

# Experiments with MaltParser for parsing Indian Languages

**Sudheer Kolachina**

LTRC, IIIT-Hyderabad

sudheer.kpg08@research.iiit.ac.in

**Prasanth Kolachina**

LTRC, IIIT-Hyderabad

prasanth.k@students.iiit.ac.in

**Manish Agarwal**

LTRC, IIIT-Hyderabad

manish.agarwal@students.iiit.ac.in

**Samar Husain**

LTRC, IIIT-Hyderabad

samar@research.iiit.ac.in

## Abstract

In this paper, we describe our experiments on applying the MaltParser for parsing Indian languages. We explore two new kinds of features hitherto unexplored in the parsing of Indian languages, namely, valency and arc-directionality. We motivate the use of arc-directionality features based on empirical evidence in the form of statistics collected over the treebanks. Valency information is made available during parsing through feature propagation. Finally, we blend the outputs of different single Malt parsers built in our experiments to create two blended systems (one coarse-grained and one fine-grained) each for Hindi, Telugu and Bangla. This is also the first attempt at exploring ensemble approaches such as blending for parsing Indian languages. Our submission consisting of these blended systems was ranked second overall at the ICON-2010 tools' contest on IL parsing.

## 1 Introduction

The earliest attempt at building NL parsing systems for Indian languages was made using a constraint-based parsing approach in the rule-based paradigm (Bharati and Sangal, 1993). This approach worked with a dependency-based representation of syntactic structure as Indian languages have relatively free word order. The main intuition behind this approach was that, given the rich morphology of these languages, morphological markers serve as strong cues for identifying the syntactic relations between the words in a sentence. In recent times, the development of syntactically annotated treebank corpora has paved the

way for an alternative paradigm for parsing Indian languages: data-driven (or statistical) parsing. Statistical approaches require less effort to build parsers, although their performance is directly related to the quality and size of the treebank used to build parsers. Although portability to new languages is one of the widely claimed advantages of the data-driven approach, porting statistical parsers to new languages is not always straightforward and has been shown to throw up challenging research issues (Eryigit et al., 2008). Much of the recent work on application of data-driven approaches for parsing Indian languages which are morphologically rich and free word order (MoR-FWO) has focused on incorporating language-specific properties as features that are crucial during parsing. We briefly overview some of this work in section 2. In this paper, we continue this search for new ways to incorporate linguistic knowledge in the form of features that might prove to be beneficial for parsing Indian languages. We experiment with the latest version of the publicly available MaltParser (Nivre et al., 2007). The primary novelty of the experiments reported in this paper is in the exploration of two new kinds of linguistic information: valency information using the new *propagation* feature in MaltParser-1.4 and arc-directionality features which we argue, based on the statistics presented in section 3, are very pertinent for Indian languages. The results of our experiments show that incorporating even a small number of such features based on linguistic intuition can improve the performance of the parser. In the second stage of parser optimization, we use an ensemble technique to combine the outputs of different single Malt parsers to build blended parsers for each language.

The rest of the paper is divided into four sec-

tions. In section 2, we briefly discuss previous work on data-driven parsing for Indian languages. In section 3 we give a very brief description of the treebanks accompanied by a few interesting statistics from the treebanks based on which we motivate feature selection in our experiments. Section 4 contains the details of our experiments with the MaltParser (Nivre et al., 2007) on the multilingual datasets along with the parsing accuracies obtained in each experiment. Finally, we conclude the paper in section 5 with a concise summary of the insights gained from our experiments.

## 2 Related Work

Bharati et al. (2008) are the first to apply data-driven parsing approaches to Indian languages. In this work, they apply two data-driven parsers-MaltParser and MSTParser (McDonald et al., 2005) to the Hindi dependency treebank. In this work, it is shown that incorporating just two light semantic features can improve the parsing accuracy considerably.

