A Data-Driven Dependency Parser for Bulgarian

Svetoslav Marinov
University of Skövde and GSLT

Joakim Nivre
Växjö University and Uppsala University

1 Introduction

One of the main motivations for building treebanks is that they facilitate the development of syntactic parsers, by providing realistic data for evaluation as well as inductive learning. In this paper we present what we believe to be the first robust data-driven parser for Bulgarian, trained and evaluated on data from BulTreeBank (Simov et al., 2002). The parser uses dependency-based representations and employs a deterministic algorithm to construct dependency structures in a single pass over the input string, guided by a memory-based classifier at each nondeterministic choice point, as described in Nivre et al. (2004). Since the original BulTreeBank annotation is based on HPSG, it has been necessary to extract dependency structures from the original annotation both for training and evaluating the parser.

The paper is structured as follows. Section 2 introduces the MaltParser system used in the experiments to induce parsers from treebank data. Section 3 presents BulTreeBank, and section 4 shows how the BulTreeBank annotation can be converted into a dependency structure annotation of the kind required by MaltParser. Section 5 describes the experimental conditions, and section 6 discusses the results of the experiments. Section 7 contains our conclusions.

2 MaltParser

Given the open-ended nature of natural language, the problem of parsing unrestricted text is essentially an empirical problem, and the accuracy of a text parsing system can only be evaluated by comparing the analysis produced by the system to some kind of gold standard. The standard method for carrying out this kind of evaluation is to apply the system to a sample of text taken from a treebank. In the data-driven approach to text parsing, treebank data may be used also in the training corpus, i.e. in the sample of text used to construct the parser based on inductive inference. Using treebank data for training and evaluation is what we normally understand by the term *treebank parsing*, a methodology that has been used to construct robust and efficient parsers for several languages over the last ten to fifteen years, most notably for English (Magerman, 1994; Collins, 1997, 1999; Charniak, 2000) and Czech (Collins et al., 1999; McDonald et al., 2005), where large treebank resources have been readily available.

In this paper, we apply this methodology to Bulgarian using the MaltParser system (Nivre and Hall, 2005), which can be described as a data-driven parser-generator framework. While a traditional parser generator constructs a parser given a grammar, a data-driven parser generator constructs a parser given a treebank. MaltParser uses dependency-based representations, which means that the treebank used for training (and for evaluation) must be annotated with dependency structures. The system is based on a methodology of *inductive dependency parsing* (Nivre, 2005), involving three essential components:

1. Deterministic parsing algorithms for building dependency graphs (Yamada and Matsumoto, 2003; Nivre, 2003).

2. History-based feature models for predicting the next parser action (Black et al., 1992; Magerman, 1995; Ratnaparkhi, 1997).

3. Discriminative machine learning to induce the mapping from histories to parser actions (Yamada and Matsumoto, 2003; Nivre et al., 2004)

The architecture of the MaltParser system imposes a strict modularization of parsing algorithms, feature models and learning methods, in order to give maximum flexibility in the way these components can be varied independently of each other. In the experiments reported below, we explore several different feature models, but we fix the parsing algorithm to that described in Nivre (2003) and the learning algorithm to memory-based learning as implemented in the TiMBL software package (Daelemans and Van den Bosch, 2005). We restrict our further description of MaltParser to the specification of feature models, which is needed to understand the experiments described in sections 5–6, and refer the reader to Nivre (2005) for more information about inductive dependency parsing in general and to Nivre and Hall (2005) for more information about MaltParser in particular.[1]

Any deterministic parsing algorithm compatible with the MaltParser architecture has to provide two basic data structures as an interface to the feature model component:

1. A stack σ of partially processed tokens, where $\sigma[i]$ is the $i+1$th token from the top of the stack, with the top being $\sigma[0]$.

2. A list τ of remaining input tokens, where $\tau[i]$ is the $i+1$th token in the list, with the first token being $\tau[0]$.

