From Chunks to Function-Argument Structure: A Similarity-Based Approach

Sandra Kübler† and **Erhard W. Hinrichs†**

†Seminar für Sprachwissenschaft
University of Tübingen
D-72074 Tübingen, Germany
{kuebler,eh}@sfs.nphil.uni-tuebingen.de

Abstract

Chunk parsing has focused on the recognition of partial constituent structures at the level of individual chunks. Little attention has been paid to the question of how such partial analyses can be combined into larger structures for complete utterances. Such larger structures are not only desirable for a deeper syntactic analysis. They also constitute a necessary prerequisite for assigning function-argument structure.

The present paper offers a similarity-based algorithm for assigning functional labels such as *subject, object, head, complement*, etc. to complete syntactic structures on the basis of pre-chunked input.

The evaluation of the algorithm has concentrated on measuring the quality of functional labels. It was performed on a German and an English treebank using two different annotation schemes at the level of function-argument structure. The results of 89.73 % correct functional labels for German and 90.40 % for English validate the general approach.

1 Introduction

Current research on natural language parsing tends to gravitate toward one of two extremes: robust, partial parsing with the goal of broad data coverage versus more traditional parsers that aim at complete analysis for a narrowly defined set of data. Chunk parsing (Abney, 1991; Abney, 1996) offers a particularly promising and by now widely used example of the former kind. The main insight that underlies the chunk parsing strategy is to isolate the (finite-state) analysis of non-recursive syntactic structure, i.e. chunks, from larger, recursive structures. This results in a highly-efficient parsing architecture that is realized as a cascade of finite-state transducers and that pursues a leftmost longest-match pattern-matching strategy at each level of analysis.

Despite the popularity of the chunk parsing approach, there seems to be a gap in current research:

Chunk parsing research has focused on the recognition of partial constituent structures at the level of individual chunks. By comparison, little or no attention has been paid to the question of how such partial analyses can be combined into larger structures for complete utterances. Such larger structures are not only desirable for a deeper syntactic analysis; they also constitute a necessary prerequisite for assigning function-argument structure.

Automatic assignment of function-argument structure has long been recognized as a desideratum beyond pure syntactic labeling (Marcus et al., 1994)[1]. The present paper offers a similarity-

[1]With the exception of dependency-grammar-based parsers (Tapanainen and Järvinen, 1997; Bröker et al., 1994; Lesmo and Lombardo, 2000), where functional labels are treated as first-class citizens as relations between words, and recent work on a semi-automatic method for treebank construction (Brants et al., 1997), little has been reported on

based algorithm for assigning functional labels such as *subject, object, head, complement*, etc. to complete syntactic structures on the basis of pre-chunked input. The evaluation of the algorithm has concentrated on measuring the quality of these functional labels.

2 The TüSBL Architecture

In order to ensure a robust and efficient architecture, TüSBL, a similarity-based chunk parser, is organized in a three-level architecture, with the output of each level serving as input for the next higher level. The first level is part-of-speech (POS) tagging of the input string with the help of the bigram tagger LIKELY (Feldweg, 1993).[2] The parts of speech serve as pre-terminal elements for the next step, i.e. the chunk analysis. Chunk parsing is carried out by an adapted version of Abney's (1996) CASS parser, which is realized as a cascade of finite-state transducers. The chunks, which extend if possible to the simplex clause level, are then remodeled into complete trees in the tree construction level.

The tree construction level is similar to the DOP approach (Bod, 1998; Bod, 2000) in that it uses complete tree structures instead of rules. Contrary to Bod, we only use the complete trees and do not allow tree cuts. Thus the number of possible combinations of partial trees is strictly controlled. The resulting parser is highly efficient (3770 English sentences took 106.5 seconds to parse on an Ultra Sparc 10).

3 Chunking and Tree Construction

The division of labor between the chunking and tree construction modules can best be illustrated by an example.