During the ICON-09 tools' contest on dependency parsing of Indian languages<sup>1</sup>, a variety of different approaches were tried out for parsing Indian languages. Husain (2009) presents a succinct summary of all these approaches. Nivre (2009) gives a detailed account of a complete set of experiments on applying the MaltParser which implements a deterministic transition-based parsing approach for parsing Indian languages, focusing on three different parsing algorithms (two projective and one non-projective), feature models and classifier settings. Feature optimization is automated using both backward deletion and forward selection techniques. The system described in this work was ranked second overall in the shared task. Ambati et al. (2009) also experiment with the MaltParser by trying out various learner settings and feature models reported by the MaltParser at the CoNLL 2007 shared task on dependency parsing (Hall et al., 2007). Additionally, they also report details of their experiments with the MSTParser. Their best results, however, are reported using MaltParser and are slightly higher than Nivre (2009). Mannem (2009) uses a bidirectional parser with two different operations to connect nodes, a non-projective operation to connect the head in a non-projective relation to its parent, and a projective operation to connect all

other nodes with their parents in the dependency tree. This approach uses a bidirectional algorithm searching the sentence for the most confident dependency hypothesis in both directions (Shen and Joshi, 2008). This approach produces better results on the coarse-grained datasets compared to the fine-grained ones. It must be mentioned that this shared task dealt with shallow parsing wherein syntactic dependencies are predicted among the chunks in the sentence.

Ambati et al. (2010b) explore further the use of transition-based parsing approach for dependency parsing of Hindi. They build on previous work such as Nivre (2009) and Ambati et al. (2009) by exploring a common pool of features used by these systems. The results reported in this work are the state of the art for Hindi chunk parsing on the ICON-09 tools' contest dataset. Ambati et al. (2010a) describe two methods to use local morpho-syntactic information such as chunk type, head/non-head information, chunk boundary information, etc. during dependency parsing of Hindi sentences. This work which uses both Malt and MST parsers is also the first attempt at word level parsing for Hindi. Ambati et al. (2010) empirically demonstrate the importance of linguistic constraints in statistical dependency parsing for Hindi. Improvements in the performance of Malt and MST parsers can be achieved by imposing the simple constraint that an argument label of a verb must be assigned uniquely to only one child node.

## 3 Datasets and Treebank statistics

A dependency annotation scheme inspired by Paninian theory was proposed for syntactic treebanking of Indian languages (ILs) which are both morphologically rich and relatively free word-order (Begum et al., 2008). Currently, treebanks of three Indian languages<sup>2</sup>, Hindi, Bangla and Telugu are being developed using this annotation scheme. The first versions of all three treebanks were released for the shared task on IL parsing held as part of ICON-2009. The latest versions of these treebanks were released for the ICON 2010 tools' contest on IL parsing earlier this year. In the latest version of the Hindi treebank, the parse trees are augmented with intra-chunk relations and the size of the treebank itself was increased considerably. The Bangla and Telugu treebanks, however, still consist of sentences annotated only for inter-

<sup>1</sup><http://ltrc.iiit.ac.in/nlptools2009/>

<sup>2</sup>HyDT-Hindi, HyDT-Bangla and HyDT-Telugu

	Bangla	Hindi	Telugu
Total #edges	7252	77066	5722
Total #edges with head to right	5278	41988	4041
Total #edges with head to left	1974	35078	1681
Ratio of right-headed to left-headed edges	2.6737	1.1969	2.4039
#edges with head to right excluding <i>main,rsym,lwg*</i>	5278	40637	4041
#edges with head to left excluding <i>main,rsym,lwg*</i>	844	5151	228
Ratio of right-headed to left-headed edges	6.2536	7.8891	17.7236

Table 1: Arc directionality statistics extracted from the treebanks

chunk relations. The previous version of the Telugu treebank was revised keeping in mind specific issues which were encountered at the ICON-2009 tools contest on IL parsing. Husain et al. (2010) gives a detailed description of the treebank versions released this year as well as the statistics for the different datasets.

At the very outset of our work, we extracted a few statistics from the treebanks (training and validation sets combined) which have proved to be helpful in our experiments on feature optimization. These statistics were extracted to empirically verify some of the widely claimed typological properties of Indian languages. For instance, Indian languages are often described as being head-final or in other words, having a basic SOV word order. In terms of dependency-based representation, what this implies is that the direction of the dependencies would be predominantly to the right. However, at the same time, Indian languages are sometimes also characterized as having relatively free word order. In order to gain some corpus-based insight into the word order aspect of these languages, we extracted the frequencies of dependency edges with respect to the direction of their heads shown in Table 1 for all three languages. The total number of dependency edges in each of the three treebanks is shown in row 1. The number of edges with respect to the direction of the head of that dependency are shown for all the three languages in rows 2 and 3. The values of the ratios presented in row 4, although greater than 1, are not high enough to empirically support the claim that Indian languages, the direction of the dependencies is predominantly rightward. In fact, the values can be interpreted as indicating a high degree of deviation from the basic SOV order. However, a closer look at the statistics shows that this is not the case. The high numbers of leftward dependencies include linguistically uninformative (perhaps