[1]MaltParser is freely available for research and educational purposes and can be downloaded from http://www.msi.vxu.se/users/nivre/research/MaltParser.html.

Stack (σ)		Input (τ)			
$p(\sigma[1])$	$p(\sigma[0])$	$p(\tau[0])$	$p(\tau[1])$	$p(\tau[2])$	$p(\tau[3])$
	$d(\sigma[0])$	$d(lc(\tau[0]))$			
	$d(lc(\sigma[0]))$				
NL	$d(rc(\sigma[0]))$				
L	$w(\sigma[0])$	$w(\tau[0])$			
E	$w(h(\sigma[0]))$		$w(\tau[1])$		

Figure 1: Feature functions. σ = stack; τ = input; w = word form; p = part-of-speech; h = head; d = dependency type; lc = leftmost child; rc = rightmost child.

In terms of these structures, we first define *address functions* as follows:

1. For any position i, $\sigma[i]$ and $\tau[i]$ are address functions.
2. If α is an address function, then $h(\alpha)$, $lc(\alpha)$ and $rc(\alpha)$ are address functions, which identify the *head* (h), *leftmost child* (lc) and *rightmost child* (rc), respectively, of the token identified by α (in the partially built dependency structure).

Given address functions, we finally define *feature functions*:

> If α is an address function, then $p(\alpha)$, $d(\alpha)$ and $w(\alpha)$ are feature functions, which identify the part-of-speech (p), dependency type (d) and word form (w), respectively, of the token identified by α (where d is defined relative to the partially built dependency structure).

A feature model Φ is defined by a set of feature functions $\{\phi_1, \ldots, \phi_p\}$. Figure 1 depicts the feature functions, or simply *features*, used in the experiments below. The simplest model uses only features defined in terms of the functions p and d, which we call non-lexical (NL) and which occur above the topmost horizontal line in Figure 1, while the other models in addition use features defined in terms of w, called lexical features and appearing below the topmost horizontal line. Adding the two features between the two horizontal lines gives us the basic lexical model (L); adding all four lexical features gives us the enhanced lexical model (E). However, in the enhanced lexical model, full word forms are replaced by suffixes of six characters to counter the sparse data problem.

3 BulTreeBank

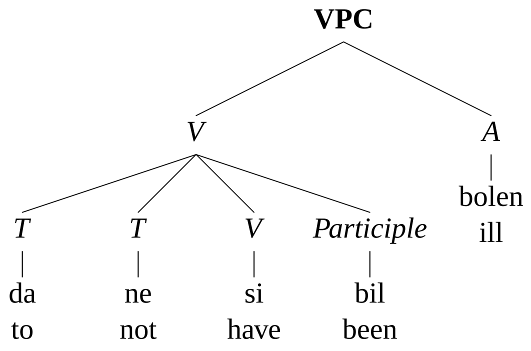
BulTreeBank (Simov et al., 2002; Osenova and Simov, 2003; Simov and Osenova, 2003) is a corpus of syntactically annotated sentences of Bulgarian. It is still of modest size and the version we work with at present contains 5080 sentences. These can be divided into two different groups: sentences extracted from grammars (1500) and sentences extracted from corpora of government documents, prose and newspapers (3580). The total number of tokens is 71703 (punctuation included), which gives a mean sentence length of 14.11 tokens.

The annotation scheme relies on basic ideas from HPSG and the analyses of the sentences comply with the HPSG formalism. Four types of elements are distinguished: *lexical elements* (e.g. V, N, Prep), *phrasal elements* (e.g. VPC, NPA), *functional elements* (e.g. Pragmatic, S, Conj) and *textual elements* (tokens). An underlying HPSG grammar, formalized as a DTD and constraints over XML documents, is employed in the actual annotation process, and *Head-Complement*, *Head-Adjunct* and *Head-Subject* relations are encoded in phrase labels of the form XPC, XPA and XPS, respectively. However, in its current incarnation BulTreeBank is not a complete HPSG treebank. The explicit marking of heads, for example, is missing in the actual analysis of a sentence. While this information can to a certain degree be derived from the accompanying DTD or the type of phrase, it is not present in the actual structure of the tree.

