International Journal of Computer Applications (0975 – 8887)
Volume 115 – No. 5, April 2015

Improving Indian Language Dependency Parsing by Combining Transition-based and Graph-based Parsers

B. Venkata Seshu Kumari
Associate professor
Dept of CSE
St. Peter's Engineering College,
Hyderabad, India

R. Rajeswara Rao
Associate Professor
Dept of CSE
JNTU college of Engineering, Vijayanagaram
JNTUK, India

ABSTRACT
We report our dependency parsing experiments on two Indian Languages, Telugu and Hindi. We first explore two most popular dependency parsers namely, Malt parser and MST parser. Considering pros of both these parsers, we develop a hybrid approach combining the output of these two parsers in an intuitive manner. For Hindi, we report our results on test data provided in the for gold standard track of Hindi Shared Task on Parsing at workshop on Machine Translation and parsing in Indian Languages, Coling 2012. Our system secured unlabeled attachment score of 95.2% and labelled attachment score 90.7%. For Telugu, we report our results on test data provided in the ICON 2010 Tools Contest on Indian Languages Dependency Parsing. Our system secured unlabeled attachment score of 92.0% and labelled attachment score 69.5%.

Keywords
Dependency Parsing; Telugu; Hindi; Malt Parser; MST Parser

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 We first explored Malt and MST parsers for parsing Telugu and Hindi. Considering pros of both these. 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 parser (Nivre et al., 2007a) and MST parser (McDonald et al., 2006).

We first explored Malt and MST parsers for parsing Telugu and Hindi. Considering pros of both these parsers, we developed a hybrid approach combining the output of these two parsers in an intuitive manner. For Hindi, we report our results on test data provided in gold standard track of Hindi Shared Task on Parsing at workshop on Machine Translation and parsing in Indian Languages (MTPIL), Coling 2012. Our system secured unlabeled attachment score of 95.2% and labelled attachment score 90.7%. For Telugu, we report our results on test data provided in the ICON 2010 Tools Contest on Indian Languages Dependency Parsing. Our system secured unlabeled attachment score of 92.0% and labelled attachment score 69.5%.

In this paper, we give a brief introduction to the related work in Section 2. We describe the parsers explored in Sections 3. Details about data and parser settings are presented in Section 4. We present our results and analysis in Section 5. We conclude the paper with future work in Section 6.

2. 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 super parser 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 graph based 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.

3. APPROACH
We explored two data-driven parsers Malt parser (Nivre et al., 2007a), and MST parser (McDonald et al., 2006) for our experiments in this paper. In this section, we first describe both these two parsers in detail. Then we explain our approach of combing these two parser to produce better parser output for Telugu and Hindi.

4. MALT PARSER
Malt parser is a freely available implementation of the parsing models described in (Nivre et al., 2007a). Malt parser 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. Malt parser 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. Malt parser also provides options for LIBSVM and LIBLINEAR learner algorithms.

5. MST PARSER

MST parser 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 parser 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).

6. OUR APPROACH

McDonald and Nivre (2007) compared the accuracy of MST parser and Malt parser along a number of structural and linguistic dimensions. They observed that, though the two parsers exhibit indistinguishable accuracies overall, MST parser tends to outperform Malt parser on longer dependencies as well as those dependencies closer to the root of the tree (e.g., verb, conjunction and preposition dependencies), whereas Malt parser 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 Malt parser 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 parsers to extract the best out of the both parsers. For this, we first tuned both Malt parser and MST parser. Details of the settings can be found in Section 4. After we got the best models of Malt and MST parsers, 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 parsers. Whenever there is a mismatch between outputs of both the parsers, we checked the dependency label given by the parsers. If MST parser marked it as a long distance label, then we considered MST parser’s output. Otherwise we considered Malt parser’s output. In this way, we gave more weightage to MST parser in case of long distance labels for which it is best. Similarly, we gave more weightage to Malt parser in case of short distance labels, as Malt parser is best at short distance relations. Intuition behind this is that Malt parser is good at short distance dependencies and MST parser is good at long distance dependencies. For Telugu, the long distance dependency labels list which we used for our approach are “main”, “ccof”, “nmod__relc”, “adv”, and “vmod”. The list for Hindi is “main”, “ccof”, “nmod__relc”, “rs”, “rsym”, and “vmod”.

