

MaltParser: A Language-Independent System for Data-Driven Dependency Parsing

Joakim Nivre and Johan Hall

Växjö University
School of Mathematics and Systems Engineering
E-mail: {nivre, jha}@msi.vxu.se

1 Introduction

One of the potential advantages of data-driven approaches to natural language processing is that they can be ported to new languages, provided that the necessary linguistic data resources are available. In practice, this advantage can be hard to realize if models are overfitted to a particular language or linguistic annotation scheme. Thus, using two state-of-the-art statistical parsers developed for English to parse Italian, Corazza et al. [6] report an increase in error rate of 15–18%, and similar results have been reported for other languages [5, 1, 9, 14]. To overcome this kind of problem, systems for data-driven processing need to be designed for flexible reconfiguration, not only with respect to the selection of linguistic features but also with respect to algorithms and learning methods.

In this paper, we present a data-driven approach to dependency parsing that has been applied to several languages, consistently giving a dependency accuracy of 80–90% without any language-specific enhancements and with fairly modest data resources. The methodology is based on three essential components:

1. Deterministic parsing algorithms for building dependency graphs [26, 17]
2. History-based feature models for predicting the next parser action [2]
3. Discriminative machine learning to map histories to parser actions [26, 20]

Given the restrictions imposed by these components, we propose an architecture with a strict modularization of parsing algorithms, feature models and learning methods. This architecture has been realized in the MaltParser system, which can

be applied to a labeled dependency treebank in order to induce a labeled dependency parser for the language represented by the treebank.¹ In order to demonstrate the generality of the approach, we present an experimental evaluation of parsers trained on treebank data from five different languages, each with a different annotation scheme. But first we introduce the framework of *inductive dependency parsing* [19] and its realization in the MaltParser system.

2 Inductive Dependency Parsing

Given a set R of dependency types, we define a *dependency graph* for a sentence $x = (w_1, \dots, w_n)$ to be a labeled directed graph $G = (V, E, L)$, where V is the set of input tokens w_1, \dots, w_n , extended with a special root node w_0 and ordered by a linear precedence relation $<$; $E \subseteq V \times V$ is a set of directed arcs; and $L : E \rightarrow R$ is a function that labels arcs with dependency types. A dependency graph G is *well-formed* iff (i) the node w_0 is a root of G , (ii) G is connected, (iii) every node in G has an indegree of at most 1, and (iv) G is acyclic. Dependency parsing is the task of mapping sentences to well-formed dependency graphs.

Inductive approaches to natural language parsing can in general be defined in terms of three essential components:

1. A formal model M defining permissible representations for sentences (such as the model of dependency graphs defined above).
2. A parameterized stochastic model M_Θ , defining a score $S(x, y)$ for every sentence x and well-formed representation y .
3. An inductive learning algorithm L for estimating the parameters Θ from a representative sample $T_t = (x_1 : y_1, \dots, x_n : y_n)$ of sentences with their correct representations (normally a treebank sample).

Inductive dependency parsing is compatible with a variety of different models, but we focus here on *history-based* models, which can be defined in three steps:

1. Define a one-to-one mapping between syntactic representations y and decision sequences $D = (d_1, \dots, d_m)$ such that D uniquely determines y .
2. Define the score $S(x, y)$ in terms of each decision d_i in the decision sequence $D = (d_1, \dots, d_m)$, conditioned on the history $H = (d_1, \dots, d_{i-1})$.
3. Define a function Φ that groups histories into equivalence classes, thereby reducing the number of parameters in Θ .

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

In a conditional history-based model, the score $S(x, y)$ defined by the model is the conditional probability $P(y | x)$ of the analysis y given the sentence x , which means that the input sentence is a conditioning variable for each decision in the decision sequence:

$$P(y | x) = P(d_1, \dots, d_m | x) = \prod_{i=1}^m P(d_i | \Phi(d_1, \dots, d_{i-1}, x)) \quad (1)$$

The parameters of this model are the conditional probabilities $P(d | \Phi(H, x))$, for all possible decisions d and non-equivalent conditions $\Phi(H, x)$.