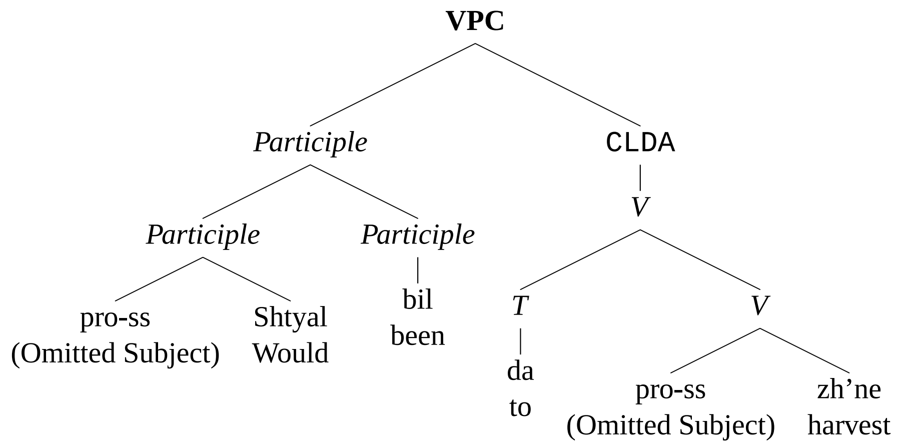
It should also be pointed out that the annotation encodes empty nodes, e.g. in connection with ellipsis and the pro-dop phenomenon in Bulgarian. Additional information about coreference, anaphora, and clitic doubling is also available as indices between nodes. This information has not been exploited in the present study, where we have concentrated on converting phrase structure trees to dependency structures, which can be used to create parsers using the MaltParser system described in the previous section.

In order to motivate our approach to treebank conversion, described in section 4, we conclude with a few examples from BulTreeBank where it is not straightforward to detect the syntactic head of a phrase. The phrasal categories are set in bold font, functional categories in typewriter font and lexical categories in italics (to distinguish them from textual elements); punctuation and coindexation have been omitted and glosses added:

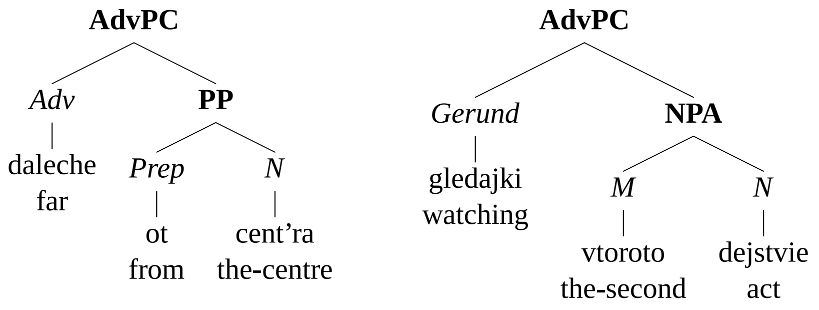
(1)



(2)



(3)



Although the label VPC in (1) indicates that the phrase has a *Head-Complement* structure, it is not explicitly marked that the V-child is the head and the A-child is the complement. And even though this can be inferred from the phrase type (VP is headed by V), the head-child is still a complex node consisting of four other lexical elements. According to the annotators the ultimate head should be the two rightmost daughters: V and Participle. Example (2) presents a choice between two equally plausible heads of the VPC, both having the same category (Participle). Example (3) shows that in the absence of an adverb, a gerund can serve as the head of an adverbial phrase.

