Hindi Dependency Parsing using a combined model of Malt and MST

B. Venkata Seshu Kumari[*#], *Ramisetty Rajeswara Rao*[*$]
* JNTU, Hyderabad, # St. Peters' Engineering College, Hyderabad, $ MGIT, Hyderabad
hemakrishna.ambati@gmail.com, hodcse@mgit.ac.in

ABSTRACT

In this paper we present our experiments in parsing Hindi. We first explored Malt and MST parsers. Considering pros of both these parsers, we developed a hybrid approach combining the output of these two parsers in an intuitive manner. We report our results on both development and test data provided in the Hindi Shared Task on Parsing at workshop on MT and parsing in Indian Languages, Coling 2012. Our system secured labeled attachment score of 90.66% and 80.77% for gold standard and automatic tracks respectively. These accuracies are 3rd best and 5th best for gold standard and automatic tracks respectively.

1 Introduction

Dependency parsing is the task of uncovering the dependency tree of a sentence, which consists of labeled links representing dependency relationships between words. Parsing is useful in major NLP applications like Machine Translation, Dialogue systems, text generation, word sense disambiguation etc. This led to the development of grammar-driven, data-driven and hybrid parsers. Due to the availability of annotated corpora in recent years, data driven parsing has achieved considerable success. The availability of phrase structure treebank for English has seen the development of many efficient parsers.

Unlike English, many Indian (Hindi, Bangla, Telugu, etc.) languages are free-word-order and are also morphologically rich. It has been suggested that free-word-order languages can be handled better using the dependency based framework than the constituency based one (Bharati et al., 1995). Due to the availability of dependency treebanks, there are several recent attempts at building dependency parsers. Two CoNLL shared tasks (Buchholz and Marsi, 2006; Nivre et al., 2007b) were held aiming at building state-of-the-art dependency parsers for different languages. Recently in two ICON Tools Contest (Husain, 2009; Husain et al., 2010), rule-based, constraint based, statistical and hybrid approaches were explored towards building dependency parsers for three Indian languages namely, Telugu, Hindi and Bangla. In all these efforts, state-of-the-art accuracies are obtained by two data-driven parsers, namely, Malt (Nivre et al., 2007a) and MST (McDonald et al., 2006).

We first explored Malt and MST parsers. Considering pros of both these parsers, we developed a hybrid approach combining the output of these two parsers in an intuitive manner. We report our results on both development and test data provided in the Hindi Shared Task on Parsing at workshop on MT and parsing in Indian Languages, Coling 2012. Our system secured labeled attachment score of 90.66% and 80.77% for gold standard and automatic tracks respectively. These accuracies are 3rd best and 5th best for gold standard and automatic tracks respectively.

In this paper, we give a brief introduction to the Shared Task in Section 2 and related work in Section 3. We describe our approach and settings in Sections 4 and 5 respectively. We present our results and analysis in Section 6. We conclude the paper with future work in Section 7.

2 Shared Task

Hindi Shared Task on Parsing was organized at workshop on MT and parsing in Indian Languages, Coling 2012. As part of the shared task, a part of the Hindi Treebank (HTB) containing gold standard morphological analyses, part-of-speech tags, chunks and dependency relations labeled in the computational paninian framework was released.

2.1 The Task

The task is to assign labeled dependency structures by means of a fully automatic dependency parser. There are two evaluation tracks namely, *gold standard* and *automatic*. In the gold standard track, the input to the system consists of sentence tokens with gold standard morphological analysis, part-of-speech tags, chunks and the additional features. In the automatic track, the input to the system contains only the sentence token and the part-of-speech tags from an automatic tagger. In both the tracks, the parser must output the head and the corresponding dependency relation for each token in the input sentence.

The teams are provided with training and development data containing gold standard morphological analysis (lemma, coarse POS tag, gender, number, person, vibhakti, tense-aspect-modality), part-of-speech tags, chunks, labeled dependency structures along with some additional features such as sentence type, sentence voice etc.

