Dependency Parsing and Domain Adaptation with LR Models and Parser Ensembles

Kenji Sagae[1] and **Jun'ichi Tsujii**[1,2,3]
[1]Department of Computer Science
University of Tokyo
Hongo 7-3-1, Bunkyo-ku, Tokyo, Japan
[2]School of Computer Science, University of Manchester
[3]National Center for Text Mining
{sagae,tsujii}@is.s.u-tokyo.ac.jp

Abstract

We present a data-driven variant of the LR algorithm for dependency parsing, and extend it with a best-first search for probabilistic generalized LR dependency parsing. Parser actions are determined by a classifier, based on features that represent the current state of the parser. We apply this parsing framework to both tracks of the CoNLL 2007 shared task, in each case taking advantage of multiple models trained with different learners. In the multilingual track, we train three LR models for each of the ten languages, and combine the analyses obtained with each individual model with a maximum spanning tree voting scheme. In the domain adaptation track, we use two models to parse unlabeled data in the target domain to supplement the labeled out-of-domain training set, in a scheme similar to one iteration of co-training.

1 Introduction

There are now several approaches for multilingual dependency parsing, as demonstrated in the CoNLL 2006 shared task (Buchholz and Marsi, 2006). The dependency parsing approach presented here extends the existing body of work mainly in four ways:

1. Although *stepwise* [1] dependency parsing has commonly been performed using parsing algorithms designed specifically for this task, such as those described by Nivre (2003) and Yamada and Matsumoto (2003), we show that this can also be done using the well known LR parsing algorithm (Knuth, 1965), providing a connection between current research on shift-reduce dependency parsing and previous parsing work using LR and GLR models;
2. We generalize the standard deterministic stepwise framework to probabilistic parsing, with the use of a best-first search strategy similar to the one employed in constituent parsing by Ratnaparkhi (1997) and later by Sagae and Lavie (2006);
3. We provide additional evidence that the parser ensemble approach proposed by Sagae and Lavie (2006a) can be used to improve parsing accuracy, even when only a single parsing algorithm is used, as long as variation can be obtained, for example, by using different learning techniques or changing parsing direction from *forward* to *backward* (of course, even greater gains may be achieved when different algorithms are used, although this is not pursued here); and, finally,
4. We present a straightforward way to perform parser domain adaptation using unlabeled data in the target domain.

We entered a system based on the approach described in this paper in the CoNLL 2007 shared

[1] *Stepwise* parsing considers each step in a parsing algorithm separately, while *all-pairs* parsing considers entire trees. For a more complete definition, see the CoNLL-X shared task description paper (Buchholz and Marsi, 2006).

Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007, pp. 1044–1050,
Prague, June 2007.

task (Nivre et al., 2007), which differed from the 2006 edition by featuring two separate tracks, one in multilingual parsing, and a new track on domain adaptation for dependency parsers. In the multilingual parsing track, participants train dependency parsers using treebanks provided for ten languages: Arabic (Hajic et al., 2004), Basque (Aduriz et al. 2003), Catalan (Martí et al., 2007), Chinese (Chen et al., 2003), Czech (Böhmova et al., 2003), English (Marcus et al., 1993; Johansson and Nugues, 2007), Greek (Prokopidis et al., 2005), Hungarian (Czendes et al., 2005), Italian (Montemagni et al., 2003), and Turkish (Oflazer et al., 2003). In the domain adaptation track, participants were provided with English training data from the Wall Street Journal portion of the Penn Treebank (Marcus et al., 1993) converted to dependencies (Johansson and Nugues, 2007) to train parsers to be evaluated on material in the biological (development set) and chemical (test set) domains (Kulick et al., 2004), and optionally on text from the CHILDES database (MacWhinney, 2000; Brown, 1973).

Our system's accuracy was the highest in the domain adaptation track (with labeled attachment score of 81.06%), and only 0.43% below the top scoring system in the multilingual parsing track (our average labeled attachment score over the ten languages was 79.89%). We first describe our approach to multilingual dependency parsing, followed by our approach for domain adaptation. We then provide an analysis of the results obtained with our system, and discuss possible improvements.