Given a conditional history-based model, the conditional probability $P(y_j | x)$ of analysis y_j given input x can be used to rank a set of alternative analyses $\{y_1, \dots, y_k\}$ of the input sentence x , derived by a nondeterministic parser. If the model allows a complete search of the analysis space, we can in this way be sure to find the analysis y_j that maximizes the probability $P(y_j | x)$ according to the model. With a deterministic parsing strategy, we instead try to find the most probable analysis y_j without exploring more than one decision sequence, based on the following approximation:

$$\max_j P(y_j | x) \approx \prod_{i=1}^m \max_i P(d_i | \Phi(d_1, \dots, d_{i-1}, x)) \quad (2)$$

A deterministic parsing strategy is in this context a *greedy algorithm*, making a locally optimal choice in the hope that this will lead to a globally optimal solution. The main advantage of the greedy strategy is that it improves parsing efficiency by avoiding an exhaustive search of the analysis space, but an additional advantage is that it reduces the effective number of parameters of the stochastic model, since only the mode of the distribution $P(d_i | \Phi(H, x))$ needs to be estimated for each distinct condition $\Phi(H, x)$. This also means that a larger class of learning methods can be used, including purely discriminative methods.

With a discriminative learning method we can reduce the learning problem to a pure classification problem, where an *input instance* is a parameterized history $\Phi(H, x)$ and an *output class* is a decision d . Using a supervised learning method, our task is then to induce a classifier C given a set of training instances D_t , derived from a treebank sample:

$$D_t = \{(\Phi(H, x), d) | O(H, x) = d, x \in T_t\} \quad (3)$$

where O is an oracle function that predicts the correct decision given the gold standard treebank.

In order to construct a specific instance of the inductive dependency parser, we therefore need to specify three things:

1. A deterministic *parsing algorithm* used to derive dependency graphs, which defines the set \mathcal{D} of permissible decisions, as well as the oracle function O that determines the correct decision given a certain history H and input sentence x . Moreover, we require that every history can be characterized by a stack σ of partially processed tokens, a list τ of remaining input tokens, and a partially built dependency graph G .
2. A parameterization function Φ used to define equivalence classes of histories and sentences in terms of a feature vector $\Phi_{(1,p)} = (\phi_1, \dots, \phi_p)$, where each feature ϕ_i is a function that maps a token to its part-of-speech, lexical form or dependency type (in the partially built dependency graph). We call $\Phi_{(1,p)}$ a *feature model*.
3. A discriminative *learning algorithm* used to approximate the mode function $f(\Phi(H, x)) = \arg \max_d P(d | \Phi(H, x))$ given a set D_t of training instances.

We will now turn to the description of a system that allows the user to construct such an instance in an efficient and flexible manner, given a suitable sample of a dependency treebank.

3 MaltParser

The architecture of the MaltParser system has been designed with two goals in mind. The first is to make it possible to flexibly combine different parsing algorithms, feature models and learning algorithms, given the constraints outlined in the preceding section. The second is to maximize the reuse of components across the *learning phase* and the *parsing phase*. In the learning phase a treebank is used to instantiate the parameters of the feature model; in the parsing phase the instantiated model is used to parse new data. Although these two phases have a different structure, they involve very similar or even identical subphases, which means that the same basic components can be reused across phases. The architecture therefore consists of three modularized components (in addition to input/output modules and overall control structure): *Parser*, *Guide* and *Learner*.

3.1 Parser

The Parser is responsible for the derivation of a dependency graph G , using a deterministic parsing algorithm that can be broken down into a sequence of decisions D . The learning phase requires that each parsing algorithm provides an implementation of the oracle function O , used for learning the correct sequence of decisions

for a given input. During the parsing phase the Parser only implements the decisions provided by the Learner (via the Guide), but the implementation is performed in exactly the same way during both learning and parsing. In this way, the parsing algorithm only defines what the permissible decisions are, not what the relevant linguistic features are, nor which learning method should be used. In the current version of MaltParser, the user can choose between several deterministic parsing algorithms, including the algorithms described by Nivre [18] and two variants of the incremental parsing algorithms described by Covington [7].

