Volume 5, Issue 9, September 2015 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

A Hybrid Dependency Parsing Approach for Telugu Language

[1]B.Venkata Seshu Kumari, [2]R. Rajeswara Rao
[1]Asst. Prof, Dept of CSE, St. Peter's Engineering College, Hyderabad, India
[2]Assoc.Prof, Dept of CSE, JNTU, Vijayanagaram, India

Abstract— In this paper we report our Telugu dependency parsing experiments. 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 test data provided in the ICON 2010 Tools Contest on Indian Languages Dependency Parsing. Our system secured best unlabeled attachment score of 91.99%.

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

I. 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 [4]. Due to the availability of dependency treebanks, there are several recent attempts at building dependency parsers. Two CoNLL shared tasks [7][23] were held aiming at building state-of-the-art dependency parsers for different languages. Recently in two ICON Tools Contest [11][12], 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 [23][17].

We first explored Malt and MST parsers for parsing Telugu. 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 the test data provided in the ICON 2010 Tools contest on Indian Language Dependency Parsing. Our system secured best unlabeled attachment score of 91.99%.

In this paper, we give a brief introduction to the related work in Section 2. We describe our approach 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.

II. RELATED WORK

In two ICON Tools Contest [12][13], different rule-based, constraint based, statistical and hybrid approaches were explored towards building dependency parsers for Indian languages. [9] used a CRF based hybrid method. [21], [1], and [14] 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. [28] combined various well known dependency parsers forming a super parser by using a voting method. [26] and [14] used a constraint based approach. The scoring function for ranking the base parses is inspired by a graph based parsing model and labeling. [2] used a transition based dependency shift reduce parser (DeSR parser) that uses a Multilayer Perceptron (MLP) classifier with a beam search strategy.

III. APPROACH

We explored two data-driven parsers Malt parser [23], and MST parser [17] 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.

A. Malt Parser

Malt parser is a freely available implementation of the parsing models described in [23]. 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.

B. MST Parser

MST parser is a freely available implementation of the parsing models described in [17]. 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.

C. Our Approach

Ref [18] 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 [1] 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 for Telugu. 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. The long distance dependency labels list which we used for our approach are "main", "ccof", "nmod__relc", "adv", and "vmod".

IV. DATA AND SETTINGS

A. 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 [4]. The annotation scheme based on this grammar has been described in [3] and [5]. 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.

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

B. 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 [23][12][13] 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 doesn't 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.

V. EXPERIMENTS AND RESULTS

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

Table 2 – Performance of different systems (Malt parser, MST parser and Our Approach) on Telugu dependency treebank test data.

Parser	*LAS*	*UAS*	*LS*
Malt Parser	**70.12 %**	91.82 %	**71.95 %**
MST Parser	65.61 %	87.98 %	68.28 %
Our Approach	69.62 %	**91.99 %**	71.29 %

On the test data, Malt parser and MST parser gave UAS of 91.82% and 87.98% respectively. Using our approach, we could achieve UAS of 91.99%, which is better than both the baseline systems. Similarly, Malt parser and MST parser gave LAS of 70.12% and 65.61% respectively. Our approach gave an LAS of 69.62%. 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 3 – Performance of different systems on top eight dependencies in Telugu dependency treebank test data.

Labels	*Malt Parser*	*MST Parser*	*Our Approach*
main	96.69	96.00	**97.33**
k1	**63.20**	58.87	62.56
k2	**59.63**	57.48	59.50
vmod	63.42	65.11	**65.21**
ccof	**86.08**	76.32	84.21
k7t	**65.22**	51.16	61.90
Adv	78.26	**80.00**	76.60
r6	81.48	42.86	**81.48**

Table 3, gives an overview of the performance of Malt parser, MST parser and our approach on the top eight dependencies in Telugu dependency treebank. Results show that our approach outperforms both Malt parser and MST parser on major dependencies. We could get much improvement in case of labels like k1 and 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.

Table 4 – Performance of different systems on top different distance ranges in Telugu dependency treebank test data.

Labels	*Malt*	*MST*	*Our Approach*
to_root	96.69	96.00	**97.33**
1-5	**98.53**	97.86	98.42
6-10	**36.36**	25.00	**36.36**
>10	00.00	00.00	00.00

VI. 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. We reported our results on the test data provided in the ICON 2010 Tools contest on Indian Language Dependency Parsing. Our system secured best unlabeled attachment score of 91.99%.

In our current approach, we combined the output of both Malt and MST parsers to get a better system over both the parsers. Table 4 shows Performance of different systems on top different distance ranges in Telugu dependency treebank test data.In future, we would like to combine both the models in a way similar to [18]. We also would like to explore the approach of voting similar to [28] 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 Telugu dependency parsers.

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.

REFERENCES

[1] B. R. Ambati, P. Gadde and K. 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] R. Begum, S. Husain, A. Dhwaj, D. Sharma, L. Bai, and R. Sangal. 2008. Dependency annotation scheme for Indian languages. In Proceedings of IJCNLP-2008.

[4] A. Bharati, V. Chaitanya and R. Sangal. 1995. Natural Language Processing: A Paninian Perspective, Prentice-Hall of India, New Delhi, pp. 65-106.
A. Bharati, S. Husain, D. M. Sharma, and R. 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.

[5] A. Bharati, D. M. Sharma, S. Husain, L. Bai, R. Begum and R. 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-guidelinesver2-28-05-09.pdf.

[6] S. Buchholz and E. Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In Proc. of the Tenth Conf. on Computational Natural Language Learning (CoNLL).

[7] 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.

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

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

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

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

[12] 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.

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

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

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

[16] R. McDonald and J. Nivre. 2007. Characterizing the errors of data-driven dependency parsing models. In Proc. of EMNLP-CoNLL.

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

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

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

[20] J. Nivre, and J. 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.

[21] J. Nivre, J. Hall, S. Kubler, R. McDonald, J. Nilsson, S. Riedel and D. Yuret. 2007a. The CoNLL 2007 Shared Task on Dependency Parsing. In Proceedings of EMNLP/CoNLL-2007.

[22] J. Nivre, J. Hall, J. Nilsson, A. Chanev, G. Eryigit, S. Kübler, S. Marinov and E Marsi. 2007b. MaltParser: A language-independent system for data-driven dependency parsing. Natural Language Engineering, 13(2), 95-135.

[23] S. Riedel, R. Çakıcı, and I. 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).

[24] S. M. Shieber. 1985. Evidence against the contextfreeness of natural language. In Linguistics and Philosophy, p. 8, 334–343.

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

[26] 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.