2 A Probabilistic LR Approach for Dependency Parsing

Our overall parsing approach uses a best-first probabilistic shift-reduce algorithm based on the LR algorithm (Knuth, 1965). As such, it follows a bottom-up strategy, or *bottom-up-trees*, as defined in Buchholz and Marsi (2006), in contrast to the shift-reduce dependency parsing algorithm described by Nivre (2003), which is a bottom-up/top-down hybrid, or *bottom-up-spans*. It is unclear whether the use of a bottom-up-trees algorithm has any advantage over the use of a bottom-up-spans algorithm (or vice-versa) in practice, but the availability of different algorithms that perform the same parsing task could be advantageous in parser ensembles. The main difference between our parser and a traditional LR parser is that we do not use an LR table derived from an explicit grammar to determine shift/reduce actions. Instead, we use a classifier with features derived from much of the same information contained in an LR table: the top few items on the stack, and the next few items of lookahead in the remaining input string. Additionally, following Sagae and Lavie (2006), we extend the basic deterministic LR algorithm with a best-first search, which results in a parsing strategy similar to generalized LR parsing (Tomita, 1987; 1990), except that we do not perform Tomita's stack-merging operations.

The resulting algorithm is projective, and non-projectivity is handled by pseudo-projective transformations as described in (Nivre and Nilsson, 2005). We use Nivre and Nilsson's *PATH* scheme[2].

For clarity, we first describe the basic variant of the LR algorithm for dependency parsing, which is a deterministic stepwise algorithm. We then show how we extend the deterministic parser into a best-first probabilistic parser.

2.1 Dependency Parsing with a Data-Driven Variant of the LR Algorithm

The two main data structures in the algorithm are a stack S and a queue Q. S holds subtrees of the final dependency tree for an input sentence, and Q holds the words in an input sentence. S is initialized to be empty, and Q is initialized to hold every word in the input in order, so that the first word in the input is in the front of the queue.[3]

The parser performs two main types of actions: *shift* and *reduce*. When a shift action is taken, a word is shifted from the front of Q, and placed on the top of S (as a tree containing only one node, the word itself). When a reduce action is taken, the

[2] The *PATH* scheme was chosen (even though Nivre and Nilsson report slightly better results with the *HEAD* scheme) because it does not result in a potentially quadratic increase in the number of dependency label types, as observed with the *HEAD* and *HEAD+PATH* schemes. Unfortunately, experiments comparing the use of the different pseudo-projectivity schemes were not performed due to time constraints.

[3] We append a "virtual root" word to the beginning of every sentence, which is used as the head of every word in the dependency structure that does not have a head in the sentence.

two top items in S (s_1 and s_2) are popped, and a new item is pushed onto S. This new item is a tree formed by making the root s_1 of a dependent of the root of s_2, or the root of s_2 a dependent of the root of s_1. Depending on which of these two cases occur, we call the action *reduce-left* or *reduce-right*, according to whether the head of the new tree is to the left or to the right its new dependent. In addition to deciding the direction of a reduce action, the label of the newly formed dependency arc must also be decided.

Parsing terminates successfully when Q is empty (all words in the input have been processed) and S contains only a single tree (the final dependency tree for the input sentence). If Q is empty, S contains two or more items, and no further reduce actions can be taken, parsing terminates and the input is rejected. In such cases, the remaining items in S contain partial analyses for contiguous segments of the input.

2.2 A Probabilistic LR Model for Dependency Parsing

In the traditional LR algorithm, parser states are placed onto the stack, and an LR table is consulted to determine the next parser action. In our case, the parser state is encoded as a set of features derived from the contents of the stack S and queue Q, and the next parser action is determined according to that set of features. In the deterministic case described above, the procedure used for determining parser actions (a classifier, in our case) returns a single action. If, instead, this procedure returns a list of several possible actions with corresponding probabilities, we can then parse with a model similar to the probabilistic LR models described by Briscoe and Carroll (1993), where the probability of a parse tree is the product of the probabilities of each of the actions taken in its derivation.