For sentences such as the input shown in Fig. 1, the chunker produces a structure in which some constituents remain unattached or partially annotated in keeping with the chunk-parsing strategy to factor out recursion and to resolve only unambiguous attachments.

Since chunks are by definition non-recursive structures, a chunk of a given category cannot contain another chunk of the same type. In the case at hand, the two prepositional phrases ('prep_p') *about nine* and *in the evening* in the chunk output cannot be combined into a single chunk, even though semantically these words constitute a single constituent. At the level of tree construction, as shown in Fig. 2, the prohibition against recursive phrases is suspended. Therefore, the proper PP attachment becomes possible. Additionally, the phrase *about nine* was wrongly categorized as a 'prep_p'. Such miscategorizations can arise if a given word can be assigned more than one POS tag. In the case of *about* the tags 'in' (for: *preposition*) or 'rb' (for: *adverb*) would be appropriate. However, since the POS tagger cannot resolve this ambiguity from local context, the underspecified tag 'about' is assigned, instead. However, this can in turn lead to misclassification in the chunker.

Input: *alright and that should get us there about nine in the evening*

Chunk parser output:

```
[uh  alright]
[simpx_ind
   [cc      and]
   [that    that]
   [vp      [md    should]
            [vb    get]]
   [pp      us]
   [adv     [rb    there]]
   [prep_p  [about about]
            [np    [cd      nine]]]
   [prep_p  [in    in]
            [np    [dt      the]
                   [daytime evening]]]]
```

Figure 1: Chunk parser output.

The most obvious deficiency of the chunk output shown in Fig. 1 is that the structure does not contain any information about the function-argument structure of the chunked phrases. However, once a (more) complete parse structure is created, the grammatical function of each major constituent needs to be identified. The labels SUBJ (for: *subject*), HD (for: *head*), ADJ (for: *adjunct*) COMP (for: *complement*), SPR (for: *specifier*), which appear as edge-labels between tree nodes in Fig. 2, signify the grammatical functions of the constituents in question. E.g. the label SUBJ encodes that the NP *that* is the

fully automatic recognition of functional labels.

[2]The inventory of POS tags is based on the STTS (Schiller et al., 1995) for German and on the Penn Treebank tagset (Santorini, 1990) for English.

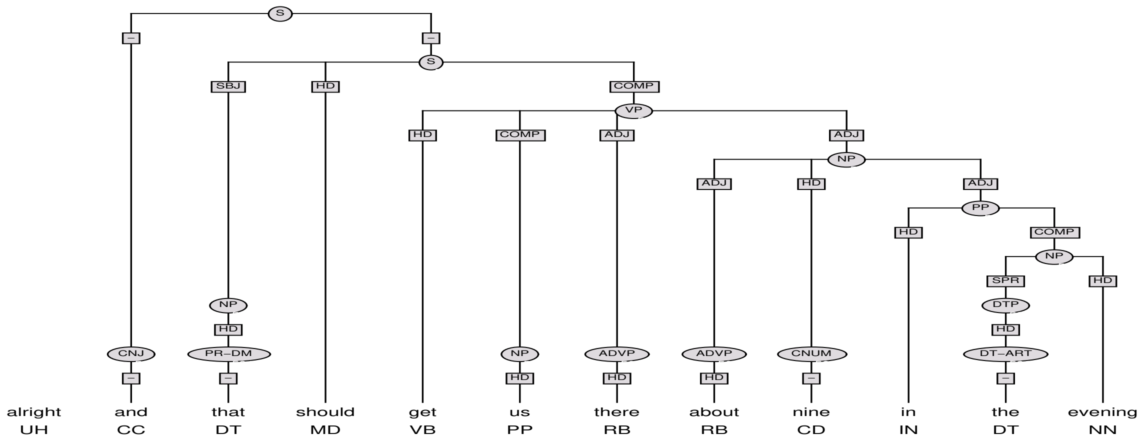

Figure 2: Sample tree construction output for the sentence in Fig. 1.

subject of the whole sentence. The label ADJ above the phrase *about nine in the evening* signifies that this phrase is an adjunct of the verb *get*.

