is not necessary to do this however, as seen in the equivalent query in (11). It is a matter of personal taste whether the formulation in (10) or (11) is preferable. For those not accustomed to using query languages, it may be better in the long run to begin writing queries that are formulated in a simpler more structured style, as debugging queries which are hard to understand requires considerably more time than does a bit of extra typing.

(12) a.
```
#n:[cat="NP"] & discontinuous(#n)
```

b.
```
#n:[cat="NP"] & continuous(#n)
```

The queries in (12) show two examples of predicates on graphs. The notion discontinuous constituent can be expressed by combining the use of relations of dominance with the use of relations of linear precedence. Expressing this in a query would not be simple, however. The predicates `continuous` and `discontinuous` are included in the TIGER language to provide a simple way of differentiating continuous and discontinuous constituents. The query (12a) would access all discontinuous noun phrases, such as that in figure 2 and the query (12b) would access all continuous noun phrases. Note that predicates on graphs require the use of a variable. For other predicates on graphs, see König et al. (2003).

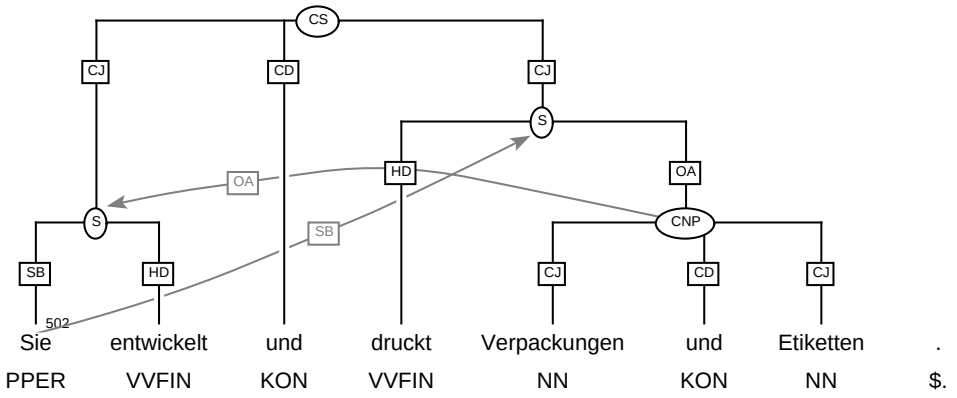

Figure 3: Secondary Edges

4 Syntactic Relations

Syntactic relations are encoded on the edge drawn from a parent node to a child node. The sentence in figure 1 contains a noun phrase (NP) which itself contains another noun phrase, the latter functioning as a genitive attribute (AG). This syntactic relation is indicated by a label on the edge drawn from parent to child. It is possible to search for pairs of nodes which are in a particular syntactic relation by using an operator for labeled immediate dominance such as >AG (attribut, genitive) or >RC (relative clause) in (13).

(13) a. `[cat = "NP"] >AG [cat = "NP"]`

b. `[] >RC [cat = "S"]`

The expression in (13a) would access all nodes of the category noun phrase (NP) which themselves have a child of the category noun phrase (NP) which functions as a genitive attribute (AG). The expression in (13b) would be matched by all nodes of any kind which have a daughter which is a sentence (S) which functions as a relative clause (RC). The node description `[]` contains no feature-value pairs and is thus matched by any node. The query (13b) would, for example, access the subject of the sentence in figure 2.

Syntactic relations which obtain between nodes which are not in a relation of immediate dominance are encoded via secondary edges. In figure 3, the initial wordform *Sie* functions not only as subject of the first of the two coordinated sentences, but also as subject of the second. Likewise, the coordinated noun phrase (CNP) functions as accusative object of both sentences.

(14) a. `[] >~SB []`

b. `[] >~OA []`

The query (14a) would access all cases in which the secondary syntactic relation of subject (SB) holds between two nodes. The query (14b) can similarly be used to search for secondary accusative objects (OA).

(15) a. `#n:[cat="NP"] >RC [cat="S"] & discontinuous(#n)`

b. `#n:[cat="NP"] >RC [cat="S"] & continuous(#n)`

The queries in (15) use two predicates on graphs we saw in section 3 along with a relation of labeled immediate dominance. The query (15a) would access all discontinuous noun phrases which immediately dominate a sentence which functions as a relative clause. This query would access noun phrases containing extraposed relative clauses such as the subject of the sentence in figure 2. The query (15b) would access noun phrases which immediately dominate a sentence which functions as a relative clause, but which are not discontinuous, excluding those containing extraposed relative clauses.