2.2 Data

A subset of the dependency annotated Hindi Treebank (HTB ver-0.5) was released as part of the Shared Task. HTB ver-0.5 is a multi-layered dependency treebank with morphological, part-of-speech and dependency annotations on sentences from news domain corpus acquired from ISI-Kolkata, India. During annotation, the dependency relations are only marked between chunks in the sentence and the words are annotated with POS tags and morphological analysis. The annotations are stored in the Shakti Standard Format. The statistics of the data released for the task are as below

- Training Data – 12041 sentences, 268,093 words
- Development Data – 1233 sentences, 26,416 words
- Testing Data – 1828 sentences, 39,775 words

3 Related Work

In two ICON Tools Contest (Husain, 2009; Husain et al., 2010), different rule-based, constraint based, statistical and hybrid approaches were explored towards building dependency parsers for Indian languages. Ghosh et al. (2009) used a CRF based hybrid method. Nivre (2009), Ambati et al. (2009), and Kosaraju et al. (2010) used Malt Parser and explored the effectiveness of local morphosyntactic features, chunk features and automatic semantic information. Parser settings in terms of different algorithms and features were also explored. Zeman (2009) combined various well known dependency parsers forming a superparser by using a voting method. Yeleti and Deepak (2009) and Kesedi et al. (2010) used a constraint based approach. The scoring function for ranking the base parses is inspired by a graphbased parsing model and labeling. Attardi et al. (2010) used a transition based dependency shift reduce parser (DeSR parser) that uses a Multilayer Perceptron (MLP) classifier with a beam search strategy.

4 Approach

For the experiments reported in this paper we used two data-driven parsers namely, MaltParser (Nivre et al., 2007a), and MST (McDonald et al., 2006).

4.1 Malt Parser

MaltParser is a freely available implementation of the parsing models described in (Nivre et al., 2007a). MaltParser implements the transition-based approach to dependency parsing, which has two essential components:

- A transition system for mapping sentences to dependency trees
- A classifier for predicting the next transition for every possible system configuration

Given these two components, dependency parsing can be realized as deterministic search through the transition system, guided by the classifier. With this technique, parsing can be performed in linear time for projective dependency trees and quadratic time for arbitrary (possibly non-projective) trees.

MaltParser comes with a number of built-in transition systems. Some of the well-known algorithms which gave best performance in previous parsing experiments are Nivre arc-eager, Nivre arc-standard, Covington non-projective, Covington projective. MaltParser also provides options for LIBSVM and LIBLINEAR learner algorithms.

4.2 MST Parser

MSTParser is a freely available implementation of the parsing models described in McDonald et al. (2006). It is a graph-based parsing system in that core parsing algorithms can be equated to finding directed maximum spanning trees (either projective or non-projective) from a dense graph representation of the sentence.

MST uses Chu-Liu-Edmonds Maximum Spanning Tree algorithm for non-projective parsing and Eisner's algorithm for projective parsing. It uses online large margin learning as the learning algorithm (McDonald et al., 2005a).

4.3 Our Approach

McDonald and Nivre (2007) compared the accuracy of MSTParser and MaltParser along a number of structural and linguistic dimensions. They observed that, though the two parsers exhibit indistinguishable accuracies overall, MSTParser tends to outperform MaltParser on longer dependencies as well as those dependencies closer to the root of the tree (e.g., verb, conjunction and preposition dependencies), whereas MaltParser performs better on short dependencies and those further from the root (e.g., pronouns and noun dependencies). Since long dependencies and those near to the root are typically the last constructed in transition-based parsing systems, it was concluded that MaltParser does suffer from some form of error propagation. Similar observations were made by Ambati et al. (2009) for Hindi.