3.2 Guide

The Guide is responsible for constructing a set of training instances D_t for the Learner during the learning phase, and for passing the Learner’s predictions to the Parser during the parsing phase. At learning time, the Guide constructs one training instance $(\Phi(H, x), d)$ for each decision d passed from the Parser, where $\Phi(H, x)$ is the current vector of feature values (given the parameterization function Φ and the current state of the system), and passes this on to the Learner. At parsing time, the trainer constructs a feature vector $\Phi(H, x)$ for each request from the Parser, sends it to the Learner and passes on the predicted decision d from the Learner to the Parser. In this way, the feature model is completely separated from the Parser, and the Learner only has to learn a mapping from feature vectors to decisions, without knowing either how the features are extracted or how the decisions are to be used.

The feature extraction uses the parameterization function Φ , which is defined in terms of a feature vector $\Phi_{(1,p)}$, where each feature ϕ_i is a function, defined in terms of two simpler functions: an *address function* a_{ϕ_i} , which identifies a specific token in a given parser configuration, and an *attribute function* f_{ϕ_i} , which picks out a specific attribute of the token:

1. For every i , $i \geq 0$, $\sigma[i]$ and $\tau[i]$ are address functions identifying the $i+1$ th token from the top of the stack σ and the start of the input list τ , respectively. (Hence, $\sigma[0]$ is the top of the stack and $\tau[0]$ is the next input token.)
2. If α is an address function, then $h(\alpha)$, $lc(\alpha)$ and $rc(\alpha)$ are address functions, identifying the head (h), the leftmost child (lc) and the rightmost child (rc), respectively, of the token identified by α .
3. If α is an address function, then $p(\alpha)$, $w(\alpha)$ and $d(\alpha)$ are feature functions, identifying the part-of-speech (p), word form (w) and dependency type (d) of the token identified by α . We call p , w and d attribute functions.

The MaltParser system comes with a formal specification language for feature functions, which enables the user to define arbitrarily complex feature models in terms of address functions and attribute functions.

3.3 Learner

The Learner, finally, is responsible for inducing a model from the set of training instances D_t created by the Guide during the learning phase, and for using this model to predict parser decisions during the parsing phase. In practice, the Learner will normally be an interface to a standard machine learning package. The interface prepares the set of training instances for the specific package and invokes the appropriate functions to learn a model or predict a decision. Currently, MaltParser supports memory-based learning, using TIMBL [8], and support vector machines, using the LIBSVM tools for multiclass classification [25].

4 Treebank Parsing

In this section, we summarize experiments with the MaltParser system on data from five languages: Swedish, English, Czech, Danish, and Bulgarian.² Although these languages are all Indo-European, they nevertheless represent fairly different language types, ranging from English, with very reduced morphology and relatively inflexible word order, to the Slavic languages Czech and Bulgarian, with rich morphology and flexible word order, with the Scandinavian languages Swedish and Danish somewhere in the middle. In addition, the treebank annotation schemes used to analyze these languages differ considerably. Whereas the treebanks for Czech and Danish are proper dependency treebanks, albeit couched in different theoretical frameworks, the annotation schemes for the remaining treebanks are based on constituency in combination with grammatical functions, which necessitates a conversion from constituency trees to dependency structures. Below we first describe the five treebanks used in the experiments, including necessary conversions. We then describe the experimental setup, report the results for different languages, and relate the results to the state of the art for languages where such comparisons are possible.

4.1 Treebank Data

The Swedish data come from Talbanken [10], a manually annotated corpus of both written and spoken Swedish, created at Lund University in the 1970s. We use the professional prose section, consisting of about 100K words of text taken from textbooks, newspapers and information brochures. Although the original annotation

²Results have been published previously for Swedish [20, 19], English [22, 19] and Czech [21], but not for Danish and Bulgarian. (The results for Bulgarian are also published in a separate paper in this volume [15].) There is also ongoing work applying the system to Chinese, German, Italian and Turkish.

scheme is an eclectic combination of constituent structure, dependency structure, and topological fields [24], it has proven possible to convert the annotated sentences to dependency graphs with very high accuracy. In the conversion process, we have reduced the original fine-grained classification of grammatical functions to a more restricted set of 17 dependency types, mainly corresponding to traditional grammatical functions such as *subject*, *object* and *adverbial*.

The data set used for English is the standard data set from the 1M word Wall Street Journal section of the Penn Treebank, with sections 2–21 used for training and section 23 for testing. The data has been converted to dependency trees using the head percolation table of Yamada and Matsumoto [26], and dependency type labels have been inferred using a variation of the scheme employed by Collins [4], which makes use of the nonterminal labels on the head daughter, non-head daughter and parent corresponding to a given dependency relation. However, instead of simply concatenating these labels, as in the Collins scheme, we use a set of rules to map these complex categories onto a set of 10 dependency types, again including traditional grammatical functions such as *subject*, *object*, etc. More details about the conversion can be found in Nivre [19].

