Ensembling Various Dependency Parsers: Adopting Turbo Parser for Indian Languages

Puneeth Kukkadapu[1], Deepak Kumar Malladi[1] and Aswarth Dara[2]
(1) Language Technologies Research Center, IIIT Hyderabad, Gachibowli, Hyderabad-32, India
(2) CNGL, School of Computing, Dublin City University, Dublin, Ireland
{puneeth.kukkadapu, deepak.malladi}@research.iiit.ac.in,
adara@computing.dcu.ie

Abstract
In this paper, we describe our experiments on applying combination of Malt, MST and Turbo Parsers for Hindi dependency parsing as part of a shared task at MTPIL 2012 Workshop, COLING 2012. We explore the usage and adoption of the recently released Turbo Parser for parsing Indian languages. Various configurations of each parser are explored before combination in order to adjust them for two different settings of the data (with gold-standard and automatic Part-Of-Speech tags). We achieved a best result of 96.50% unlabeled attachment score (UAS), 92.90% labeled accuracy (LA), 91.49% labeled attachment score (LAS) using voting method on data with gold POS tags. In case of data with automatic POS tags, we achieved a best result of 93.99% UAS, 90.04% LA and 87.84% LAS respectively.

Keywords: Hindi Dependency Parsing, Voting, Blending and Turbo Parser.

1 Introduction

Parsing helps in understanding the relations between words in a sentence. It plays an important role in a lot of applications like machine translation, word sense disambiguation, search engines, dialogue systems etc. Parsers are mainly classified into two categories - grammar driven and data driven. The combination of these approaches led to the development of hybrid parsers. In recent years, there is a lot of interest in data driven dependency parsing due to the availability of annotated corpora thereby building accurate parsers (Nivre et al., 2006; McDonald et al., 2005; Mannem and Dara, 2011).

Majority of the Indian languages are morphologically rich in nature. They pose various challenges like presence of large number of inflectional variations, relatively free word order and non-projective. Previous research showed that some of these challenges can be better handled by using dependency based annotation scheme rather than using phrase based annotation scheme (Hudson, 1984; Shieber, 1985; Bharati et al., 1995). As a result of this, dependency annotation for Hindi is based on Paninian framework for building the treebank (Begum et al., 2008). Considering the above challenges and due to the increase in the availability of the annotated data, there is a need to develop good quality dependency parsers for Indian Languages.

In this paper, we explore the approaches related to the combination of different parsers. Since these approaches are well studied in the literature, we will focus on providing a detailed analysis of the various features used and also about the various errors caused by the dependency parsers.

The rest of the paper is organized as follows: we will describe the related work in this area in

Proceedings of the Workshop on Machine Translation and Parsing in Indian Languages (MTPIL-2012), pages 179–184,
COLING 2012, Mumbai, December 2012.

Section 2 and our approach is described in Section 3 . Section 4 lists the experiments conducted and presents the results obtained. Section 5 ends the paper with conclusions and scope of future work in this direction.

2 Related Work

There are previously two instances of the task on dependency parsing for Indian languages (Husain, 2009; Husain et al., 2010). The previous best performed systems on both the times used the transition based Malt Parser (Ambati et al., 2009; Kosaraju et al., 2010) for their experiments exploring various different features and also reported the results of MST parser (Ambati et al., 2009). Kolachina et al. (2010) used the approach of blending within various configurations of the malt parser and (Zeman, 2009) used the voting approach of combining multiple parsing systems into a hybrid parser for this task. Abhilash and Mannem (2010) used the bidirectional parser (Shen and Joshi, 2008). There, the approach does a best first search for every sentence and selects the most confident relation at each step with out following a specific direction (either from left-to-right or right-to-left). This year also the shared task is on Dependency Parsing with significantly higher data than the previous two parsing contests and this time it has been limited only to Hindi language excluding previously explored Telugu and Bangla languages.

3 Our Approach

For this shared task on Hindi dependency parsing, we tried the combination of various parsing systems using two different methods i.e., simple voting (Zeman, 2009) and blending method (Sagae and Lavie, 2006).

In case of voting, once the outputs from all the parsing systems are obtained, each dependency relation that has the maximum number of votes from the various systems are included in the output. In case of a tie, the dependency relation predicted by the high accurate parser is picked in the final output. The one drawback that is inherent in the voting method is that the dependency tree resulted for each sentence may not be fully connected as we are including each dependency relation at a time rather including the whole dependency tree. In order to mitigate this drawback, the concept of blending has been introduced by (Sagae and Lavie, 2006) and it has been used in the winning team of CONLL-XI Shared Task (Hall et al., 2007) and the software has been released as MaltBlender. In this approach, a graph is built for the dependency relations obtained from the various outputs and then it selects a maximum spanning tree out of it. The tool also provides various options for selecting the weights for different parsers and these weights are determined by tuning on the development data set.

In this work, we used Malt, MST and Turbo parsers for the parser combination using the above mentioned two approaches. To the best of our knowledge, the process of adopting Turbo parser for Indian languages has not been explored previously.