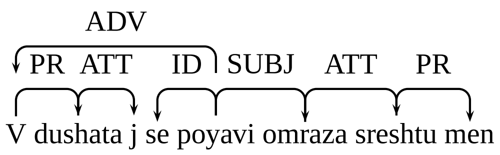
4 Treebank Conversion

Converting phrase structure trees into dependency trees is sometimes considered a trivial matter. Few authors discuss this and mainly with reference to English (Lin, 1995; Magerman, 1994; Collins, 1999; Xia, 2001; Xia and Palmer, 2001) and (to a lesser extent) German (Kübler and Telljohann, 2002; Bohnet, 2003; Ule and Kübler, 2004). Since heads are an important notion in dependency graphs, they also need to be explicitly marked. Phrase-structure trees on the other hand normally do not encode this information straightforwardly. Therefore, in order to convert from the one format to the other, we need a way to identify the head of a given phrase. The standard way to proceed is by using a so-called *head percolation* *table* (first mention by Magerman (1994, p. 66) as *Tree Head Table*). An entry in such a table is of the form (α *direction* $\beta|\gamma|\delta|$...). For example, (VP left-to-right VP|VB|VBG) means that the head of the VP-node is the first child of this node from the left, which carries a label VP, VB or VBG (see also (Collins, 1999, p. 147, 238–240) for a similar treatment of head percolation tables).

Bulgarian, as all other Slavic languages, exibits relatively free word order. This means that the head of a given phrase may precede its dependents (first element left-to-right), follow them (first element right-to-left) or occur in between them (left-to-right or right-to-left). In this respect specifying a direction in the head percolation table will not necessarily lead to a quicker or better discovery of heads. We have thus chosen a default one (left-to-right). The entries in our table look as follows:

Nomin	<I> <Prep> <N> <T> <Adv> <C>
NPA	<N> <M> <I> <Pron> <A> <Participle> <H> <Gerund> <V> <Adv>
S	<V> <Gerund> <Participle> <N> <M> <I> <Pron> <A> <H> <Adv>
VPA	<V> <Gerund> <Participle> <N> <M> <I> <Pron> <A> <H> <Adv>
VPC	<V> <Gerund> <Participle> <N> <M> <I> <Pron> <A> <H> <Adv>

In creating the head percolation table we relied mainly on the DTD accompanying the treebank data and the technical reports with additional fine-tunings during the actual conversion process. The first column in this table is the node whose head we search for, the second is an ordered list of possible heads for this node. Given a node and a list of its children the algorithm searches first for the first element in the list. If it cannot find such a node, it then searches for the second element, etc. Once a head-child is found it is considered the head of the node. The mother node is then substituted by its head. The following is an example of a dependency tree for a sentence from the BulTreeBank (punctuation omitted).

(4)


In addition to the head percolation table we need a dependency table in order to assign proper labels to the already discovered unlabeled arcs. In our case it looks like this:[2]

<Prep>from<PP><N>from<N>	PR
<N>from<N><Pron>	ATT
<V>from<VPS><Pron>	ID
<V>from<VPS><N>from<NPA>	SUBJ
<V>from<VPA><Prep>from<PP>	ADV

[2]All labels of the heads and phrases are according to the BulTreeBank notation.

Table 1: Dependency types

ADV	Adverbial modifier
APP	Apposition
ATT	Attribute (adnominal modifier)
CC	Coordination (conjunction or second conjunct)
DET	Determiner
ID	Non-first element of multi-word expression (idiom)
IP	Punctuation
OBJ	Object
PR	Complement of preposition
PRD	Predicative complement
SUBJ	Subject
UK	Head-verb of subordinate clause dependent on complementizer
VC	Verb chain (non-finite verb dependent on other verb)
ROOT	Dependent of special root node

In this table, the first column represents the head and dependent of an arc (e.g. <`V`>`from`<`VPS`><`N`>`from`<`NPA`> means that the head <`V`> is the head of a <`VPS`> phrase, while the dependent <`N`> is the head of an <`NPA`> phrase), and the second column is a label for that relation (e.g. SUBJ – subject relation). In creating the dependency table, we relied heavily on information in the phrasal categories, which encode information about *Head-Subject*, *Head-Complement* and *Head-Adjuct* relations. Altogether, fourteen different dependency types were used as labels, given in Table 1.