The Prague Dependency Treebank (PDT) consists of 1.5M words of newspaper text, annotated on three levels, the morphological, analytical and tectogrammatical levels [11]. Our experiments all concern the analytical annotation, which uses a set of 28 surface-oriented grammatical functions [3]. Unlike the treebanks of Swedish and English, PDT is a genuine dependency treebank also including non-projective dependencies.

The Danish Dependency Treebank (DDT) comprises about 100K words of text selected from the Danish PAROLE corpus, with annotation of primary and secondary dependencies [13]. Our experiments only concern primary dependencies, which are annotated using a fine-grained set of 48 dependency types. Like PDT, DDT is a proper dependency treebank and includes non-projective dependencies.

The BulTreeBank [23] contains about 70K words of Bulgarian text from different sources, annotated with constituency structure. Although the annotation scheme is meant to be compatible with the framework of HPSG, syntactic heads are not explicitly annotated, which means that the treebank must be converted to dependency structures using the same kind of head percolation tables and inference rules used for the English data. The set of dependency types used for Bulgarian is modeled after the Swedish set and includes 14 distinct grammatical functions. More details about the conversion can be found in Marinov and Nivre [15].

Table 1 gives an overview of the data sets for the five languages, in terms of annotation scheme, projectivity, tagset size for dependency types and parts-of-speech, and number of words and sentences. The latter figures refer in each case to the complete treebank, of which at most 90% has been used for training and at

Table 1: Data sets. AS = Annotation scheme (C = Constituency, D = Dependency, G = Grammatical functions); Pro = Projective; #D = Number of dependency types; #P = Number of PoS tags; #W = Number of words; #S = Number of sentences; W/S = Mean sentence length; MS = Model selection (DTS = Development test set, CV_n = N-fold cross-validation); MA = Model assessment on independent test set

Language	AS	Pro	#D	#P	#W	#S	W/S	MS	MA
Swedish	C+G	yes	17	46	97623	6316	15.46	CV ₉	yes
English	C+G	yes	10	48	1173766	49208	23.85	DTS	yes
Czech	D	no	26	28	1507333	87914	17.15	DTS	yes
Danish	D	no	54	33	100238	5512	18.19	DTS	no
Bulgarian	C	yes	14	51	71703	5080	14.11	CV ₈	no

least 10% for testing. It should also be pointed out that the tagset size for parts-of-speech refers to the tagset actually used in parsing, which (except in the case of English) is a reduced version of the complete tagset used in the annotation.

4.2 Experimental Setup

All the experiments reported in this paper were performed with the parsing algorithm described in Nivre [17, 18, 19] and with memory-based learning and classification as implemented in the TIMBL software package by Daelemans and Van den Bosch [8]. A variety of feature models were tested, but we only report results for the optimal model, averaged over all languages, which combines part-of-speech features, dependency features and lexical features in the way depicted in Figure 1.³ Optimal parameter settings for the TIMBL learner 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 [8].)

Model selection was performed using a single development test set for larger data sets and cross-validation for smaller ones, as summarized in Table 1. For all languages except Danish and Bulgarian, the final results reported were obtained using a separate test set for model assessment after completing the model selection (cf. Table 1). The part-of-speech tagger used for preprocessing is a standard HMM tagger with suffix smoothing developed by Hall [12], normally trained on the training part of the treebank used in the experiment,⁴ except for Czech where we use

³For Bulgarian, full word forms were replaced by suffixes of six characters as values for lexical features in order to counter the sparse data problem.

⁴For Danish, the tagger was trained on a larger corpus containing the training set as a subset,

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

Figure 1: Optimal feature model. w = word form; p = part-of-speech; h = head; d = dependency type; lc = leftmost child; rc = rightmost child.

the HMM tagging provided with the distribution of the treebank [11]. The tagging accuracy is included in the presentation of results below.