7. TELUGU: DATA AND SETTINGS

7.1 Data

For our experiments, we used Telugu data from ICON 2010 Tools contest. Data released has both fine-grained and coarse-grained versions of dependency labels. We used fine-grained version here. This data was annotated using the Computational Paninian Grammar (Bharati et al., 1995). The annotation scheme based on this grammar has been described in Begum et al. (2008) and Bharati et al. (2009). Subject and direct object equivalent dependency in this framework are kartha karaka (k1) and karma karaka (k2). Table 1 shows the training, development and the testing data sizes the Telugu treebank. Statistics on sentence count, word count and average sentence length are provided in this table.

7.2 Parser Settings

As the training data size is small, we merged training and development data and did 10-fold cross validation for tuning the parameters of the parsers and for feature selection. Best settings obtained using cross-validated data are applied on test set. In case of Malt parser, liblinear learner and arc-eager parsing algorithm consistently gave better performance. For feature model we tried best feature settings of the same parser on different languages in CoNLL and ICON shared tasks (Hall et al., 2007; Husain 2009; Husain et al., 2010) and applied the best feature model.

In case of MST, order=2, training-k=1 and non-projective algorithm gave the best results. It was difficult to do feature tuning with MST parser as it do not provide nice options similar to Malt parser. We explored different features in labelling module of the MST parser and selected the settings which gave best results on 10-fold cross-validation.

Table 1 – Telugu treebank statistics

Type	Sent Count	Word Count	Avg. sent_length
Train	1,400	7602	5.43
Devel	150	839	5.59
Test	150	836	5.57

8. HINDI: DATA AND SETTINGS

8.1 Data

We used gold standard track of Hindi Shared Task on Parsing at Coling 2012 MTPIL workshop. Similar to Telugu, this data was annotated using the Computational Paninian Grammar (Bharati et al., 1995). The annotation scheme based on this grammar has been described in Begum et al. (2008) and Bharati et al. (2009). Subject and direct object equivalent dependency in this framework are kartha karaka (k1) and karma karaka (k2). 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). Table 2 shows the training, development and the testing data sizes the Telugu treebank. Statistics on sentence count, word count and average sentence length are provided in this table.

8.2 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.

Table 2 – Hindi treebank statistics

Type	Sent Count	Word Count	Avg. sent_length
Train	12,041	268,093	22.27
Devel	1,233	26,416	21.42
Test	1,828	39,775	21.76

9. EXPERIMENTS AND RESULTS

Performance of Malt parser and MST parser on test data are provided in Table 3. We used standard Labelled Attachment Score (LAS), Un-labeled Attachment Score (USA) and Labeled Score (LS) metrices for our evaluation.

Table 3 – Performance of different systems on test data.

Approach	*Telugu*			*Hindi*		
	UAS	*LAS*	*LS*	*UAS*	*LAS*	*LS*
Malt	91.8%	**70.0%**	**72.3%**	93.9%	89.4%	90.9%
MST	90.0%	67.1%	68.6%	**95.8%**	89.2%	90.8%
Our Approach	**92.0%**	69.5%	71.8%	95.2%	**90.7%**	**92.3%**

Table 4 – Performance of Malt parser, MST parser and our approach on top five dependencies in the test data.

