

# Exploring self training for Hindi dependency parsing

Rahul Goutam  
Language Technologies Research Centre  
IIIT-Hyderabad, India  
[rahul.goutam@research.iiit.ac.in](mailto:rahul.goutam@research.iiit.ac.in)

Bharat Ram Ambati  
Language Technologies Research Centre  
IIIT-Hyderabad, India  
[ambati@research.iiit.ac.in](mailto:ambati@research.iiit.ac.in)

## Abstract

In this paper we explore the effect of self-training on Hindi dependency parsing. We consider a state-of-the-art Hindi dependency parser and apply self-training by using a large raw corpus. We consider two types of raw corpus, one from same domain as of training and testing data and the other from different domain. We also do an experiment, where we add small gold-standard data to the training set. Comparing these experiments, we show the impact of adding small, but gold-standard data to training data versus large, but automatically parsed data on Hindi parser.

## 1 Introduction

Parsing morphologically rich free-word-order languages like Czech, Hindi, Turkish, etc., is a challenging task. Unlike English, most of the parsers for such languages have adopted the dependency grammatical framework. It is well known that for these languages, dependency framework is better suited (Shieber, 1985; Mel'čuk, 1988, Bharati et al., 1995). Due to the availability of annotated corpora in recent years, data driven dependency parsing has achieved considerable success. In spite of availability of annotated treebanks, state-of-the-art parsers for these languages have not reached the performance obtained for English (Nivre et al., 2007a). Frequently stated reasons for low performance are small treebank size, complex linguistic phenomenon, long distance dependencies, and non-projective structures (Nivre et al., 2007a; Nivre et al., 2007b; Bharati et al., 2008).

In this paper, we try to address the problem of small treebank size. We have lots of un-annotated data. One way to increase treebanks' size is to manually annotate this data. But it is very time consuming task. Other way is to automatically parse this data and consider highly reliable parses. But, what criteria should be used

for extracting reliable parses is a really challenging task. In this paper, we explore a bootstrapping technique called self training and see its impact on dependency parsing accuracy. We consider a state-of-the-art Hindi dependency parser and analyze its performance using self-training. We consider two types of raw corpus, one from the same domain as of training and testing data and the other from a different domain. We also show the impact of adding small, but gold-standard data to training data versus large, but automatically parsed data on Hindi dependency parsing.

The paper is arranged as follows. In section 2, we describe the related work in this field. In section 3, we present the current state-of-the-art of Hindi dependency parser. Section 3, talks about different experiments conducted and presents the results. We conclude with possible future work in section 4.

## 2 Related Work

In this section, we briefly describe major works on bootstrapping using statistical parsers.

Steedman et al. (2003) did experiments to show that raw data can be used to improve the performance of statistical parsers by bootstrapping. Although their main focus was on co-training between two statistical parsers, they have also performed self-training for each parser but the results are not that promising with self-training. They have also done cross-genre experiments to show that co-training is beneficial even when the seed set was from a different domain compared to the raw data.

Reichert and Rappoport (2007) also perform similar cross-genre experiments to improve the quality of their parser and to adapt the parser to a different domain. They have also reported significant reduction in annotation cost and amount of work because only small amount of manually annotated seed data was used.

McClosky et al. (2006) used a two phase parser-reranker system for self-training using readily available raw data. In their approach, instead of adding the raw data in steps, they have added the entire data in one go. They have reported significant improvement in accuracy over the previous state-of-the-art accuracy for Wall Street Journal parsing. They have also performed sentence length analysis to show that there is a general improvement in intermediate-length sentences, but no improvement at the extremes.

All the above mentioned works are on phrase structure parsing of English. There is an attempt at exploring usefulness of large raw corpus for dependency parsing by Chen et al. (2008). They could achieve considerable improvement over baseline for Chinese using only high confident edges instead of entire sentences. In our work the focus is dependency parsing of Hindi. We also explore how domain and quality of data affects the parser performance.