For the datasets that include non-projective dependencies (Czech and Danish), training data were projectivized prior to training using the procedure described in Nivre and Nilsson [21]. The output of the parser was then deprojectivized by an inverse graph transformation also described in Nivre and Nilsson [21]. The software needed for these transformations is not included in the MaltParser system as such but is freely available from the same location.

4.3 Results and Discussion

Table 2 reports the parsing accuracy obtained when applying MaltParser to each of the five languages under the conditions described in previous sections, sorted in order of decreasing accuracy. Unlabeled attachment score (UAS) is the proportion of words that are assigned the correct head (but possibly an incorrect dependency type), while labeled attachment score (LAS) is the proportion of words that are assigned both the correct head and the correct dependency type. Tagging accuracy (TA) is the proportion of words that are assigned the correct part-of-speech tag in preprocessing. Punctuation tokens are excluded in parsing accuracy results (LAS, UAS) but included in tagging accuracy results (TA).

Although MaltParser achieves an unlabeled attachment score above 80% for all languages, there is also a considerable range of variation, which seems to correlate fairly well with the linguistic dimensions morphological richness and word order flexibility, with the highest accuracy for English, the lowest accuracy for Bulgarian and Czech, and with Swedish and Danish in an intermediate position. The influence of these typological factors is discernible also in the tagging accuracy, which is lower for Bulgarian and Czech than for the other languages, a circumstance that

while for Swedish the tagger was trained on a completely separate corpus.

Table 2: Parsing accuracy. UAS = unlabeled attachment score; LAS = labeled attachment score; TA = tagging accuracy

Language	UAS	LAS	TA
English	88.1	86.3	96.1
Swedish	86.3	82.0	95.6
Danish	85.6	79.8	96.3
Bulgarian	80.4	72.9	93.5
Czech	80.1	72.8	94.1

in itself may lead to lower parsing accuracy.

The results for labeled attachment score reflect the same overall tendency but are also influenced by the complexity of the dependency type classification. By and large, the gap between labeled and unlabeled accuracy grows with the number of distinct dependency types. For example, while the unlabeled attachment score is roughly equivalent for Swedish and Danish, the higher labeled attachment score for Swedish can probably be explained by the smaller number of dependency types distinguished. The only exception to this generalization is Bulgarian, which has the widest gap between labeled and unlabeled accuracy despite a relatively small set of dependency types. This is probably due to the limited amount of data available for Bulgarian.

In comparison to structural properties of the languages involved, the size of the training set seems to be a very weak predictor of parsing accuracy, given that the two languages with the largest treebanks are ranked first (English) and last (Czech) and given that the two structurally similar languages Bulgarian and Czech have very similar results despite the large difference in available data resources. It is likely that the labeled parsing accuracy is more sensitive to sparse data, as indicated both by the very high labeled attachment score of English and the rather low score for Bulgarian, but on the whole it seems that the approach implemented in the MaltParser system is relatively robust in the face of sparse data, giving reasonable performance for a wide range of languages with datasets in the order of 100K words or less. Needless to say, a more detailed error analysis will be needed before we can draw any reliable conclusions about the influence of different factors, so the tentative conclusions advanced here are best regarded as conjectures to be corroborated or refuted by future research.

For English and Czech, the unlabeled attachment scores obtained are within a 5% increase in error rate compared to the state of the art, which is about 92% for English and 84% for Czech [16]. For Swedish, Danish and Bulgarian there

are no comparable results in the literature, which makes it difficult to assess the accuracy in absolute terms. However, given the fact that unlabeled attachment score is consistently above 80%, it seems that the parsing methodology is relatively robust to differences in language typology as well as in annotation schemes.

5 Conclusion

We have presented a data-driven system for dependency parsing that appears to give good parsing accuracy for a wide range of languages without language-specific enhancements and with relatively modest requirements on the quantity of data available. Unlabeled dependency accuracy is consistently above 80%, regardless of annotation scheme and training set size, and parsing accuracy remains within a 5% margin from the best performing parsers where comparative results are available.