In our approach, we tried to combine both Malt and MST to extract the best out of the both parsers. For this, we first tuned both Malt and MST for Hindi. Details of the settings can be found in Section 5. After we got the best models of Malt and MST, we extracted the output of both the parsers on the development data. We also made a list of long distance labels. We compared the output of Malt and MST. Whenever there is a mismatch between outputs of both the parsers, we checked the dependency label given by the parsers. If MST marked it as a long distance label, then we considered MST's output. Otherwise we considered Malt's output. In this way, we gave more weightage to MST in case of long distance label for which it is best. Similarly, we gave more weightage to Malt in case of short distance labels, as Malt is best at short distance relations. Intuition behind this is that Malt is good at short distance dependencies and MST is good at long distance dependencies. The long distance dependency labels list which we used for our approach are "main", "ccof", "nmod__relc", "rs", "rsym", and "vmod".

5 Settings

We first developed our baseline models using Malt Parser and MST Parser. We CoNLL format of the data in wx-notation for our experiments.

5.1 Data Settings

In case of gold standard track, we explored different features provided in the FEATS column and found that only root, category, vibhatki, TAM and chunk information are useful. Gender, number, person and other information didn't give any improvements. This observation is similar to previous work by Ambati et al. (2010) and Kosaraju et al. (2010). For *automatic* track, as the FEATS column is empty, we used the file provided as it is.

5.2 Parser Settings

For Malt, we explored different parser algorithms for Hindi and found that nivre arc-standard gave better performance over others. In case of learning algorithms, LIBLINEAR gave better performance compared to LIBSVM. Also, LIBLINEAR was very faster than LIBSVM learner.

In case MST, we different options provided by the parser and found that non-projective algorithm and training-k=5, gave best results.

6 Results and Analysis

We considered Malt and MST as baselines. Out Approach performed better than these baselines in all the experiments. Performance of Malt, MST, and Our Approach on development and test data for both gold standard and automatic track are provided in Table 1 and Table 2 respectively. We used standard Labelled Attachment Score (LAS), Un-labeled Attachment Score (USA) and Labeled Score (LS) metrices for our evaluation.

	Gold standard			Automatic		
	LAS	UAS	LS	LAS	UAS	LS
Malt	89.76 %	94.39 %	91.31 %	79.84 %	87.66 %	82.42 %
MST	95.99 %	89.54 %	91.19 %	75.08 %	90.55 %	77.13 %
Our Approach	**95.54 %**	**90.85 %**	**92.52 %**	**81.46 %**	**89.45 %**	**83.62 %**

TABLE 1 – Performance of different systems on development data.

	Gold standard			Automatic		
	LAS	UAS	LS	LAS	UAS	LS
Malt	89.38 %	93.86 %	90.93 %	79.18 %	87.11 %	81.86 %
MST	89.20 %	95.78 %	90.79 %	75.12 %	90.45 %	77.37 %
Our Approach	**90.66 %**	**95.18 %**	**92.28 %**	**80.77 %**	**88.91 %**	**83.03 %**

TABLE 2 – Performance of different systems on test data.

On the test data, Malt and MST gave LAS of 89.38% and 89.20% respectively for gold standard track. Using our approach, we could achieve LAS of 90.66%, which is 1.28% better than the baseline systems. Similarly, in automatic track, Malt and MST gave LAS of 79.18% and 75.12% respectively. Whereas our approach could achieve LAS of 80.77%, which is 1.59% better than the baseline systems.