calling them less informative would be more apt) relations such as ‘main’, ‘rsym’, etc. In the case of Hindi, the very high number of leftward dependencies is also owing to intra-chunk relations such as the ones between nominal post-positions and the nouns whose inflections they mark, and also the ones between verbal auxiliaries and the main verbs they follow. The frequencies of the edges excluding these relations are shown in rows 5 and 6. The ratios in row 7 confirm that the distribution of edges with respect to their directionality is in Indian languages is skewed to the right. It may be noticed that the degree of skewing is much higher in the case of Telugu as compared to the other two languages. This is in agreement with linguistically attested facts about Hindi and Bangla in that deviations from the basic SOV order are possible in the case of finite clausal complements (which occur to the right rather than the left of the verb) in these languages. In section 4.5, we describe how we base our choice of feature on insights gained from such statistics.

## 4 Experiments

In our experiments, we use the freely available MaltParser (Nivre et al., 2007) which implements deterministic, classifier-based parsing with history-based feature models and discriminative learning. The parameters available in the MaltParser can be divided into three groups:

1. Parsing algorithm parameters
2. Learning algorithm parameters
3. Feature model parameters

Although we experiment with different combinations of these three parameters while defining the baseline systems initially, the main focus of our latter experiments is on feature optimization.

In fact, the main novelty of our experiments lies in the exploration of feature propagation as it is implemented in the latest version of the MaltParser<sup>3</sup>. There is an additional parameter that can be varied with the MaltParser- Single Malt versus Blended. Blending the output of different single Malt system proved to be beneficial for multilingual parsing at the CoNLL 2007 shared task (Hall et al., 2007). We follow this precedent of blending for multilingual parsing to combine the output of various parsers obtained in our experiments with different parameters.

#### 4.1 Experiments with Parsing Algorithm

Malt parser implements three different families of parsing algorithms, comprising of a total of nine parsing algorithms (Nivre et al., 2007). In our experiments, we restricted our choices of the parsing algorithm to the Nivre family (Nivre, 2003; Nivre, 2004), a linear-time algorithm for projective structures that can be run in either arc-eager mode or arc-standard mode. We applied the left-to-right versions of this algorithm in both the modes to the multilingual datasets while fixing the feature model and the learner settings. We also used the root-handling option to change the parser’s behavior with respect to root tokens by changing the label to *main*. We observed that the arc-eager algorithm performs better in the case of Hindi and Telugu, whereas arc-standard gives a slightly higher accuracy for Bangla.

#### 4.2 Experiments with SVM Learner

MaltParser can be used with different learning algorithms that induce classifiers from training data (Nivre et al., 2007). The current version of MaltParser provides two inbuilt learning algorithms: LIBSVM and LIBLINEAR. In our experiments, we used the LIBSVM learner algorithm following the experiments reported by Hall et al. (Hall et al., 2007) at the CoNLL Shared Task 2007. We try out the learner settings of various languages presented in this work. The aim as well as intuition in this set of experiments was to see if the choice of learner settings can be based on the typological similarities between Indian languages and the languages explored at CoNLL 2007. We observed for all three languages that the effect of varying the learner settings on the parsing accuracies is relatively weaker when compared to the

<sup>3</sup>MaltParser 1.4.1

Language	Algorithm	SVM settings
Bangla	Arc-standard	Czech settings s0t1d2g0.2c0.25r0.3e1.0
Hindi	Arc-eager	Turkish settings s0t1d2g0.12c0.7r0.3e0.5
Telugu	Arc-eager	Italian settings s0t1d2g0.1c0.5r0.6e1.0

Table 2: Best combinations of parsing algorithm and learner settings for different languages

parsing algorithm parameter discussed in the previous experiment.

In table 2, we report the combinations of learner settings and the parsing algorithm for different languages for which the best accuracies were obtained in this set of experiments. As can be seen from the table, the values of the *svm\_type*, *kernel type* and *degree* parameters of the learner are the same for all three languages. Hindi and Bangla share the same value for the kernel coefficient parameter ( $r$ ). On the other hand, the value of the termination criterion ( $e$ ) is the same for Telugu and Bangla.