### 3 Hindi Dependency Parsing

In ICON 2009 and 2010, two tools contests were held that focused on Indian Language dependency parsing (Husain, 2009; Husain et al., 2010). In these contests, rule-based, constraint based, statistical and hybrid approaches were explored towards building dependency parsers for Hindi. In 2009 contest, given the gold standard chunk heads, the task was to find dependencies between them. But in 2010 contest, given words with gold features like part-of-speech (POS) and morph information, the task was to find word level dependency parse. Table 1, gives the basic statistics about the data.

Type	Sentences	Words	Average Sentence Length
Training	2,972	64632	22.69
Development	543	12617	23.28
Testing	320	6589	20.59

Table 1: Hindi ICON 2010 data statistics

#### 3.1 Baseline (State-of-the-art) System

We consider the best system (Kosaraju et al., 2010) in ICON 2010 tools contest as the starting point. Kosaraju et al. (2010) used MaltParser (Nivre et al., 2007b) and achieved 94.5% Unlabeled Attachment Score (UAS) and 88.6% Labeled Attachment Score (LAS). They could achieve this using *liblinear* learner and *nivrestandard* parsing algorithm. But, as mentioned

above, POS and other features used in this system were gold standard. The only available system which uses automatically extracted features and does complete word level parsing for Hindi is Ambati et al. (2010). Though both Ambati et al. (2010) and Kosaraju et al. (2010) used Malt-Parser, the data used is the subset of the one used by the latter and the parser settings were slightly different.

Taking training data and parser settings of Kosaraju et al. (2010) and automatic features similar to Ambati et al. (2010), we developed a parser and evaluated it on the ICON 2010 tools contest test data. We could achieve LAS of 77.9% and UAS of 86.5% on test set. This is the state-of-the-art system for Hindi dependency parsing using automatic features. We consider this system as our baseline and try to explore self-training technique.

System	UAS	LAS	LS
1) Ambati et. al. (2010); automatic features	85.5%	75.4%	78.9%
2) Kosaraju et. al. (2010); gold features	94.5%	88.6%	90.0%
3) Kosaraju et. al. (2010) + automatic features	<b>86.5%</b>	<b>77.9%</b>	<b>81.7%</b>

Table 2: Comparison of different systems

### 4 Experiments and Analysis

We have modified the Malt parser used in baseline system so that it gives a confidence value for each arc-decision taken. We have taken the average confidence value for all the arcs in the sentence to be the confidence value of a sentence. This system was first trained on ICON 2010 tools contest training data for Hindi. The model generated was then used to parse the large raw corpus. The output sentences were then sorted in descending order based on their scores.

In the self-training experiments, in each iteration, we have added 1000 sentences from the sorted output generated above, to the training data and re-trained the parser. The resulting model was then used to parse the test data.

Hindi data released in ICON-2010 tools contest is a portion of large treebank (Bhatt et al., 2009), which is under development. This is a news corpus taken from well-known Hindi news daily. Self-training experiments were performed using two types of data: one from the same news domain (in-domain) and another from a different domain (out-of-domain).

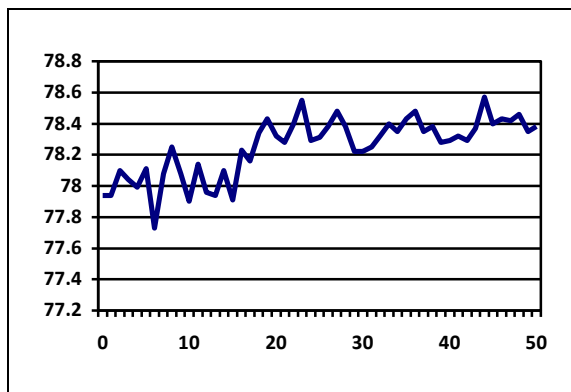


Figure 1a. Self training in-domain (LAS)

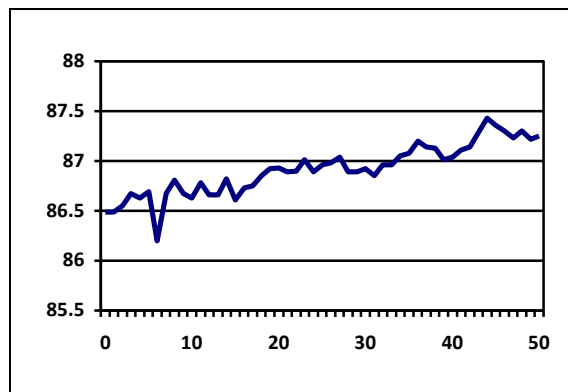


Figure 1b. Self training in-domain (UAS)