Approach	*Telugu*			*Hindi*		
	Malt Parser	*MST Parser*	*Our Approach*	*Malt Parser*	*MST Parser*	*Our Approach*
Main	97.0	95.3	**97.0**	78.2	**97.2**	89.9
k1	**63.0**	59.4	60.3	85.6	83.4	**85.7**
k2	58.8	62.7	**62.8**	73.5	**75.0**	74.0
ccof	83.1	74.4	**83.8**	89.0	**91.1**	90.8
r6	81.5	53.9	**81.5**	90.0	87.9	**90.0**

9.1 Analysis: Telugu

On the test data, Malt parser and MST parser gave UAS of 91.8% and 90.0% respectively. Using our approach, we could achieve UAS of 92.0%, which is better than both the baseline systems. Similarly, Malt parser and MST parser gave LAS of 70.0% and 67.1% respectively. Our approach gave an LAS of 69.5%. Though it is slightly lower than Malt parser's performance, it is much higher than the MST parser's performance. As performance of MST parser is much lower compared to Malt, there is only slight improvement in case of UAS and slight decrement in case of LAS. We hope that if we can improve MST parser's performance then we could achieve much better improvements with our approach. As the training data is very low, and also as Telugu is agglutinative language, LAS for the all the systems is very low. With more training data and specialized techniques for handling agglutinative languages like Telugu, we can achieve better results in LAS.

Table 4, gives an overview of the performance of Malt parser, MST parser and our approach on the top five dependencies. Results show that our approach outperforms both Malt parser and MST parser on major dependencies. We couldn't get much improvement in case of k2. We believe this could be because of low performance of MST parser. 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 parser is good at short distance dependencies and MST parser is good at long distance dependencies, into consideration, we developed our system, which outperformed both Malt and MST parsers.

9.2 Analysis: Hindi

On the test data, Malt parser and MST parser gave UAS of 93.9% and 95.8% respectively. Using our approach, we could achieve UAS of 95.2%, which is better than Malt but slightly lower than Malt. Malt parser and MST parser gave LAS of 89.4% and 89.2% respectively. Our approach gave an LAS of 90.7% which is better than both the baselines. As performance of Malt parser is much lower compared to MST, there is only slight decrement in case of UAS. We hope that if we can improve Malt parser's performance then we could achieve much better improvements with our approach.

Table 4, gives an overview of the performance of Malt parser, MST parser and our approach on the top five dependencies. Results show that our approach outperforms Malt parser in all the cases and MST parser on few dependencies. We couldn't get much improvement in case of main and k1 as MST is far better at these labels compared to Malt. 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 parser is good at short distance dependencies and MST parser is good at long distance dependencies, into consideration, we developed our system, which outperformed both Malt and MST parsers.

10. 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 parser is good at short distance dependencies and MST parser is good at long distance dependencies, we gave more weightage to Malt parser in case of short distance dependencies and gave more weightage to MST parser in case of long distance dependencies. We showed that a simple system like combining both MST and Malt parsers in an intuitive way, can perform better than both the parsers. For Hindi, we reported our results on test data provided in the for gold standard track of Colling 2012 MTPIL workshop. Our system secured unlabeled attachment score of 95.2% and labelled attachment score 90.7%. For Telugu, we report our results on test data provided in the ICON 2010 Tools Contest on Indian Languages Dependency Parsing. Our system secured unlabeled attachment score of 92.0% and labelled attachment score 69.5%.

In our current approach, we combined the output of both Malt and MST parsers 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). We also would like to explore the approach of voting similar to Zeman (2009) by taking advantages of different available parsers. We also plan to explore the usefulness of large un-annotated data using self-training and co-training techniques to improve the performance of the Indian Language dependency parsers.

10.1 Acknowledgment

We would like to thank Language Technologies Research Institute (LTRC), International Institute of Information Technology, Hyderabad (IIIT-H) for providing the data and previous best settings for the parser.

11. REFERENCES

[1] Bharat Ram Ambati, Phani Gadde, and Karan Jindal. 2009. Experiments in Indian Language Dependency Parsing. In Proceedings of the ICON09 NLP Tools Contest: Indian Language Dependency Parsing, pp 32-37.

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

[3] Rafiya Begum, Samar Husain, Arun Dhwaj, Dipti Misra Sharma, Lakshmi Bai and Rajeev Sangal. 2008. Dependency annotation scheme for Indian languages. In Proceedings of IJCNLP-2008.

[4] Akshar Bharati, Vineet Chaitanya, and Rajeev Sangal. 1995. Natural Language Processing: A Paninian Perspective, Prentice-Hall of India, New Delhi, pp. 65-106.