TüSBL currently uses as its instance base two semi-automatically constructed treebanks of German and English that consist of appr. 67,000 and 35,000 fully annotated sentences, respectively[3]. Each treebank uses a different annotation scheme at the level of function-argument structure[4]. As shown in Table 1, the English treebank uses a total of 13 functional labels, while the German treebank has a richer set of 36 function labels.

For German, therefore, the task of tree construction is slightly more complex because of the larger set of functional labels. Fig. 3 gives an example for a German input sentence and its corresponding chunk parser output.

In this case, the subconstituents of the extraposed coordinated noun phrase are not attached to the simplex clause that ends with the non-finite verb that is typically in clause-final position in declarative main clauses of German. Moreover, each conjunct of the coordinated noun phrase forms a completely flat structure. TüSBL's tree construction module enriches the chunk output as shown in Fig. 4. Here the internally recursive NP conjuncts have been coordinated and integrated correctly into the clause as a whole. In addition, function labels such as MOD (for: *modifier*), HD (for *head*), ON (for: *subject*), OA (for: *direct object*), OV (for: *verbal object*), and APP (for: *apposition*) have been added that encode the function-argument structure of the sentence.

[3]See (Stegmann et al., 2000; Kordoni, 2000) for further details.

[4]The annotation for German follows the topological-field-model standardly used in empirical studies of German syntax. The annotation for English is modeled after the theoretical assumptions of Head-Driven Phrase Structure Grammar.

Input:
dann w”urde ich vielleicht noch vorschlagen Donnerstag den elften und Freitag den zw”olften August (then I would suggest maybe Thursday eleventh and Friday twelfth of August)

Chunk parser output:

```
[simpx [advx   [adv    dann]]
       [vxfin  [vafin  w"urde]]
       [nx2    [pper   ich]]
       [advx   [adv    vielleicht]]
       [advx   [advmd  noch]]
       [vvinf  vorschlagen]]
[nx3   [day    Donnerstag]
       [art    den]
       [adja   elften]]
[kon   und]
[nx3   [day     Freitag]
       [art     den]
       [adja    zw"olften]
       [month   August]]
```

Figure 3: Chunk parser output for a German sentence.

4 Similarity-based Tree Construction

The tree construction algorithm is based on the machine learning paradigm of *memory-based*

German label	description	English label	description
HD	head	HD	head
-	non-head	-	intentionally empty
ON	nominative object	COMP	complement
OD	dative object	SPR	specifier
OA	accusative object	SBJ	subject
OS	sentential object	SBQ	subject, wh-
OPP	prepositional object	SBR	subject, rel.
OADVP	adverbial object	ADJ	adjunct
OADJP	adjectival object	ADJ?	adjunct ambiguities
PRED	predicate	FIL	filler
OV	verbal object	FLQ	filler, wh-
FOPP	optional prepositional object	FLR	filler, rel.
VPT	separable verb prefix	MRK	marker
APP	apposition		
MOD	ambiguous modifier		
x-MOD	8 distinct labels for specific modifiers, e.g. V-MOD		
*y*K	13 labels for second conjuncts in split-up coordinations, e.g. ONK		

Table 1: The functional label set for the German and the English treebanks.