To find the most probable parse tree according to the probabilistic LR model, we use a best-first strategy. This involves an extension of the deterministic shift-reduce into a best-first shift-reduce algorithm. To describe this extension, we first introduce a new data structure T_i that represents a parser state, which includes a stack S_i, a queue Q_i, and a probability P_i. The deterministic algorithm is a special case of the probabilistic algorithm where we have a single parser state T_0 that contains S_0 and Q_0, and the probability of the parser state is 1. The best-first algorithm, on the other hand, keeps a heap H containing multiple parser states $T_0 ... T_m$. These states are ordered in the heap according to their probabilities, which are determined by multiplying the probabilities of each of the parser actions that resulted in that parser state. The heap H is initialized to contain a single parser state T_0, which contains a stack S_0, a queue Q_0 and probability $P_0 = 1.0$. S_0 and Q_0 are initialized in the same way as S and Q in the deterministic algorithm. The best-first algorithm then loops while H is non-empty. At each iteration, first a state $T_{current}$ is popped from the top of H. If $T_{current}$ corresponds to a final state ($Q_{current}$ is empty and $S_{current}$ contains a single item), we return the single item in $S_{current}$ as the dependency structure corresponding to the input sentence. Otherwise, we get a list of parser actions $act_0 ... act_n$ (with associated probabilities $Pact_0 ... Pact_n$) corresponding to state $T_{current}$. For each of these parser actions act_j, we create a new parser state T_{new} by applying act_j to $T_{current}$, and set the probability T_{new} to be $P_{new} = P_{currnet} * Pact_j$. Then, T_{new} is inserted into the heap H. Once new states have been inserted onto H for each of the n parser actions, we move on to the next iteration of the algorithm.

3 Multilingual Parsing Experiments

For each of the ten languages for which training data was provided in the multilingual track of the CoNLL 2007 shared task, we trained three LR models as follows. The first LR model for each language uses maximum entropy classification (Berger et al., 1996) to determine possible parser actions and their probabilities[4]. To control overfitting in the MaxEnt models, we used box-type inequality constraints (Kazama and Tsujii, 2003). The second LR model for each language also uses MaxEnt classification, but parsing is performed *backwards*, which is accomplished simply by reversing the input string before parsing starts. Sagae and Lavie (2006a) and Zeman and Žabokrtský (2005) have observed that reversing the direction of stepwise parsers can be beneficial in parser combinations. The third model uses support vector machines[5] (Vapnik, 1995) using the polynomial

---

[4] Implementation by Yoshimasa Tsuruoka, available at http://www-tsujii.is.s.u-tokyo.ac.jp/~tsuruoka/maxent/

[5] Implementation by Taku Kudo, available at http://chasen.org/~taku/software/TinySVM/ and all vs. all was used for multi-class classification.

kernel with degree 2. Probabilities were estimated for SVM outputs using the method described in (Platt, 1999), but accuracy improvements were not observed during development when these estimated probabilities were used instead of simply the single best action given by the classifier (with probability 1.0), so in practice the SVM parsing models we used were deterministic.

At test time, each input sentence is parsed using each of the three LR models, and the three resulting dependency structures are combined according to the maximum-spanning-tree parser combination scheme[6] (Sagae and Lavie, 2006a) where each dependency proposed by each of the models has the same weight (it is possible that one of the more sophisticated weighting schemes proposed by Sagae and Lavie may be more effective, but these were not attempted). The combined dependency tree is the final analysis for the input sentence.

Although it is clear that fine-tuning could provide accuracy improvements for each of the models in each language, the same set of meta-parameters and features were used for all of the ten languages, due to time constraints during system development. The features used were[7]:

- For the subtrees in *S(1)* and *S(2)*
  - the number of children of the root word of the subtrees;
  - the number of children of the root word of the subtree to the right of the root word;
  - the number of children of the root word of the subtree to the left of the root word;
  - the POS tag and DEPREL of the rightmost and leftmost children;
- The POS tag of the word immediately to the right of the root word of *S(2)*;
- The POS tag of the word immediately to the left of *S(1)*;
- The previous parser action;
- The features listed for the root words of the subtrees in table 1.

In addition, the MaxEnt models also used selected combinations of these features. The classes used to represent parser actions were designed to encode all aspects of an action (shift vs. reduce, right vs. left, and dependency label) simultaneously.

Results for each of the ten languages are shown in table 2 as labeled and unlabeled attachment scores, along with the average labeled attachment score and highest labeled attachment score for all participants in the shared task. Our results shown in boldface were among the top three scores for those particular languages (five out of the ten languages).