References

- [1] D. Bikel and D. Chiang. Two statistical parsing models applied to the Chinese treebank. In *Proceedings of the Second Chinese Language Processing Workshop*, pp. 1–6, 2000.
- [2] E. Black, F. Jelinek, J. Lafferty, D. Magerman, R. Mercer, and S. Roukos. Towards history-based grammars: Using richer models for probabilistic parsing. In *Proceedings of the 5th DARPA Speech and Natural Language Workshop*, pp. 31–37, 1992.
- [3] A. Böhmová, J. Hajič, E. Hajičová, and B. Hladká. The Prague Dependency Treebank: A three-level annotation scenario. In Anne Abeillé, editor, *Treebanks: Building and Using Parsed Corpora*, pp. 103–127. Kluwer, 2003.
- [4] M. Collins. Head-Driven Statistical Models for Natural Language Parsing. PhD Thesis, University of Pennsylvania, 1999.
- [5] M. Collins, J. Hajič, E. Brill, L. Ramshaw, and C. Tillmann. A statistical parser for Czech. In *Proceedings of ACL*, pp. 505–512, 1999.
- [6] A. Corazza, A. Lavelli, G. Satta, and R. Zanolini. Analyzing an Italian treebank with state-of-the-art statistical parsers. In *Proceedings of TLT*, pp. 39–50, 2004.
- [7] M. A. Covington. A fundamental algorithm for dependency parsing. In *Proceedings of the 39th Annual ACM Southeast Conference*, pp. 95–102, 2001.
- [8] W. Daelemans and A. Van den Bosch. *Memory-Based Language Processing*. Cambridge University Press, 2005.
- [9] A. Dubey and F. Keller. Probabilistic parsing for German using sister-head dependencies. In *Proceedings of ACL*, pp. 96–103, 2003.

- [10] J. Einarsson. Talbankens skriftspråkskonkordans. Lund University, Department of Scandinavian Languages, 1976.
- [11] J. Hajič, B. Vidova Hladka, B. Panevová, E. Hajičová, P. Sgall and P. Pajas. Prague Dependency Treebank 1.0. LDC, 2001T10, 2001.
- [12] J. Hall. A probabilistic part-of-speech tagger with suffix probabilities. Master's thesis, Växjö University, 2003.
- [13] M. T. Kromann. The Danish Dependency Treebank and the DTAG treebank tool. In *Proceedings of TLT*, pp. 217–220. Växjö University Press, 2003.
- [14] R. Levy and C. Manning. Is it harder to parse Chinese, or the Chinese treebank? In *Proceedings of ACL*, pp. 439–446, 2003.
- [15] S. Marinov and J. Nivre. A data-driven dependency parser for Bulgarian. In *Proceedings of TLT*, 2005.
- [16] R. McDonald, K. Crammer, and F. Pereira. Online large-margin training of dependency parsers. In *Proceedings of ACL*, pp. 91–98, 2005.
- [17] J. Nivre. An efficient algorithm for projective dependency parsing. In *Proceedings of IWPT*, pp. 149–160, 2003.
- [18] J. Nivre. Incrementality in deterministic dependency parsing. In *Proceedings of the Workshop on Incremental Parsing (ACL)*, pp. 50–57, 2004.
- [19] J. Nivre. *Inductive Dependency Parsing of Natural Language Text*. PhD thesis, Växjö University, 2005.
- [20] J. Nivre, J. Hall, and J. Nilsson. Memory-based dependency parsing. In *Proceedings of CoNLL*, pp. 49–56, 2004.
- [21] J. Nivre and J. Nilsson. Pseudo-projective dependency parsing. In *Proceedings of ACL*, pp. 99–106, 2005.
- [22] J. Nivre and M. Scholz. Deterministic dependency parsing of English text. In *Proceedings of COLING*, pp. 64–70, 2004.
- [23] K. Simov, G. Popova, and P. Osenova. HPSG-based syntactic treebank of Bulgarian (BulTreeBank). In Andrew Wilson, Paul Rayson, and Tony McEnery, editors, *A Rainbow of Corpora: Corpus Linguistics and the Languages of the World*, pp. 135–142. Lincon-Europa, 2002.
- [24] U. Teleman. *Manual för grammatisk beskrivning av talad och skriven svenska*. Studentlitteratur, 1974.
- [25] T.-F. Wu, C.-J. Lin, and R. C. Weng. Probability estimates for multi-class classification by pairwise coupling. *Journal of Machine Learning Research*, 5:975–1005, 2004.
- [26] H. Yamada and Y. Matsumoto. Statistical dependency analysis with support vector machines. In *Proceedings of IWPT*, pp. 195–206, 2003.