[5] Akshar Bharati, Samar Husain, and Rajeev Sangal. 2008. A Two-Stage Constraint Based Dependency Parser for Free Word Order Languages. In Proceedings of the COLIPS International Conference on Asian Language Processing 2008 (IALP), Chiang Mai, Thailand.

[6] Akshar Bharati, Dipti Misra Sharma, Samar Husain, Lakshmi Bai, Rafiya Begum, and Rajeev Sangal. 2009. AnnCorra: TreeBanks for Indian Languages, Guidelines for Annotating Hindi TreeBank (version 2.0). http://ltrc.iiit.ac.in/MachineTrans/research/tb/DS-guidelines/DS-guidelines-ver2-28-05-09.pdf.

[7] Sabine Buchholz and Erwin Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In Tenth Conf. on Computational Natural Language Learning (CoNLL).

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

[9] E. Hajicova. 1998. Prague Dependency Treebank: From Analytic to Tectogrammatical Annotation. In Proc. TSD'98.

[10] Richard Hudson. 1984. Word Grammar, Basil Blackwell, 108 Cowley Rd, Oxford, OX4 1JF, England.

[11] Samar Husain. 2009. Dependency Parsers for Indian Languages. In ICON09 NLP Tools Contest: Indian Language Dependency Parsing. Hyderabad, India.

[12] Samar Husain, Prashanth Mannem, Bharat Ambati and Phani Gadde. 2010. The ICON-2010 Tools Contest on Indian Language Dependency Parsing. In ICON-2010 Tools Contest on Indian Language Dependency Parsing. Kharagpur, India.

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

[14] Prudhvi Kosaraju, Sruthilaya Reddy Kesidi, Vinay Bhargav Reddy Ainavolu and Puneeth Kukkadapu. 2010. Experiments on Indian Language Dependency Parsing. In ICON-2010 tools contest on Indian language dependency parsing. Kharagpur, India.

[15] M. Marcus, B. Santorini, and M.A. Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank, Computational Linguistics

[16] Ryan McDonald, Kevin Lerman, and Fernando Pereira. 2006. Multilingual dependency analysis with a two-stage discriminative parser. In Tenth Conference on Computational Natural Language Learning (CoNLL-X), pp. 216–220.

[17] Ryan McDonald and Joakim Nivre. 2007. Characterizing the errors of data-driven dependency parsing models. In Proceedings of the Conference on Empirical Methods in Natural Language Processing and Natural Language Learning.

[18] Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. In Proceedings of the 8th International Workshop on Parsing Technologies (IWPT), pages 149–160.

[19] Joakim Nivre. 2008. Algorithms for deterministic incremental dependency parsing. Computational Linguistics, 34(4):513–553.

[20] Joakim Nivre. 2009. Parsing Indian Languages with MaltParser. In ICON09 NLP Tools Contest: Indian Language Dependency Parsing. Hyderabad, India.

[21] Joakim Nivre and Jens Nilsson. 2005. Pseudo-projective dependency parsing. In ACL '05: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics, pages 99–106, Ann Arbor, Michigan.

[22] Joakim Nivre, Johan Hall, Sandra Kubler, Ryan McDonald, Jens Nilsson, Sebastian Riedel, and Deniz Yuret. 2007a. The CoNLL 2007 Shared Task on Dependency Parsing. In Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007. Prague, Czech Republic, 915–932.

[23] Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Gulsen Eryigit, Sandra Kubler, Svetoslav Marinov, and Erwin Marsi. 2007b. Malt parser: A language-independent system for data-driven dependency parsing. Natural Language Engineering 13, 2 (2007), 95–135.

[24] Sebastian Riedel , Ruket Cakici and Ivan Meza-Ruiz. 2006. Multi-lingual Dependency Parsing with Incremental Integer Linear Programming. In Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X).

[25] *Stuart M. Shieber. 1985. Evidence against the context-freeness of natural language. In Linguistics and Philosophy, p. 8, 334–343.*

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

[27] D. Zeman. 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.