In addition to the basic tree conversion a simple post-processing of the data was necessary. In the phrase structure trees of BulTreeBank there is no distinction in the nodes between reflexive pronouns and argument clitic pronouns. An additional program changed the labels of the argument clitics from ID (the default one according to the dependency table) to OBJ.

Besides the tree conversion a simplification of the BulTreeBank part-of-speech tagset was performed. The tags in BulTreeBank can be up to twelve characters long and carry a lot of morphological information. Not all of this information is relevant for syntactic parsing and the fine-grainedness of the tagset may hurt performance because of data sparseness. We therefore collapsed all tags to only two characters. Decisions about which morphological information to keep was based on experiments done with the Prague Dependency Treebank (Hajič, 1998; Hajič et al., 2001), as described in (Collins et al., 1999), as well as some personal intuition of one of the authors.[3] This compression resulted in a tagset with 51 part-of-speech tags.

[3]Whether the resulting tags are optimal is left for future research.

5 Experimental Setup

The data we work with consist of 5080 sentences, split in eight different files. Each of these contains sentences from different domains and/or sources. We have therefore used this split for an eight-fold cross-validation of the parser, training on seven of the files and testing on the remaining one.

The experiments have been performed under two different conditions. In the first condition, we used the gold standard part-of-speech tags provided with the treebank (in their compressed version) both for training and testing of the parsers. In the second condition, we trained on the same data but tested on data that had been tagged automatically using a statistical tagger (Hall, 2003), trained on the seven files that were used for training in that fold. The mean accuracy of the tagger over the eight folds was 93.5% for the 51 part-of-speech tags.

As indicated in section 2, we have used three different feature models in the experiments, which we refer to as the *non-lexical* model, the (basic) *lexical* model, and the *enhanced* (lexical) model, respectively. The non-lexical model only incorporates the nonlexical features specified in Figure 1 (above the topmost horizontal line), while the lexical model also includes the two lexical features between the two horizontal lines. The enhanced model, finally, adds two more lexical features (below the second horizontal line) but also differs from the basic lexical model by using a suffix of (up to) six characters instead of the full word form as the value for all lexical features.

For all feature models, the parameter settings for the TiMBL learner used to predict the next parser actions involved setting the number k of nearest distances to 5 and using the Modified Value Difference Metric (MVDM) together with Inverse Distance (ID) weighted class voting. (For more information about these parameters, see Daelemans and Van den Bosch, 2005.)

6 Results and Discussion

The results of our experiments are given in Tables 2 and 3. They show how the three different feature models fare, evaluated with respect to unlabeled as well as labeled dependency accuracy. Unlabeled dependency accuracy only considers whether each dependendent is assigned the correct head, while labeled dependency accuracy also takes the dependency type label into account. The results, which in all cases are mean scores based on an eight-fold cross-validation, are presented as the mean accuracy per sentence and as the mean accuracy per word/dependency. In all scores, punctuation tokens have been omitted from the counts.

Although there are no strictly comparable results available for Bulgarian, the parsing accuracy obtained is similar to that reported with the same parsing methodology for Czech (Nivre and Nilsson, 2005), but lower than the corresponding results for Swedish (Nivre et al., 2004) and English (Nivre and Scholz, 2004). Tanev and Mitkov (2002) report precision and recall in the low 60's for a rule-based parser

Table 2: Accuracy on gold standard part-of-speech tags

	Per sentence		Per word/dependency	
	Unlabeled	Labeled	Unlabeled	Labeled
Non-lexical	83.3%	74.8%	81.9%	73.7%
Lexical	85.6%	79.2%	84.2%	78.2%
Enhanced	85.6%	79.1%	84.2%	78.0%

Table 3: Accuracy on automatically assigned part-of-speech tags

	Per sentence		Per word/dependency	
	Unlabeled	Labeled	Unlabeled	Labeled
Non-lexical	78.8%	68.8%	77.5%	68.1%
Lexical	81.0%	73.2%	79.7%	72.2%
Enhanced	81.7%	73.8%	80.4%	72.9%

of Bulgarian, which seems to indicate that our performance is competitive, even though the results of the two studies cannot be compared directly.