	S(1)	*S(2)*	*S(3)*	*Q(0)*	*Q(1)*	*Q(3)*
WORD	x	x	x	x	x	
LEMMA	x	x		x		
POS	x	x	x	x	x	x
CPOS	x	x		x		
FEATS	x	x		x		

Table 1: Additional features.

Language	LAS	UAS	Avg LAS	Top LAS
Arabic	74.71	84.04	68.34	76.52
Basque	74.64	**81.19**	68.06	76.94
Catalan	**88.16**	**93.34**	79.85	88.70
Chinese	**84.69**	**88.94**	76.59	84.69
Czech	74.83	81.27	70.12	80.19
English	**89.01**	**89.87**	80.95	89.61
Greek	73.58	80.37	70.22	76.31
Hungarian	**79.53**	**83.51**	71.49	80.27
Italian	**83.91**	**87.68**	78.06	84.40
Turkish	75.91	82.72	70.06	79.81
ALL	79.90	85.29	65.50	80.32

Table 2: Multilingual results.

4 Domain Adaptation Experiments

In a similar way as we used multiple LR models in the multilingual track, in the domain adaptation track we first trained two LR models on the out-of-

[6] Each dependency tree is deprojectivized before the combination occurs.

[7] *S(n)* denotes the *nth* item from the top of the stack (where *S(1)* is the item on top of the stack), and *Q(n)* denotes the *nth* item in the queue. For a description of the features names in capital letters, see the shared task description (Nivre et al., 2007).

domain labeled training data. The first was a forward MaxEnt model, and the second was a backward SVM model. We used these two models to perform a procedure similar to a single iteration of co-training, except that selection of the newly (automatically) produced training instances was done by selecting sentences for which the two models produced identical analyses. On the development data we verified that sentences for which there was perfect agreement between the two models had labeled attachment score just above 90 on average, even though each of the models had accuracy between 78 and 79 over the entire development set.

Our approach was as follows:

1. We trained the forward MaxEnt and backward SVM models using the out-of-domain labeled training data;
2. We then used each of the models to parse the first two of the three sets of domain-specific unlabeled data that were provided (we did not use the larger third set)
3. We compared the output for the two models, and selected only identical analyses that were produced by each of the two separate models;
4. We added those analyses (about 200k words in the test domain) to the original (out-of-domain) labeled training set;
5. We retrained the forward MaxEnt model with the new larger training set; and finally
6. We used this model to parse the test data.

Following this procedure we obtained a labeled attachment score of 81.06, and unlabeled attachment score of 83.42, both the highest scores for this track. This was done without the use of any additional resources (closed track), but these results are also higher than the top score for the open track, where the use of certain additional resources was allowed. See (Nivre et al., 2007).

5 Analysis and Discussion

One of the main assumptions in our use of different models based on the same algorithm is that while the output generated by those models may often differ, agreement between the models is an indication of correctness. In our domain adaptation approach, this was clearly true. In fact, the approach would not have worked if this assumption was false. Experiments on the development set were encouraging. As stated before, when the parsers agreed, labeled attachment score was over 90, even though the score of each model alone was lower than 79. The domain-adapted parser had a score of 82.1, a significant improvement. Interestingly, the ensemble used in the multilingual track also produced good results on the development set for the domain adaptation data, without the use of the unlabeled data at all, with a score of 81.9 (although the ensemble is more expensive to run).

The different models used in each track were distinct in a few ways: (1) direction (forward or backward); (2) learner (MaxEnt or SVM); and (3) search strategy (best-first or deterministic). Of those differences, the first one is particularly interesting in single-stack shift-reduce models, as ours. In these models, the context to each side of a (potential) dependency differs in a fundamental way. To one side, we have tokens that have already been processed and are already in subtrees, and to the other side we simply have a look-ahead of the remaining input sentence. This way, the context of the same dependency in a forward parser may differ significantly from the context of the same dependency in a backward parser. Interestingly, the accuracy scores of the MaxEnt backward models were found to be generally just below the accuracy of their corresponding forward models when tested on development data, with two exceptions: Hungarian and Turkish. In Hungarian, the accuracy scores produced by the forward and backward MaxEnt LR models were not significantly different, with both labeled attachment scores at about 77.3 (the SVM model score was 76.1, and the final combination score on development data was 79.3). In Turkish, however, the backward score was significantly higher than the forward score, 75.0 and 72.3, respectively. The forward SVM score was 73.1, and the combined score was 75.8. In experiments performed after the official submission of results, we evaluated a *backward* SVM model (which was trained after submission) on the same development set, and found it to be significantly more accurate than the forward model, with a score of 75.7. Adding that score to the combination raised the combination score to 77.9 (a large improvement from 75.8). The likely reason for this difference is that over 80% of the dependencies in the Turkish data set have the head to the right of the dependent, while only less than 4% have the head to the left. This means that the backward model builds much more partial structure in the stack as it consumes input tokens, while the forward model must consume most tokens before it starts making attachments. In other words, context in general in the backward model has more structure, and attachments are made while there are still look-ahead tokens, while the opposite is generally true in the forward model.