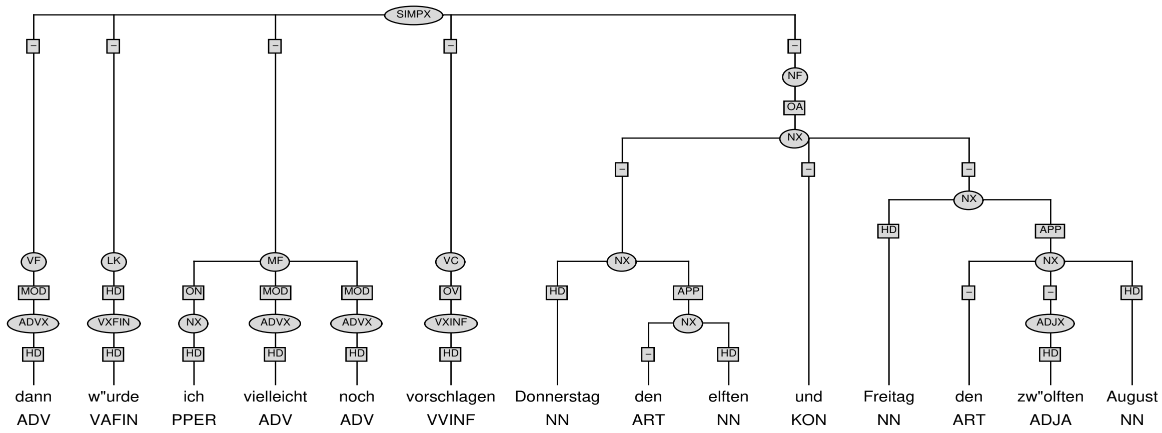

Figure 4: Tree construction output for the German sentence in Fig. 3.

learning (Stanfill and Waltz, 1986).[5] Memory-based learning assumes that the classification of a given input should be based on the similarity to previously seen instances of the same type that have been stored in memory. This paradigm is an instance of *lazy learning* in the sense that these previously encountered instances are stored "as is" and are crucially not abstracted over, as is typically the case in rule-based systems or other learning approaches. Previous applications of memory-based learning to NLP tasks consisted of classification problems in which the set of classes to be learnt was simple in the sense that the class items did not have any internal structure and the number of distinct items was small. Since in the current application, the set of classes are parse trees, the classification task is much more complex. The classification is simple only in those cases where a direct hit is found, i.e. where a complete match of the input with a stored instance exists. In all other cases, the most similar tree from the instance base needs to be modified to match the chunked input. This means that the output tree will group together only those elements from the chunked input for which there is evidence in

[5] Memory-based learning has recently been applied to a variety of NLP classification tasks, including part-of-speech tagging, noun phrase chunking, grapheme-phoneme conversion, word sense disambiguation, and PP attachment (see (Daelemans et al., 1999; Veenstra et al., 2000; Zavrel et al., 1997) for details).

the instance base. If these strategies fail for complete chunks, TüSBL attempts to match smaller subchunks.

The algorithm used for tree construction is presented in a slightly simplified form in Figs. 5-8. For readability, we assume here that chunks and complete trees share the same data structure so that subroutines like *string_yield* can operate on both of them indiscriminately.

The main routine *construct_tree* in Fig. 5 separates the list of input chunks and passes each one to the subroutine *process_chunk* in Fig. 6 where the chunk is then turned into one or more (partial) trees. *process_chunk* first checks if a complete match with an instance from the instance base is possible.[6] If this is not the case, a partial match on the lexical level is attempted. If a partial tree is found, *attach_next_chunk* in Fig. 7 and *extend_tree* in Fig. 8 are used to extend the tree by either attaching one more chunk or by resorting to a comparison of the missing parts of the chunk with tree extensions on the POS level. *attach_next_chunk* is necessary to ensure that the best possible tree is found even in the rare case that the original segmentation into chunks contains mistakes. If no partial tree is found, the tree construction backs off to finding a complete match at the POS level or to starting the subroutine for processing a chunk recursively with all the subchunks of the present chunk.