Out of the fourteen dependency relations which the parser had to assign to a sentence, six showed relatively low scores. These were the coordination relation, adverbials, punctuation marks, verb chain, main verb of a subordinate clause dependent on the complementizer and apposition. Of these relations, apposition, coordination and adverbials were hardest for the parser to learn, sometimes getting a score of less than 50%. Another problematic issue is that projectivity can be hard to maintain with free word order languages. An example where the parser fails to assign proper subject and verb chain relations is *da*-clauses, as shown below:

(5)	... i	vmesto	tya	da	se	kazva	Genka Ginkova ...
	... and	instead	she	to	refl	be-named	Genka Ginkova ...

“... and instead of her name to be Genka Ginkova ...”

Here *tya* is the subject dependent on the verb *kazva*, which in itself is dependent on *da*, the complementizer, which is dependent on *vmesto*.

On the positive side, the parser generally has high accuracy on core grammatical functions such as subject (SUBJ), object (OBJ), predicative complement (PRD) and root (ROOT), where unlabeled dependency accuracy is around 85% or better on automatically assigned part-of-speech tags. Table 4 in addition shows the labeled precision and recall, which can be seen to be considerably lower, especially for the nominal argument types SUBJ and OBJ, a result that is probably related to the relatively free word order in Bulgarian.

Table 4: Accuracy for selected dependency types (enhanced model, automatically assigned part-of-speech tags)

	Unlabeled	Labeled	
	Accuracy	Precision	Recall
ROOT	86.8%	66.7%	86.8%
SUBJ	83.1%	68.7%	67.3%
OBJ	85.3%	66.7%	70.4%
PRD	89.0%	64.7%	74.6%

7 Conclusion

In this paper we have presented the first robust data-driven dependency parser for Bulgarian, trained and evaluated on dependency structures extracted from BulTreeBank. Although the amount of training data is still rather limited, the parsing accuracy obtained is similar to that reported for other languages with the same parsing methodology. In addition, we have shown how to extract dependency structures from the phrase structure annotation of the original treebank. With respect to the last point, we see two directions for future research. One is to create specific dependency labels for Bulgarian and the other is to improve the conversion procedure, taking into account additional information present in the original format (e.g. coindexation, marking of discontinuous elements and non-immediate dominance).

References

Black, E., F. Jelinek, J. Lafferty, D. Magerman, R. Mercer, and S. Roukos. 1992. Towards history-based grammars: Using richer models for probabilistic parsing. In *Proceedings of the 5th DARPA Speech and Natural Language Workshop*, 31–37.

Bohnet, B. 2003. Mapping Phrase Structures to Dependency structures in the Case of Free Word Order Languages. In *Proceedings of The First International Conference on Meaning-Text Theory*, 138–148.

Charniak, Eugene. 2000. A Maximum-Entropy-Inspired Parser. In *Proceedings of the 1st NAACL*, 132–139.

Collins, Michael. 1997. Three Generative, Lexicalized Models for Statistical Parsing. In *Proceedings of the 35th ACL*.

Collins, Michael. 1999. Head-Driven Statistical Models for Natural Language Parsing. Doctoral Dissertation, University of Pennsylvania.

Collins, Michael, Jan Hajič, Lance Ramshaw, and Christoph Tillman. 1999. A statistical parser for czech. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL)*, 505–512.

Daelemans, Walter, and Antal Van den Bosch. 2005. *Memory-based language processing*. Cambridge University Press.

Hajič, Jan. 1998. Building a Syntactically Annotated Corpus: The Prague Dependency Treebank. In *Issues of Valency and Meaning. Studies in Honor of Jarmila Panevová*, ed. Eva Hajičová, 12–19. Prague Karolinum, Charles University Press.