		Gold standard			Automatic		
		Malt	MST	Our Approach	Malt	MST	Our Approach
Development Data	*k1*	85.65 %	82.56 %	**85.71 %**	66.38 %	55.77 %	**66.85 %**
	k2	**76.00 %**	75.13 %	75.94 %	**66.55 %**	58.81 %	66.47 %
	r6	**91.58 %**	88.46 %	91.48 %	68.97 %	51.61 %	**69.20 %**
	main	79.93 %	**97.73 %**	91.12 %	71.60 %	**95.46 %**	81.23 %
	ccof	90.64 %	**91.37 %**	91.32 %	81.62 %	**85.23 %**	84.58 %
	nmod__ relc	34.95 %	51.28 %	**51.28 %**	22.44 %	25.00 %	**25.80 %**
	rsym	91.79 %	96.30 %	**96.36 %**	83.50 %	**93.92 %**	93.87 %
Test Data	*k1*	85.62 %	83.43 %	**85.72 %**	**65.89 %**	56.92 %	65.88 %
	k2	73.52 %	75.03 %	**73.99 %**	65.88 %	59.24 %	**66.18 %**
	r6	89.99 %	87.88 %	**89.99 %**	**66.89 %**	52.21 %	66.76 %
	main	78.22 %	**97.16 %**	89.93 %	71.80 %	**95.13 %**	81.75 %
	ccof	88.96 %	**91.08 %**	90.77 %	80.06 %	**84.38 %**	83.34 %
	nmod__ relc	42.85 %	48.64 %	**48.64 %**	26.42 %	27.23 %	**28.45 %**
	rsym	91.12 %	96.49 %	**96.51 %**	82.71 %	92.53 %	**92.66 %**

TABLE 3 – Performance of different systems on short distance vs. long distance dependencies on development and test data.

Table 3, gives an overview of how Malt, MST and our approach perform on short distance vs. long distance dependencies. We have taken three short distance dependencies (k1, k2, r6) and four long distance dependencies (main, ccof, nmod__relc, rysm). From Table 3, it is clear that our approach gives similar performance to Malt in case of short distance dependencies. In case of long distance dependencies like "main" and "ccof", MST still gives the best accuracies. But, note that, our system give great improvements, nearly 10% for "main" and 2-3% for "ccof", over Malt for these labels. In case of "nmod__relc", and "rsym" our system gave better results over both Malt and MST. By obtaining similar performance on short distance dependencies and huge improvements on long distance dependencies (by taking MST output) over Malt, we could achieve better accuracies over both the parsers. Taking the fact that Malt is good at short distance dependencies and MST is good at long distance dependencies, into consideration, we developed our system, which outperformed both Malt and MST.

7 Conclusion and Future Work

In this paper, we first explored Malt and MST parsers and developed best models, which we considered as the baseline models for our approach. Considering pros of both these parsers, we developed a hybrid approach combining the output of these two parsers in an intuitive manner. As Malt is good at short distance dependencies and MST is good at long distance dependencies, we gave more weightage to Malt in case of short distance dependencies and gave more weightage to MST in case of long distance dependencies. We showed that a simple system like combining both MST and Malt in an intuitive way, can perform better than both the parsers. We reported our results on both development and test data provided in the Hindi Shared Task on Parsing at workshop on MT and parsing in Indian Languages, Coling 2012. Our system secured labeled attachment score of 90.66% and 80.77% for gold standard and automatic tracks respectively. These accuracies are 3rd best and 5th best for gold standard and automatic tracks respectively.

In our current approach, we combined the output of both Malt and MST to get a better system over both the parsers. In future, we would like to combine both the models in a way similar to McDonald and Nivre (2007). In case of automatic track, we only, used the input provided. For our experience with gold standard track, and from the previous literature, we can say that chunk information and morphological information in the form of vibhakti and TAM plays an important role in Hindi dependency parsing. In future, we would like to experiment with the usefulness of these features in automatic track data by using automatic morphological analyser and chunker for Hindi.

Acknowledgments

We would like to thank Mr. Bharat Ram Ambati for his valuable suggestions and for clarifying the doubts. We would also like to thank the Shared Task organizers for their support and providing the necessary information as and when required.

References

Ambati, B. R., Gadde, P., and Jindal, K. (2009). Experiments in Indian Language Dependency Parsing. In *ICON09 NLP Tools Contest: Indian Language Dependency Parsing*. Hyderabad, India.