#### 4.3 Initial experiments with Feature Models

In the initial set of experiments described above, we used a baseline feature model for Hindi that consisted of features reported in Ambati et al (2010b). In the case of Bangla and Telugu, we treated the feature models used by the best performing system in last year’s contest (Ambati et al., 2009) as our baseline feature models. Using these feature models and the best combinations of the other two parameters obtained from the previous experiments, we built a parser for each language which we define as our baseline system for that language for the rest of the experiments in our work. Using these baseline settings, we performed ten-fold cross-validation on the multilingual datasets, and the corresponding accuracies are shown in table 3. Also shown are the accuracies obtained on the development set. Note the absence of accuracies on the coarse-grained<sup>4</sup> datasets from this table. This is because we restricted ourselves to the fine-grained datasets in our experiments on feature optimization.

We also explored the possibility of directly using the best feature models of different languages reported by the MaltParser at the CoNLL 2007

<sup>4</sup>The coarse-grained datasets were created by reducing the granularity of the dependency labels in the fine-grained sets. See Husain et al. (2010) for further details.

Dataset	Cross-Validation			Development set		
	LAS	UAS	LAcc	LAS	UAS	LAcc
Bangla-fine	67.74	84.84	70.01	68.47	88.42	70.44
Hindi-fine	85.73	92.79	87.59	85.91	92.37	87.60
Telugu-fine	67.98	89.78	69.90	70.85	91.12	72.70

Table 3: Baseline parsing accuracies on all datasets; Cross-validation accuracies with baseline model on training sets; baseline results on development set. LAS = labeled attachment score; UAS = unlabeled attachment score; LAcc = label accuracy.

shared task (Hall et al., 2007), the intuition behind this set of experiments being similar to the one discussed in the case of learner settings. We were able to complete this set of experiments for Telugu and Bangla as the datasets of these languages are quite small. But in the case of Hindi, due to the relatively large size of the datasets and the tight constraints on time and resources, we could not train parsers for Hindi using all these different feature models. We used parsers built this way in our final experiment while blending the output of different single Malt parsers to build our final system. In this set of experiments, we observed that the best accuracies for both Telugu and Bangla were obtained using the feature model for Basque which is also a non-configurational language with rich morphology and relatively free word order.

#### 4.4 Experiments with Feature Propagation

After establishing the baseline systems, we began to explore the idea of feature propagation through dependency arcs as a means to use valency information during parsing. Feature propagation is a well-known strategy used in classical unification-based grammars, as a means of propagating linguistic information through the syntax tree. The application of this idea was explored previously for dependency parsing of Basque by Bengoetxea and Gojenola (2009; 2010). Feature propagation in this work was achieved through stacking. In our experiments, we use feature propagation as it is implemented in the recent version of the MaltParser<sup>5</sup>. In this implementation, values of propagated features get copied to temporary data columns and are accessible during parsing. Since the goal of our experiments on feature propagation was to incorporate information about valency into the parsing process, we propagated the labels of the outgoing arcs up to the parent nodes. The idea is to make the valency and hence argument struc-

<sup>5</sup>MaltParser 1.4.1

Experiment	Dataset	LAS	UAS	LAcc
V	Bangla-fine	67.61	87.81	69.95
	Hindi-fine	86.07	92.54	87.87
	Telugu-fine	70.69	92.13	72.36
V+ArcDir	Bangla-fine	68.23	88.30	70.57
	Hindi-fine	85.91	92.41	87.81
	Telugu-fine	71.36	92.13	73.37

Table 4: Parser accuracies on development set using valency information and arc-directionality features; V: Valency information, ArcDir: Arc Directionality information

ture of nodes accessible during parsing. Due to the fine-grained nature of the dependency labels in the datasets, we only propagated the labels of mandatory arguments as opposed to adjuncts. The following labels were considered as argument labels: *k1*, *k1s*, *k2*, *k2p*, *k4* and *k4a*. As expected, we observed an improvement, although slight over the baseline accuracies when the argument labels of the top of Stack and the head of the top of Stack were propagated. Further, we observed that the better improvements could be obtained by merging the valency of these elements with the POS-tag, CPOS-tag and lemma features of the next token in the input buffer (Input[0]). The accuracies in Table 4 show that valency information improves only the UAS in the case of Telugu and both UAS and LAS in the for Hindi. In the case of Bangla, the accuracies actually drop with the addition of these new features.