The application of memory-based techniques is implemented in the two subroutines *complete_match* and *partial_match*. The presentation of the two cases as two separate subroutines is for expository purposes only. In the actual implementation, the search is carried out only once. The two subroutines exist because of the postprocessing of the chosen tree, which is necessary for partial matches and which also deviates from standard memory-based applications. Postprocessing mainly consists of shortening the tree from the instance base so that it covers only those parts of the chunk that could be matched. However, if the match is done on the lexical level, a correction of tagging errors is possible if there is enough evidence in the instance base. TüSBL currently uses an *overlap metric*, the most basic metric for instances with symbolic features, as its similarity metric. This overlap metric is based on either lexical or POS features. Instead of applying a more sophisticated metric like the weighted overlap metric, TüSBL uses a backing-off approach that heavily favors similarity of the input with prestored instances on the basis of substring identity. Splitting up the classification and adaptation process into different stages allows TüSBL to prefer analyses with a higher likelihood of being correct. This strategy enables corrections of tagging and segmentation errors that may occur in the chunked input.

[6] *string_yield* returns the sequence of words included in the input structure, *pos_yield* the sequence of POS tags.

5 Quantitative Evaluation

Quantitive evaluations of robust parsers typically focus on the three PARSEVAL measures: labeled precision, labeled recall and crossing accuracy. It has frequently been pointed out that these evaluation parameters provide little or no information as to whether a parser assigns the correct semantic structure to a given input, if the set of category labels comprises only syntactic categories in the narrow sense, i.e. includes only names of lexical and phrasal categories. This justified criticism observes that a measure of semantic accuracy can only be obtained if the gold standard includes annotations of syntactic-semantic dependencies between bracketed constituents. It is to answer this criticism that the evaluation of the TüSBL system presented here focuses on the correct assignment of functional labels. For an in-depth evaluation that focuses on syntactic categories, we refer the interested reader to (Kübler and Hinrichs, 2001).

The quantitative evaluation of TüSBL has been conducted on the treebanks of German and English described in section 3. Each treebank uses a different annotation scheme at the level of function-argument structure. As shown in Table 1, the English treebank uses a total of 13 functional labels, while the German treebank has a richer set of 36 function labels.

The evaluation consisted of a ten-fold cross-validation test, where the training data provide an instance base of already seen cases for TüSBL's tree construction module. The evaluation was performed for both the German and English data. For each language, the following parameters were measured: 1. labeled precision for syntactic cat-

```
construct_tree(chunk_list, treebank):
    while (chunk_list is not empty) do
        remove first chunk from chunk_list
        process_chunk(chunk, treebank)
```

Figure 5: Pseudo-code for tree construction, main routine.

```
process_chunk(chunk, treebank):
    words := string_yield(chunk)
    tree := complete_match(words, treebank)
    if (tree is not empty)                                  direct hit,
    then output(tree)                                       i.e. complete chunk found in treebank
    else
        tree := partial_match(words, treebank)
        if (tree is not empty)
        then
            if (tree = postfix of chunk)
            then
                tree1 := attach_next_chunk(tree, treebank)
                if (tree is not empty)
                then tree := tree1
            if ((chunk - tree) is not empty)                if attach_next_chunk succeeded
            then tree := extend_tree(chunk - tree, tree, treebank)   chunk might consist of both chunks
            output(tree)
            if ((chunk - tree) is not empty)                chunk might consist of both chunks (s.a.)
            then process_chunk(chunk - tree, treebank)      i.e. process remaining chunk
        else                                                back off to POS sequence
            pos := pos_yield(chunk)
            tree := complete_match(pos, treebank)
            if (tree is not empty)
            then output(tree)
            else                                            back off to subchunks
                while (chunk is not empty) do
                    remove first subchunk c1 from chunk
                    process_chunk(c1, treebank)
```

Figure 6: Pseudo-code for tree construction, subroutine process_chunk.

```
attach_next_chunk(tree, treebank):                          attempts to attach the next chunk to the tree
    take first chunk chunk2 from chunk_list
    words2 := string_yield(tree, chunk2)
    tree2 := complete_match(words2, treebank)
    if (tree2 is not empty)
    then
        remove chunk2 from chunk_list
        return tree2
    else return empty
```

Figure 7: Pseudo-code for tree construction, subroutine attach_next_chunk.

```
extend_tree(rest_chunk, tree, treebank):                    extends the tree on basis of POS comparison
    words := string_yield(tree)
    rest_pos := pos_yield(rest_chunk)
    tree2 := partial_match(words + rest_pos, treebank)
    if ((tree2 is not empty) and (subtree(tree, tree2)))
    then return tree2
    else return empty
```

Figure 8: Pseudo-code for tree construction, subroutine extend_tree.

egories alone, and 2. labeled precision for functional labels.