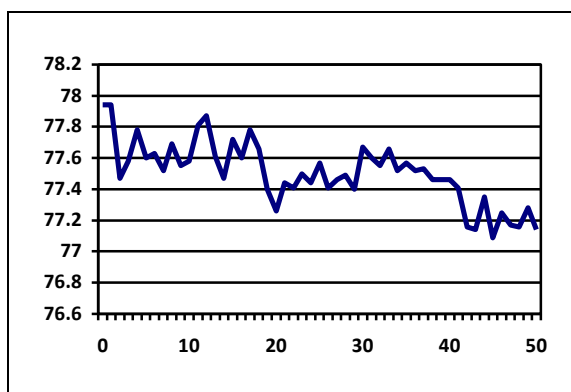


Figure 2a. Self training out-of-domain (LAS)

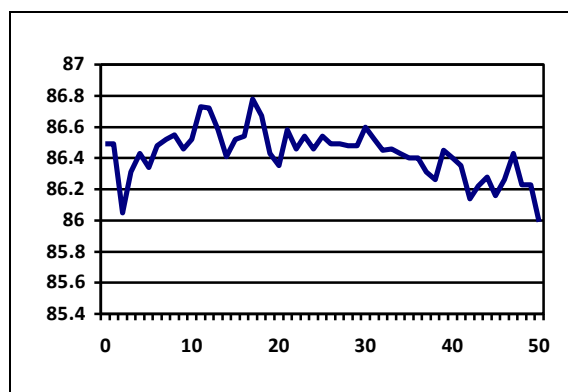


Figure 2b. Self training out-of-domain (UAS)

#### 4.1 Self training: In-domain

We have taken raw news corpus of about 108,000 sentences. As a first step, we have cleaned the data. In this process, we removed the repeated sentences, and very large sentences (>100 words per sentence). Using the above mentioned approach of self-training, we added top 1000 sentences one by one to the training data. Performance of the resulting system on test data for the first 50 iterations is shown in Figures 1a and 1b. After 50 iterations, there was steady drop in the accuracy of the system. This could be because of less confidence values of the sentences after 50th iteration. As the confidence values are low, major arcs in these sentences might be wrong. As a result, these sentences were giving negative impact on the parser performance. There were slight fluctuations in the initial iteration and peaked at 23rd iteration. At this iteration, accuracy of 78.6% LAS and 86.9% UAS was achieved. With this data, we could achieve 0.7% and 0.4% improvement in LAS and UAS respectively over the baseline.

#### 4.2 Self training: Out-of-domain

In this experiment, raw data of a domain different from the actual training, and testing data was used for self-training. For this purpose, we have taken raw non-news corpus of about 700,000 sentences. Major part of this data is from tourism domain. Similar to in-domain data, we first cleaned the data. Apart from repeated, and very large, there were a few non-Hindi sentences. We also removed them during the process of cleaning. Using the above mentioned approach of self-training (see section 4), we added top 1000 sentences one by one to the training data. Performance of the resulting system on test data for the first 50 iterations is shown in Figures 2a and 2b. After 50 iterations, there was sharp drop in the accuracy of the system. There isn't any improvement in LAS over the baseline. But in case of UAS after initial fluctuations, accuracy peaked at 17th iteration. At this iteration, accuracy of 77.8% LAS and 86.8% UAS was observed. We could achieve an improvement of 0.3% in UAS, but a decrement 0.1% in LAS.