5 Conclusion

This document was intended as a brief introduction to the TIGER Treebank (Version 1). The best way to explore the treebank at this point is to experiment with the TIGERSearch query tool, possibly modifying and expanding upon the various queries given here. An introduction to the query language, including a complete list of operators, can be found in König et al. (2003). For a formal definition of the query language see König and Lezius (2003). Examples of queries relating to word structure can be found in Smith (2003). These and other documents can be found on the TIGER website, located at:

http://www.ims.uni-stuttgart.de/projekte/TIGER/

A Tagsets

A.1 The Tagset for Part of Speech

The tagset for part of speech is the STTS tagset (Schiller et al., 1999) with minor modifications outlined in A.2. As the abbreviations of the STTS tagset are based on German terminology, it may be easier for readers of German to use the original documentation in Schiller et al. (1999). English translations are given here for those less familiar with German linguistic terminology.

ADJA	adjective, attributive	[das] große [Haus]
ADJD	adjective, adverbial or predicative	[er fährt] schnell, [er ist] schnell
ADV	adverb	schon, bald, doch
APPR	preposition; circumposition left	in [der Stadt], ohne [mich]
APPRART	preposition with article	im [Haus], zur [Sache]
APPO	postposition	[ihm] zufolge, [der Sache] wegen
APZR	circumposition right	[von jetzt] an
ART	definite or indefinite article	der, die, das, ein, eine, ...
CARD	cardinal number	zwei [Männer], [im Jahre] 1994
FM	foreign language material	[Er hat das mit “] A big fish [” übersetzt]
ITJ	interjection	mhm, ach, tja
KOUI	subordinate conjunction with *zu* and infinitive	um [zu leben], anstatt [zu fragen]
KOUS	subordinate conjunction with sentence	weil, daß, damit, wenn, ob
KON	coordinate conjunction	und, oder, aber
KOKOM	comparative conjunction	als, wie
NN	common noun	Tisch, Herr, [das] Reisen
NE	proper noun	Hans, Hamburg, HSV
PDS	substituting demonstrative pronoun	dieser, jener
PDAT	attributive demonstrative pronoun	jener [Mensch]
PIS	substituting indefinite pronoun	keiner, viele, man, niemand
PIAT	attributive indefinite pronoun without determiner	kein [Mensch], irgendein [Glas]
PIDAT	attributive indefinite pronoun with determiner	[ein] wenig [Wasser], [die] beiden [Brüder]
PPER	non-reflexive personal pronoun	ich, er, ihm, mich, dir
PPOSS	substituting possessive pronoun	meins, deiner
PPOSAT	attributive possessive pronoun	mein [Buch], deine [Mutter]

PRELS	substituting relative pronoun	[der Hund ,] der
PRELAT	attributive relative pronoun	[der Mann ,] dessen [Hund]
PRF	reflexive personal pronoun	sich, dich, mir
PWS	substituting interrogative pronoun	wer, was
PWAT	attributive interrogative pronoun	welche [Farbe], wessen [Hut]
PWAV	adverbial interrogative or relative pronoun	warum, wo, wann, worüber, wobei
PAV	pronominal adverb	dafür, dabei, deswegen, trotzdem
PTKZU	*zu* before infinitive	zu [gehen]
PTKNEG	negative particle	nicht
PTKVZ	separable verbal particle	[er kommt] an, [er fährt] rad
PTKANT	answer particle	ja, nein, danke, bitte
PTKA	particle with adjective or adverb	am [schönsten], zu [schnell]
SGML	SGML markup	<turnid=n022k_TS2004>
SPELL	letter sequence	S-C-H-W-E-I-K-L
TRUNC	word remnant	An- [und Abreise]
VVFIN	finite verb, full	[du] gehst, [wir] kommen [an]
VVIMP	imperative, full	komm [!]
VVINF	infinitive, full	gehen, ankommen
VVIZU	Infinitive with *zu*, full	anzukommen, loszulassen
VVPP	perfect participle, full	gegangen, angekommen
VAFIN	finite verb, auxiliary	[du] bist, [wir] werden
VAIMP	imperative, auxiliary	sei [ruhig !]
VAINF	infinitive, auxiliary	werden, sein
VAPP	perfect participle, auxiliary	gewesen
VMFIN	finite verb, modal	dürfen
VMINF	infinitive, modal	wollen
VMPP	perfect participle, modal	gekonnt, [er hat gehen] können
XY	non-word containing non-letter	3:7, H2O, D2XW3
$,	comma	,
$.	sentence-final punctuation mark	. ? ! ; :
$(	other sentence-internal punctuation mark	- [,]()