The number of features a dependency parser uses are typically huge. Selection of features has a lot of impact on both the run-time complexity and also on the performance of a parser. We tried various uni-gram and bi-gram features related to words, lemmas, POS tags, Coarse POS tags, vibhakti (post-positional marker), TAM (tense, aspect and modality) and other available morphological information. In addition to these features, we also used chunk features like chunk head, chunk distance and chunk boundary information. Finally we also experimented with some clause-based features like head/child of a clause, clausal boundary information. Turbo parser (Martins et al., 2010) uses the concept of supported and unsupported features to mitigate the effect of having large number of features to some extent.

4 Experiments and Results

4.1 Data

The training and development data sets was released by the organizers as part of the shared task in two different settings. One being the manually annotated data with POS tags, Chunks and other information such as gender, number, person etc. whereas the other one contains only automatic POS tags without any other information. Training set contains 12,041 sentences (2,68,093 words) and development data set contains 1233 sentences (26,416 words). The size of the data set is significantly higher when compared to the previous shared tasks on Hindi Dependency Parsing (Husain, 2009; Husain et al., 2010).

The test data set contains 1828 sentences (39,775 words). For the final system, we combined the training and development data sets into one data set and used this set for the training. We tried different types of features as mentioned above and selected the best feature set by tuning it on the development data set. The parser settings and feature set related to each parser are explained in the section 4.2

4.2 Parser Settings

4.2.1 Malt

Malt parser provides two learning algorithms LIBSVM and LIBLINEAR. It also gives various options for parsing algorithms and we have experimented on nivre-eager, nivre-standard and stack-proj parsing algorithms. Finally, we chose *nivre-standard* parsing algorithm and *LIBSVM* learning algorithm options by tuning it on the development data set.

4.2.2 MST

The setting that performed best for MST parser is second order non-projective with beam width (k-best parses) of 5 and default iterations of 10. The tuning of MST parser in second order non-projective is hard since it is computationally intensive.

4.2.3 Turbo

Turbo parser provides three settings for training: basic, standard and full. We experimented with all three models for both data settings and consistently the Turbo parser in full mode performs the better and it trains a second-order non-projective parser.

For the two different data settings, we used the same feature set used in (Martins et al., 2010). For the gold-standard POS data setting, we added chunk based features as previously mentioned. We also experimented with the clause based features but we didn't get much performance gain using them.

4.3 Evaluation Results

Table 1 lists the accuracy with gold-standard POS tags setting of the data using various parsers. It also lists the accuracy obtained when the parsers are combined using simple voting method and an intelligent blending method. It can be observed from the table that Turbo parser performs well in all the evaluation metrics in terms of using a single parser whereas voting method performs well overall. The results submitted for this data setting using Turbo parser was ranked second in terms of

LAS, LS but first in UAS. If we take the voting results, then it will be ranked first in terms of all the evaluation metrics when compared to the other systems submitted for this shared task.

Table 2 gives the accuracy for the test data with automatic POS tags. The results on this data setting using Turbo parser are higher among all the methods we tried. It was ranked first among the systems submitted in the shared task. The voting and blending systems did not perform well than Turbo Parser because of the lower accuracies from Malt and MST. It can be inferred from the results that the voting and blending systems benefit if the accuracies produced by the parsers are comparable to each other. On the other hand if the difference is very high then it will hurt their performance.

With gold-standard Part-Of-Speech Tags			
Method	*UAS*	*LA*	*LAS*
Malt	93.32%	90.56%	88.86%
MST	94.88%	88.13%	86.45%
Turbo	96.37%	92.14%	90.83%
Voting	**96.50**%	**92.90**%	**91.49**%
Blending	96.34%	92.83%	91.49%

Table 1: Accuracies on gold-standard data. UAS, LA, LAS denote the Unlabeled Attachment score, Labeled Accuracy score and Labeled Attachment score respectively.

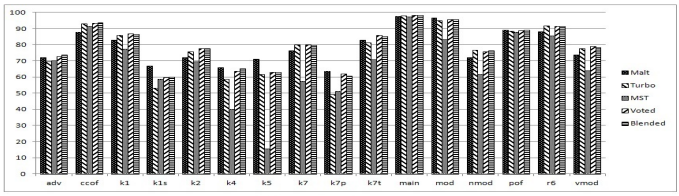
With automatic Part-Of-Speech Tags			
Method	*UAS*	*LA*	*LAS*
Malt	81.23%	76.76%	73.69%
MST	91.02%	84.76%	82.55%
Turbo	**93.99**%	**90.04**%	**87.84**%
Voting	93.24%	89.01%	86.62%
Blending	93.47%	89.15%	87.05 %

Table 2: Accuracies on data annotated with automatic POS tags

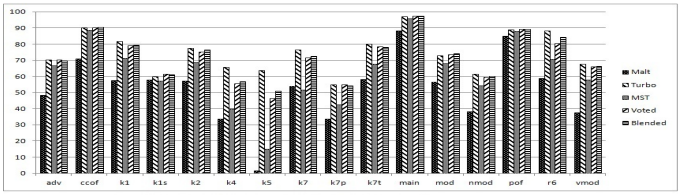
4.4 Error Analysis

The presence of 20.4% non-projective sentences (469 arcs) in the test data reflects the complexity of developing a good quality parser for Hindi. We observed the performance of the individual parsers on these specific arcs, the accuracy has ranged from 27-28%. One trivial reason involves the inherent complexity and other might be due to the less amount of training examples related to non-projectivity.