Hajič, Jan, Barbora Vidova Hladka, Jarmila Panevová, Eva Hajičová, Petr Sgall, and Petr Pajas. 2001. Prague Dependency Treebank 1.0. LDC, 2001T10.

Hall, Johan. 2003. A Probabilistic Part-of-Speech Tagger with Suffix Probabilities. Master's thesis, School of Mathematics and Systems Engineering, Växjö University.

Kübler, Sandra, and Heike Telljohann. 2002. Towards a dependency-based evaluation for partial parsing. In *Proceedings of the LREC-Workshop Beyond PARSEVAL – Towards Improved Evaluation Measures for Parsing Systems*, 9–16.

Lin, Dekang. 1995. A Dependency-based Method for Evaluating Broad-Coverage Parsers. In *Proceedings of IJCAI-95*.

Magerman, David M. 1994. Natural Language Parsing as Statistical Pattern Recognition. Doctoral Dissertation, Stanford University.

Magerman, David M. 1995. Statistical Decision-Tree Models for Parsing. In *Proceedings of the 33rd ACL*.

McDonald, Ryan, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, 91–98.

Nivre, Joakim. 2003. An efficient algorithm for projective dependency parsing. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT)*, ed. Gertjan Van Noord, 149–160.

Nivre, Joakim. 2005. Inductive Dependency Parsing. Doctoral Dissertation, Växjö University.

Nivre, Joakim, and Johan Hall. 2005. MaltParser: A language-independent system for data-driven dependency parsing. In *Proceedings of the Fourth Workshop on Treebanks and Linguistic Theory (TLT)*.

Nivre, Joakim, Johan Hall, and Jens Nilsson. 2004. Memory-based Dependency Parsing. In *Proceedings of the Eighth Conference on Computational Natural Language Learning (CoNLL)*, ed. H. T. Ng and E. Riloff, pp. 49–56. Boston, Massachusetts.

Nivre, Joakim, and Jens Nilsson. 2005. Pseudo-Projective Dependency Parsing. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*.

Nivre, Joakim, and Mario Scholz. 2004. Deterministic Dependency Parsing of English Text. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING)*, 64–70.

Osenova, Petya, and Kiril Simov. 2003. The Bulgarian HPSG Treebank: Specialization of the Annotation Scheme. In *Proceedings of The Second Workshop on Treebanks and Linguistic Theories (TLT2003)*. Växjö, Sweden.

Ratnaparkhi, Adwait. 1997. A linear observed time statistical parser based on maximum entropy models. In *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1–10.

Simov, Kiril, and Petya Osenova. 2003. Practical Annotation Scheme for an HPSG Treebank of Bulgarian. In *Proceedings of the 4th International Workshop on Linguistically Interpreteted Corpora (LINC-2003)*. Budapest, Hungary.

Simov, Kiril, Gergana Popova, and Petya Osenova. 2002. HPSG-based syntactic treebank of Bulgarian (BulTreeBank). In *A Rainbow of Corpora: Corpus Linguistics and the Languages of the World*, ed. Andrew Wilson, Paul Rayson, and Tony McEnery, 135–142. Lincon-Europa, Munich.

Tanev, Hristo, and Ruslan Mitkov. 2002. Shallow Language Processing Architecture for Bulgarian. In *Proceedings of the 17th International Conference on Computational Linguistics (COLING)*.

Ule, Tylman, and Sandra Kübler. 2004. From phrase structure to dependencies, and back. In *Proceedings of the International Conference on Linguistic Evidence*, 169–170.

Xia, Fei. 2001. Automatic Grammar Generation from Two Different Perspectives. Doctoral Dissertation, University of Pennsylvania.

Xia, Fei, and Martha Palmer. 2001. Converting Dependency Structures to Phrase Structures. In *Proceedings of the 1st Human Language Technology Conference (HLT-2001)*. San Diego.

Yamada, Hiroyasu, and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT)*, ed. Gertjan Van Noord, 195–206.