The results of the quantitative evaluation are shown in Tables 2 and 3. The results for labeled recall underscore the difficulty of applying the classical PARSEVAL measures to a partial pars-

language	parameter	minimum	maximum	average
German	true positives	60.38 %	64.23 %	61.45 %
	false positives	2.93 %	3.14 %	3.03 %
	unattached constituents	15.15 %	19.23 %	18.18 %
	unmatched constituents	17.05 %	17.59 %	17.35 %
English	true positives	59.11 %	60.18 %	59.78 %
	false positives	3.11 %	3.39 %	3.25 %
	unattached constituents	9.57 %	10.30 %	9.88 %
	unmatched constituents	26.80 %	27.54 %	27.10 %

Table 2: Quantitative evaluation: recall.

language	parameter	minimum	maximum	average
German	labeled precision for synt. cat.	81.28 %	82.08 %	81.56 %
	labeled precision for funct. cat.	89.26 %	90.13 %	89.73 %
English	labeled precision for synt. cat.	66.15 %	67.34 %	66.84 %
	labeled precision for funct. cat.	90.07 %	90.93 %	90.40 %

Table 3: Quantitative evaluation: precision.

ing approach like ours. We have, therefore divided the incorrectly matched nodes into three categories: the genuine false positives where a tree structure is found that matches the gold standard, but is assigned the wrong label; nodes which, relative to the gold standard, remain unattached in the output tree; and nodes contained in the gold standard for which no match could be found in the parser output. Our approach follows a strategy of positing and attaching nodes only if sufficient evidence can be found in the instance base. Therefore the latter two categories cannot really be considered errors in the strict sense. Nevertheless, in future research we will attempt to significantly reduce the proportion of unattached and unmatched nodes by exploring matching algorithms that permit a higher level of generalization when matching the input against the instance base. What is encouraging about the recall results reported in Table 2 is that the parser produces genuine false positives for an average of only 3.03 % for German and 3.25 % for English.

For German, labeled precision for syntactic categories yielded 81.56 % correctness. While these results do not reach the performance reported for other parsers (cf. (Collins, 1999; Charniak, 1997)), it is important to note that the two treebanks consist of transliterated spontaneous speech data. The fragmentary and partially ill-formed nature of such spoken data makes them harder to analyze than written data such as the Penn treebank typically used as gold standard.

It should also be kept in mind that the basic PARSEVAL measures were developed for parsers that have as their main goal a complete analysis that spans the entire input. This runs counter to the basic philosophy underlying an amended chunk parser such as TüSBL, which has as its main goal robustness of partially analyzed structures.

Labeled precision of functional labels for the German data resulted in a score of 89.73 % correctness. For English, precision of functional labels was 90.40 %. The slightly lower correctness rate for German is a reflection of the larger set of function labels used by the grammar. This raises interesting more general issues about trade-offs in accuracy and granularity of functional annotations.

6 Conclusion and Future Research

The results of 89.73 % (German) and 90.40 % (English) correctly assigned functional labels validate the general approach. We anticipate further improvements by experimenting with more sophisticated similarity metrics[7] and by enriching the linguistic information in the instance base. The latter can, for example, be achieved by preserving more structural information contained in the chunk parse. Yet another dimension for experimentation concerns the way in which the algorithm generalizes over the instance base. In the current version of the algorithm, generalization heavily relies on lexical and part-of-speech information. However, a richer set of backing-off strategies that rely on larger domains of structure are easy to envisage and are likely to significantly improve recall performance.