The gold-standard data contains the chunk information marked for every word i.e., whether it is the head or child of a particular chunk. The dependency relations within a chunk are called intra-chunk dependency relations whereas across the chunks are called inter-chunk dependency relations. Figure 4.4 shows the accuracies of the dependency labels (that are frequent in the data) for inter-chunk dependency relations and these are ones that are hard to predict. It can be observed in part (a) of the figure 4.4 that if the accuracies of the individual parsers are comparable then we are getting a good result in the voting and blending approaches. In part (b), we can see a drop in the accuracies on the voting and blending approaches since the difference in accuracies between the parsers is high. The analysis of dependency labels on intra-chunk dependencies has not been shown since there is not much difference in the accuracies between the parsing models.



a) Inter-chunk dependency label accuracy on gold POS data



b) Inter-chunk dependency label accuracy on auto POS data

5 Conclusion

In this work, we applied combination of Malt, MST and Turbo parsers using voting and blending approaches. We observed that the voting accuracy improves when the accuracies of the individual parsers are comparable to each other. Our system achieved a best result of 96.50% UAS, 92.90% LA, 91.49% LAS using voting method on data with gold POS tags. In case of data with automatic POS tags, we achieved a best result of 93.99% UAS, 90.04% LA and 87.84% LAS. Our future work involves in looking at the specific cases in case of non-projective errors and figuring out a way to find out the syntactic and semantic cues to reduce the impact of these errors on the overall accuracy.

6 Acknowledgements

This work was partly supported by Science Foundation Ireland (Grant No. 07/CE/I1142) as part of the Centre for Next Generation Localisation (www.cngl.ie) at Dublin City University.

7 References

References

Abhilash, A. and Mannem, P. (2010). Bidirectional dependency parser for indian languages. *Proceedings of the ICON10 NLP Tools Contest: Indian Language Dependency Parsing*.

Ambati, B., Gadde, P., and Jindal, K. (2009). Experiments in indian language dependency parsing. *Proceedings of the ICON09 NLP Tools Contest: Indian Language Dependency Parsing*, pages 32–37.

Begum, R., Husain, S., Dhwaj, A., Sharma, D., Bai, L., and Sangal, R. (2008). Dependency annotation scheme for indian languages. In *Proceedings of International International Joint Conference on Natural Language Processing*.

Bharati, A., Chaitanya, V., Sangal, R., and Ramakrishnamacharyulu, K. (1995). *Natural language processing: A Paninian perspective*. Prentice-Hall of India.

Hall, J., Nilsson, J., Nivre, J., Eryigit, G., Megyesi, B., Nilsson, M., and Saers, M. (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.

Hudson, R. (1984). *Word grammar*. Blackwell Oxford.

Husain, S. (2009). Dependency parsers for indian languages. *Proceedings of the ICON09 NLP Tools Contest: Indian Language Dependency Parsing*.

Husain, S., Mannem, P., Ambati, B., and Gadde, P. (2010). The icon-2010 tools contest on indian language dependency parsing. *Proceedings of the ICON-2010 Tools Contest on Indian Language Dependency Parsing, ICON*, 10:1–8.

Kolachina, S., Kolachina, P., Agarwal, M., and Husain, S. (2010). Experiments with maltparser for parsing indian languages. *Proceedings of the ICON-2010 tools contest on Indian language dependency parsing. Kharagpur, India*.

Kosaraju, P., Kesidi, S., Ainavolu, V., and Kukkadapu, P. (2010). Experiments on indian language dependency parsing. *Proceedings of the ICON10 NLP Tools Contest: Indian Language Dependency Parsing*.

Mannem, P. and Dara, A. (2011). Partial parsing from bitext projections. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1597–1606.

Martins, A., Smith, N., Xing, E., Aguiar, P., and Figueiredo, M. (2010). Turbo parsers: Dependency parsing by approximate variational inference. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 34–44.

McDonald, R., Pereira, F., Ribarov, K., and Hajič, J. (2005). Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing (EMNLP)*, pages 523–530.

Nivre, J., Hall, J., and Nilsson, J. (2006). Maltparser: A data-driven parser-generator for dependency parsing. In *Proceedings of LREC*, volume 6, pages 2216–2219.

Sagae, K. and Lavie, A. (2006). Parser combination by reparsing. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pages 129–132.

Shen, L. and Joshi, A. (2008). Ltag dependency parsing with bidirectional incremental construction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 495–504.

Shieber, S. (1985). Evidence against the context-freeness of natural language. *Linguistics and Philosophy*, 8(3):333–343.

Zeman, D. (2009). Maximum spanning malt: Hiring world's leading dependency parsers to plant indian trees. *Proceedings of the ICON09 NLP Tools Contest: Indian Language Dependency Parsing*, page 19.