#### 4.5 Additional features

In this last set of experiments on tuning the feature models, we tried to augment the current feature models of different languages with more features. It is at this point that we found the statistics extracted from the treebanks to be useful while handpicking features to augment the feature mod-

Dataset	Baseline model			Best Feature model			Blended		
	LAS	UAS	LACC	LAS	UAS	LACC	LAS	UAS	LACC
Bangla-Fine	68.26	86.16	71.49	68.57	85.54	71.28	70.14	87.1	73.26
Hindi-Fine	86.60	93.37	88.22	86.49	93.50	88.16	86.22	93.25	87.95
Telugu-Fine	67.78	88.48	69.95	69.28	90.48	70.95	68.11	90.15	70.12
Bangla-Coarse	73.15	86.47	76.69	73.26	87.51	75.67	75.65	88.14	78.67
Hindi-Coarse	88.57	94.01	90.75	88.96	94.13	90.91	88.96	94.13	90.91
Telugu-Coarse	68.95	88.81	71.45	69.62	90.15	71.79	69.45	89.98	71.29

Table 5: Final results on the test set for different languages

els. Based on the insights we gained about the skewed distribution of the edges with respect to the direction of their heads (see table 1), we chose to experiment with the InputArcDir feature function in MaltParser. In our experiments, we observed that merging this feature function that deals with arc-directionality of the arc outgoing from the top of Stack with POS-tag, CPOS-tag and lemma features of both the head of top of Stack and the next token in the input buffer brings about an improvement in the LAS for Telugu. Information about the direction of the arc seems to be helpful for label prediction in Telugu. Tuning the feature models this way provides a better understanding of how a particular feature works across different languages. Apart from automating feature selection from a large pool of features, tuning the feature model using handpicked features based on linguistic properties of individual languages can give insights into how different features interact during parsing. The accuracies obtained in this set of experiments are also shown in table 4. While the accuracies for Telugu and Bangla show improvements, there is a drop in the case of Hindi. This can be attributed to the difference in the nature of the datasets- the Telugu and Bangla datasets contain chunk-level dependency trees while the Hindi data consists of word-level trees. There are a number of intra-chunk dependencies (*lwg\**) in the case of Hindi which are not head-final (see table 1).

#### 4.6 Blended Malt

Finally, following Hall et al. (2007), we decided to blend the different parsers (using the Maltblender tool) built in our experiments to create Blended MaltParsers for each language. An important point about an ensemble approach like blending is that its efficacy depends on the degree of variation among the parsers being blended. In our initial experiments on blending, we noticed that it is

beneficial to include parsers which are trained using entirely different settings although their performance may be relatively low. In other words, blending systems built using different parameter settings is preferable to simply blending systems with the highest accuracies. Further, we also observed that weighting the dependencies according to labeled accuracy per coarse-grained POS-tag of the parsers produces the best results. Here again, as elsewhere, we resort to the typological properties of languages to decide which parsers to combine for each language. We blend the single Malt parsers built using our feature models with the ones built using feature models of non-configurational languages from the CoNLL shared task- Arabic, Basque, Czech, Greek, Hungarian and Turkish. In the case of Telugu, we additionally used the parser built using the feature model for Catalan as it gave good accuracies on the development set. In the case of Bangla, blending was done similarly except that we used the parser built using the feature model of English instead of Catalan. For Hindi, as mentioned earlier, we did not have such a variety of systems and therefore, the blended system was created by combining the output of the only five parsers. As a result, blending does not bring about any improvements in the case of Hindi.

The final results obtained on the test sets for all three languages using the various systems built in our experiments are shown in Table 5. The accuracies in column 2 are using the best feature model for each language obtained in our experiments on feature model tuning.

## 5 Conclusions and Future work

In this paper, we experiment with different parameters in the MaltParser to build dependency parsers for three Indian languages: Hindi, Telugu and Bangla. The novelty of our experiments lies in

the exploration of new features pertaining to valency and arc-directionality. Using feature propagation, we make valency information accessible to the parser during parsing. We motivate the use of arc-directionality features on the basis of empirical evidence collected from the treebank. The results of our preliminary experiments show that the use of both these novel features can improve parsing performance. Finally, we blend the output of different single Malt parsers to create a blended Malt parser for each of the languages. We report the parsing accuracies obtained in each set of experiments. Since our main aim was to build high quality parsers for Indian languages, we tried out different ways to this end. The experiments in our work must, therefore, be seen as preliminary and further experiments need to be carried out before definitive claims can be made about the impact of both valency and arc-directionality information. Also, with regards to blending, in hindsight, it would have been really useful to have more base parsers for Hindi. However, the results of blending for Telugu and Bangla show ensemble approaches to be a new direction worth pursuing for parsing Indian Languages.