While we intend to pursue all three dimensions of refining the basic algorithm reported here, we have to leave an experimentation of which modifications yield improved results to future research.

References

Steven Abney. 1991. Parsing by chunks. In Robert Berwick, Steven Abney, and Caroll Tenney, editors, *Principle-Based Parsing*. Kluwer Academic Publishers.

Steven Abney. 1996. Partial parsing via finite-state cascades. In John Carroll, editor, *Workshop on Robust Parsing (ESSLLI '96)*.

Rens Bod. 1998. *Beyond Grammar: An Experience-Based Theory of Language*. CSLI Publications, Stanford, California.

Rens Bod. 2000. Parsing with the shortest derivation. In *Proceedings of COLING 2000, Saarbrücken, Germany*.

Thorsten Brants, Wojiech Skut, and Brigitte Krenn. 1997. Tagging grammatical functions. In *Proceedings of EMNLP-2 1997, Providence, RI*.

Norbert Bröker, Udo Hahn, and Susanne Schacht. 1994. Concurrent lexicalized dependency parsing: the ParseTalk model. In *Proceedings of COLING 94, Kyoto, Japan*.

Eugene Charniak. 1997. Statistical parsing with a context-free grammar and word statistics. In *Proceedings of the Fourteenth National Conference on Artifical Intelligence, Menlo Park*.

Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

[7](Daelemans et al., 1999) reports that the *gain ratio* similarity metric has yielded excellent results for the NLP applications considered by these investigators.

Walter Daelemans, Jakub Zavrel, and Antal van den Bosch. 1999. Forgetting exceptions is harmful in language learning. *Machine Learning: Special Issue on Natural Language Learning*, 34.

Helmut Feldweg. 1993. Stochastische Wortartendisambiguierung für das Deutsche: Untersuchungen mit dem robusten System LIKELY. Technical report, Universität Tübingen. SfS-Report-08-93.

Valia Kordoni. 2000. Stylebook for the English Treebank in VERBMOBIL. Technical Report 241, Verbmobil.

Sandra Kübler and Erhard W. Hinrichs. 2001. TüSBL: A similarity-based chunk parser for robust syntactic processing. In *Proceedings of HLT 2001, San Diego, Cal.*

Leonardo Lesmo and Vincenzo Lombardo. 2000. Automatic assignment of grammatical relations. In *Proceedings of LREC 2000, Athens, Greece*.

Mitchell Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Anne Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. 1994. The Penn Treebank: Annotating predicate argument structure. In *Proceedings of HLT 94, Plainsboro, New Jersey*.

Beatrice Santorini. 1990. Part-Of-Speech Tagging Guidelines for the Penn Treebank Project. University of Pennsylvania, 3rd Revision, 2nd Printing.

Anne Schiller, Simone Teufel, and Christine Thielen. 1995. Guidelines für das Tagging deutscher Textkorpora mit STTS. Technical report, Universitäten Stuttgart and Tübingen. http://www.sfs.nphil.uni-tuebingen.de/Elwis/stts/stts.html.

Craig Stanfill and David L. Waltz. 1986. Towards memory-based reasoning. *Communications of the ACM*, 29(12).

Rosmary Stegmann, Heike Schulz, and Erhard W. Hinrichs. 2000. Stylebook for the German Treebank in VERBMOBIL. Technical Report 239, Verbmobil.

Pasi Tapanainen and Timo Järvinen. 1997. A non-projective dependency parser. In *Proceedings of ANLP'97, Washington, D.C.*

Jorn Veenstra, Antal van den Bosch, Sabine Buchholz, Walter Daelemans, and Jakub Zavrel. 2000. Memory-based word sense disambiguation. *Computers and the Humanities, Special Issue on Senseval, Word Sense Disambiguations*, 34.

Jakub Zavrel, Walter Daelemans, and Jorn Veenstra. 1997. Resolving PP attachment ambiguities with memory-based learning. In *Proceedings of CoNLL'97, Madrid, Spain*.