6 Conclusion

Our results demonstrate the effectiveness of even small ensembles of parsers that are relatively similar (using the same features and the same algorithm). There are several possible extensions and improvements to the approach we have described. For example, in section 3 we mention the use of different weighting schemes in dependency voting. We list additional ideas that were not attempted due to time constraints, but that are likely to produce improved results.

One of the simplest improvements to our approach is simply to train more models with no other changes to our set-up. As mentioned in section 5, the addition of a backward SVM model did improve accuracy on the Turkish set significantly, and it is likely that improvements would also be obtained in other languages. In addition, other learning approaches, such as memory-based language processing (Daelemans and Van den Bosch, 2005), could be used. A drawback of adding more models that became obvious in our experiments was the increased cost of both training (for example, the SVM parsers we used required significantly longer to train than the MaxEnt parsers) and run-time (parsing with MBL models can be several times slower than with MaxEnt, or even SVM). A similar idea that may be more effective, but requires more effort, is to add parsers based on different approaches. For example, using MSTParser (McDonald and Pereira, 2005), a large-margin all-pairs parser, in our domain adaptation procedure results in significantly improved accuracy (83.2 LAS). Of course, the use of different approaches used by different groups in the CoNLL 2006 and 2007 shared tasks represents great opportunity for parser ensembles.

Acknowledgements

We thank the shared task organizers and treebank providers. We also thank the reviewers for their comments and suggestions, and Yusuke Miyao for insightful discussions. This work was supported in part by Grant-in-Aid for Specially Promoted Research 18002007.

References

A. Abeillé, editor. 2003. *Treebanks: Building and Using Parsed Corpora*. Kluwer.

A. Berger, S. A. Della Pietra, and V. J. Della Pietra. 1996. A maximum entropy approach to naturallanguage processing. *Computational Linguistics*, 22(1):39–71.

I. Aduriz, M. J. Aranzabe, J. M. Arriola, A. Atutxa, A. Diaz de Ilarraza, A. Garmendia and M. Oronoz. 2003. Construction of a Basque Dependency Treebank. In *Proc. of the 2nd Workshop on Treebanks and Linguistic Theories (TLT)*, pages 201–204.

A. Böhmová, J. Hajic, E. Hajicová and B. Hladká. 2003. The PDT: a 3-level annotation scenario. In Abeillé (2003), chapter 7, 103–127.

E. Briscoe and J. Carroll. 1993. Generalized Probabilistic LR Parsing of Natural Language (Corpora) with Unification-Based Grammars. In *Computational Linguistics*, 19(1), pages 25-59.

R. Brown. 1973. *A First Language: The Early Stages*. Harvard University Press.

S. Buchholz and E. Marsi. 2006. CoNLL-X Shared Task on Multilingual Dependency Parsing. In *Proc. of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*. New York, NY.

K. Chen, C. Luo, M. Chang, F. Chen, C. Chen, C. Huang and Z. Gao. 2003. Sinica Treebank: Design Criteria, Representational Issues and Implementation. In Abeillé (2003), chapter 13, pages 231–248.

D. Csendes, J. Csirik, T. Gyimóthy, and A. Kocsor. 2005. *The Szeged Treebank*. Springer.

W. Daelemans and A. Van den Bosch. 2005. *Memory-based language processing*. Cambridge University Press.

J. Hajic, O. Smrz, P. Zemánek, J. Snaidauf and E. Beska. 2004. Prague Arabic Dependency Treebank: Development in Data and Tools. In *Proc. of the NEMLAR Intern. Conf. on Arabic Language Resources and Tools*, pages 110–117.