Ambati, B. R., Husain, S., Jain, S., Sharma, D. M., and Sangal, R. (2010). Two methods to incorporate 'local morphosyntactic' features in Hindi dependency parsing. In *NAACL-HLT 2010 workshop on Statistical Parsing of Morphologically Rich Languages (SPMRL 2010)*, Los Angeles, CA.

Attardi, G., Rossi, S. D., and Simi, M. (2010). Dependency Parsing of Indian Languages with DeSR. In *ICON-2010 tools contest on Indian language dependency parsing*. Kharagpur, India.

Bharati, A., Chaitanya, V., and Sangal, R. (1995). *Natural Language Processing: A Paninian Perspective*, Prentice-Hall of India, New Delhi, pp. 65-106.

Buchholz, S., and Marsi, E. (2006). CoNLL-X shared task on multilingual dependency parsing. In *Tenth Conf. on Computational Natural Language Learning (CoNLL)*.

Ghosh, A., Bhaskar, P., Das, A. and Bandyopadhyay, S. (2009). Dependency Parser for Bengali: the JU System at ICON 2009. In ICON09 NLP Tools Contest: Indian Language Dependency Parsing. Hyderabad, India.

Husain, S. (2009). Dependency Parsers for Indian Languages. In *ICON09 NLP Tools Contest: Indian Language Dependency Parsing*. Hyderabad, India.

Husain, S., Mannem, P., Ambati, B. and Gadde, P. (2010). The ICON-2010 Tools Contest on Indian Language Dependency Parsing. In *ICON-2010 Tools Contest on Indian Language Dependency Parsing*. Kharagpur, India.

Kesidi, S. R., Kosaraju, P., Vijay, M. and Husain, S. (2010). A Two Stage Constraint Based Hybrid Dependency Parser for Telugu. In *ICON-2010 tools contest on Indian language dependency parsing*. Kharagpur, India.

Kosaraju, P., Kesidi, S. R., Ainavolu, V. B. R., and Kukkadapu, P. (2010). Experiments on Indian Language Dependency Parsing. In *ICON-2010 tools contest on Indian language dependency parsing*. Kharagpur, India.

Mannem., P. (2009). Bidirectional Dependency Parser for Hindi, Telugu and Bangla. In ICON09 NLP Tools Contest: Indian Language Dependency Parsing. Hyderabad, India.

McDonald, R., Crammer, K., and Pereira, F. (2005). Online large-margin training of dependency parsers. In *43rd Annual Meeting on Association for Computational Linguistics*, pages 91–98.

McDonald, R., Lerman, K., and Pereira, F. (2006). Multilingual dependency analysis with a two-stage discriminative parser. In *Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, pp. 216–220.

McDonald, R., and Nivre, J. (2007). Characterizing the errors of data-driven dependency parsing models. In *EMNLP-CoNLL*.

Nivre, J. (2009). Parsing Indian Languages with MaltParser. In *ICON09 NLP Tools Contest:*

Indian Language Dependency Parsing. Hyderabad, India.

Nivre, J., Hall, J., Nilsson, J., Chanev, A., Eryigit, G., Kübler, S., Marinov, S., and Marsi, E. (2007a). MaltParser: A language-independent system for datadriven dependency parsing. *Natural Language Engineering*, 13(2), 95-135.

Nivre, J., Hall, J., Kübler, S., McDonald, R., Nilsson, J., Riedel, S., and Yuret, D. (2007b). The CoNLL 2007 Shared Task on Dependency Parsing. In *CoNLL Shared Task Session of EMNLP-CoNLL*.

Yeleti, M. V., and Deepak, K. (2009). Constraint based Hindi dependency parsing. In ICON09 NLP Tools Contest: Indian Language Dependency Parsing. Hyderabad, India.

Zeman, D. (2009). Maximum Spanning Malt: Hiring World's Leading Dependency Parsers to Plant Indian Trees. In ICON09 NLP Tools Contest: Indian Language Dependency Parsing. Hyderabad, India.