## Acknowledgements

We thank Prof. Joakim Nivre for helping us out with the new propagation feature in MaltParser 1.4. We would also like to thank Bharat Ram Ambati and Phani Gadde for giving us the baseline feature models. Thanks are also due to two reviewers for their useful comments and feedback.

## References

- B.R. Ambati, P. Gadde, and K. Jindal. 2009. Experiments in Indian Language Dependency Parsing. *Proceedings of the ICON09 NLP Tools Contest: Indian Language Dependency Parsing*, pages 32–37.
- B.R. Ambati, S. Husain, S. Jain, D.M. Sharma, and R. Sangal. 2010a. Two methods to incorporate local morphosyntactic features in Hindi de-pendency parsing. In *The First Workshop on Statistical Parsing of Morphologically Rich Languages (SPMRL 2010)*, page 22.
- B.R. Ambati, S. Husain, J. Nivre, and R. Sangal. 2010b. On the role of morphosyntactic features in Hindi dependency parsing. In *The First Workshop on Statistical Parsing of Morphologically Rich Languages (SPMRL 2010)*, page 94.
- B.R. Ambati. 2010. Importance of linguistic constraints in statistical dependency parsing. In *Proceedings of the ACL 2010 Student Research Workshop*, pages 103–108. Association for Computational Linguistics.
- R. Begum, S. Husain, A. Dhvaj, D.M. Sharma, L. Bai, and R. Sangal. 2008. Dependency annotation scheme for Indian languages. *Proceedings of IJCNLP-2008*.
- K. Bengoetxea and K. Gojenola. 2009. Application of feature propagation to dependency parsing. In *Proceedings of the 11th International Conference on Parsing Technologies*, pages 142–145. Association for Computational Linguistics.
- K. Bengoetxea and K. Gojenola. 2010. Application of different techniques to dependency parsing of Basque. In *The First Workshop on Statistical Parsing of Morphologically Rich Languages*, page 31.
- A. Bharati and R. Sangal. 1993. Parsing free word order languages in the Paninian framework. In *Proceedings of the 31st annual meeting on Association for Computational Linguistics*, pages 105–111. Association for Computational Linguistics.
- A. Bharati, S. Husain, B. Ambati, S. Jain, D. Sharma, and R. Sangal. 2008. Two semantic features make all the difference in parsing accuracy. *Proc. of ICON*, 8.
- G. Eryigit, J. Nivre, and K. Oflazer. 2008. Dependency parsing of Turkish. *Computational Linguistics*, 34(3):357–389.
- J. Hall, J. Nilsson, J. Nivre, G. Eryigit, B. Megyesi, M. Nilsson, and M. Saers. 2007. Single malt or blended? A study in multilingual parser optimization. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 933–939. Association for Computational Linguistics.
- S. Husain, P. Mannem, B. Ambati, and P. Gadde. 2010. The ICON-2010 tools contest on Indian language dependency parsing. *Proceedings of ICON10 NLP Tools Contest on Indian Language Dependency Parsing*.
- S. Husain. 2009. Dependency Parsers for Indian Languages. *Proceedings of ICON09 NLP Tools Contest: Indian Language Dependency Parsing*.
- P. Mannem. 2009. Bidirectional Dependency Parser for Hindi, Telugu and Bangla. *ICON09 NLP TOOLS CONTEST: INDIAN LANGUAGE DEPENDENCY PARSING*, page 49.
- R. McDonald, K. Crammer, and F. Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 91–98. Association for Computational Linguistics.
- J. Nivre, J. Hall, J. Nilsson, A. Chanev, G. Eryigit, S. Kübler, S. Marinov, and E. Marsi. 2007. Malt-Parser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(02):95–135.

- J. Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT)*. Citeseer.
- J. Nivre. 2004. Incrementality in deterministic dependency parsing. In *Proceedings of the Workshop on Incremental Parsing: Bringing Engineering and Cognition Together*, pages 50–57. Association for Computational Linguistics.
- J. Nivre. 2009. Parsing Indian Languages with Malt-Parser. *Proc. of ICON09 NLP Tools Contest: Indian Language Dependency Parsing*, pages 12–18.
- L. Shen and A.K. Joshi. 2008. Ltag dependency parsing with bidirectional incremental construction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 495–504. Association for Computational Linguistics.