R. Johansson and P. Nugues. 2007. Extended constituent-to-dependency conversion for English. In *Proc. of the 16th Nordic Conference on Computational Linguistics (NODALIDA).*

J. Kazama, and J. Tsujii. 2003. Evaluation and extension of maximum entropy models with inequality constraints. In *Proceedings of EMNLP 2003.*

D. Knuth. 1965. On the translation of languages from left to right, *Information and Control* 8, 607-639.

S. Kulick, A. Bies, M. Liberman, M. Mandel, R. McDonald, M. Palmer, A. Schein, and L. Ungar. 2004. Integrated annotation for biomedical information extraction. In *Proc. of the Human Language Technology Conference and the Annual Meeting of the North American Chapter of the Association for Computational Linguistics (HLT/NAACL).*

B. MacWhinney. 2000. The CHILDES Project: Tools for Analyzing Talk. Lawrence Erlbaum.

R. McDonald, K.Crammer, and F. Pereira. 2005. Online large-margin training of dependency parsers. In *Proc. of the 43rd Annual Meeting of the Association for Computational Linguistics*, 2005

M. Marcus, B. Santorini, and M. Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313–330.

M. A. Martí, M. Taulé, L. Màrquez and M. Bertran. 2007. CESS-ECE: A Multilingual and Multilevel Annotated Corpus. Available for download from: http://www.lsi.upc.edu/~mbertran/cess-ece/.

S. Montemagni, F. Barsotti, M. Battista, N. Calzolari, O. Corazzari, A. Lenci, A. Zampolli, F. Fanciulli, M. Massetani, R. Raffaelli, R. Basili, M. T. Pazienza, D. Saracino, F. Zanzotto, N. Nana, F. Pianesi, and R. Delmonte. 2003. Building the Italian Syntactic-Semantic Treebank. In Abeillé (2003), chapter 11, pages 189–210.

J. Nivre. 2003. An efficient algorithm for dependency parsing. In *Proc. of the Eighth International Workshop on Parsing Technologies (IWPT'03).* Nancy, France.

J. Nivre, and J. Nilsson. 2005. Pseudo-Projective Dependency Parsing. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL),* 99-106. Ann Arbor, MI.

J. Nivre, J. Hall, S. Kübler, R. McDonald, J. Nilsson, S. Riedel, and D. Yuret. 2007. The CoNLL 2007 shared task on dependency parsing. In *Proc. of the CoNLL 2007 Shared Task. Joint Conf. on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL).*

K. Oflazer, B. Say, D. Zeynep Hakkani-Tür, and G. Tür. 2003. Building a Turkish treebank. In Abeillé (2003), chapter 15, pages 261–277.

J. Platt. 1999. Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods. In *Advances in Large Margin Classiers*, MIT Press.

P. Prokopidis, E. Desypri, M. Koutsombogera, H. Papageorgiou, and S. Piperidis. 2005. Theoretical and practical issues in the construction of a Greek depen- dency treebank. In Proc. of the 4th Workshop on Treebanks and Linguistic Theories (TLT), pages 149–160.

A. Ratnaparkhi. 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*. Providence, RI

K. Sagae, and A. Lavie. 2006. A best-first probabilistic shift-reduce parser. *Proceedings of the 43rd Meeting of the Association for Computational Linguistics - posters (ACL'06).* Sydney, Australia.

K. Sagae, and A. Lavie. 2006a. Parser combination by reparsing. *Proceedings of the 2006 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics - short papers (HLT-NAACL'06).* New York, NY.

M. Tomita. 1987. An efficient augmented context-free parsing algorithm. *Computational Linguistics*, 13:31–46.

M. Tomita. 1990. The generalized LR parser/compiler - version 8.4. In *Proceedings of the International Conference on Computational Linguistics (COLING'90)*, pages 59–63. Helsinki, Finland.

V. N. Vapnik. 1995. *The Nature of Statistical Learning Theory*. Springer-Verlag.

H. Yamada, and Y. Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proceedings of the Eighth International Workshop on Parsing Technologies (IWPT'03).* Nancy, France.

D. Zeman, Z. Žabokrtský. 2005. Improving Parsing Accuracy by Combining Diverse Dependency Parsers. In *Proceedings of the International Workshop on Parsing Technologies (IWPT 2005).* Vancouver, British Columbia.