A.2 Deviation from STTS in the TIGER Treebank

- PIDAT vs. PIAT
  No distinction is made between PIAT and PIDAT. PIAT is used for indefinite pronouns in an attributive function both with and without determiner.

- ADV
  Prepositions are tagged as ADV when the modify numerals.

- PAV vs. PROAV
  The tag PROAV replaces the STTS tag PAV.

A.3 The Tagset for Node Labels

AA	superlative phrase with *am*
AP	adjective phrase
AVP	adverbial phrase
CAC	coordinated adposition
CAP	coordinated adjective phrase
CAVP	coordinated adverbial phrase
CCP	coordinated complementiser
CH	chunk
CNP	coordinated noun phrase
CO	coordination
CPP	coordinated adpositional phrase
CS	coordinated sentence
CVP	coordinated verb phrase (non-finite)
CVZ	coordinated infinitive with *zu*
DL	discourse level constituent
ISU	idiosyncratic unit
MTA	multi-token adjective
NM	multi-token number
NP	noun phrase
PN	proper noun
PP	adpositional phrase
QL	quasi-language
S	sentence
VP	verb phrase (non-finite)
VZ	infinitive with *zu*

A.4 The Tagset for Edge Labels

AC	adpositional case marker
ADC	adjective component
AG	genitive attribute
AMS	measure argument of adjective
APP	apposition
AVC	adverbial phrase component
CC	comparative complement
CD	coordinating conjunction
CJ	conjunct
CM	comparative conjunction
CP	complementizer
CVC	collocational verb construction (*Funktionsverbgefüge*)
DA	dative
DH	discourse-level head
DM	discourse marker
EP	expletive *es*
HD	head
JU	junctor
MNR	postnominal modifier
MO	modifier
NG	negation
NK	noun kernel element
NMC	numerical component
OA	accusative object
OA	second accusative object
OC	clausal object
OG	genitive object
OP	prepositional object
PAR	parenthetical element
PD	predicate
PG	phrasal genitive
PH	placeholder
PM	morphological particle
PNC	proper noun component
RC	relative clause
RE	repeated element
RS	reported speech
SB	subject

SBP	passivised subject (PP)
SP	subject or predicate
SVP	separable verb prefix
UC	unit component
VO	vocative

References

Brants, Thorsten, Wojciech Skut, and Hans Uszkoreit. 1999. Syntactic annotation of a German newspaper corpus. In *Proceedings of the ATALA Treebank Workshop*, 69–76. Paris, France.

König, Esther, and Wolfgang Lezius. 2003. The TIGER language - a description language for syntax graphs, Formal definition. Tech. rep., IMS, University of Stuttgart.

König, Esther, Wolfgang Lezius, and Holger Voormann. 2003. *TIGERSearch User's Manual*. IMS, University of Stuttgart, Stuttgart.

Lezius, Wolfgang. 2002. Ein suchwerkzeug für syntaktisch annotierte textkorpora. Ph.D. thesis, IMS, University of Stuttgart. Arbeitspapiere des Instituts ür Maschinelle Sprachverarbeitung (AIMS), Volume 8, Number 4.

Schiller, Anne, Simone Teufel, Christine Stöckert, and Christine Thielen. 1999. Guidelines für das Tagging deutscher Textcorpora mit STTS. Universität Stuttgart and Universität Tübingen.

Skut, Wojciech, Thorsten Brants, Brigitte Krenn, and Hans Uszkoreit. 1998. A linguistically interpreted corpus of German newspaper text. In *Proceedings of the Conference on Language Resources and Evaluation LREC-98*, 705–711. Granada, Spain.

Skut, Wojciech, Brigitte Krenn, Thorsten Brants, and Hans Uszkoreit. 1997. An annotation scheme for free word order languages. In *Proceedings of ANLP-97*. Washington.

Smith, George. 2003. Searching for morphological structure with regular expressions. Tiger Projektbericht, Univ. Potsdam.