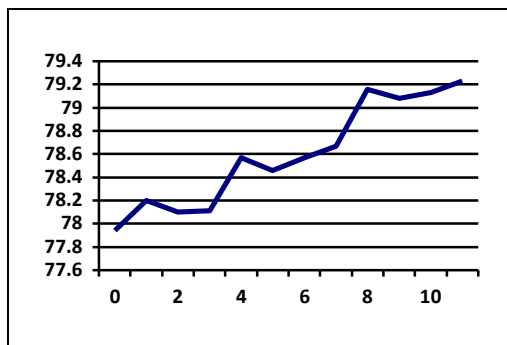


Figure 3a. Gold-standard data (LAS)

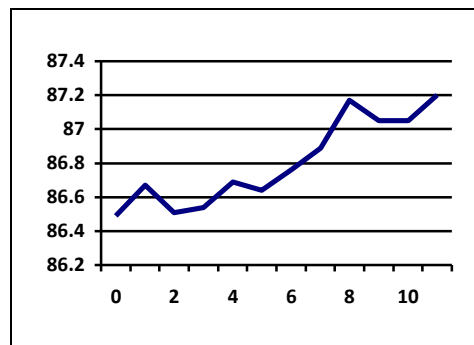


Figure 3b. Gold-standard data (UAS)

### 4.3 Gold-Standard Data

In the previous two experiments (sections 4.1 and 4.2), we have taken large amount of raw data from same and different domains and applied self-training technique. In both these experiments, impact of large automatically annotated data was observed. In this experiment, we shall observe the impact of small but, gold-standard data. We have taken the development data of ICON 2010 tools contest. We have divided the data into sets of 50 sentences and added one by one similar to above experiments. We could achieve the accuracy of 79.2% LAS and 87.2% UAS at the final iteration. With this gold standard data, we could achieve an improvement of 0.7% in UAS and 1.3% in LAS.

### 4.4 Analysis

Table 3, gives the summary comparing all the experiments performed. “\*” mark in the table shows that, accuracy is statistically significant over the baseline. Significance is calculated using McNemar’s test ( $p \leq 0.05$ ) made available with MaltEval (Nilsson and Nivre, 2008).

System	UAS	LAS	LS
1) Baseline System	86.5%	77.9%	81.7%
2) In-domain self-training	87.0%*	78.6%*	82.3%*
3) Out-of-domain self-training	86.8%	77.8%	81.6%
4) Gold-Standard Data	87.2%*	79.2%*	82.9%*

Table 2. Summary of Experiments.

We could achieve significant improvement in the accuracy when the raw data is from the same domain. But, when the data is from different domain, we haven’t seen any significant increase in the performance. This clearly shows the importance of domain of the training data. One can get better accuracies when training data is similar to testing data. As expected, adding gold-standard data outperformed both the self-training experiments. Our experiments show that gold-standard

data is the best solution for improving the parser performance. When this is not possible, raw data from same domain seems to be a better option.

Interesting observation is that even when gold data is being added, there isn’t steady increase. Slight drop was observed when some sentences are added. This clearly shows that nature of the sentences being added to training data is very important. Currently, criterion being used to extract reliable sentences from automatically parsed ones is average confidence score given by the parser. We are considering all the nodes in the sentences for calculating confidence score of sentence. It was shown by Ambati et al. (2010) that accuracy for intra-chunk dependencies is pretty high and that of inter-chunk dependencies is low. We can explore considering sentences with high confidence scores for inter-chunk nodes, rather than average score considering all the nodes. We can also explore considering only high confidence nodes rather than entire sentence, similar to works of Chen et al. (2008) and Mannem and Dara (2011).

## 5 Conclusions and Future Work

We explored the effect of self-training on Hindi dependency parsing. We showed the impact of adding small, but gold-standard data to training data versus large, but automatically parsed data on Hindi dependency parsing.

We are planning to explore the importance of co-training technique also using another parser like MSTParser, as the parser can learn new information in case of co-training. We did experiments on Hindi. There are several other languages like Telugu, Bangla etc. for which annotated data is less but large amount of raw corpus is available. We are also planning to explore the importance of self-training and co-training techniques for parsing these languages.

## References

- B. R. Ambati, M. Gupta, S. Husain and D. M. Sharma. 2010. A high recall error identification tool for Hindi treebank validation. In *Proceedings of The 7th International Conference on Language Resources and Evaluation (LREC), Valleta, Malta*.
- A. Bharati, V. Chaitanya and R. Sangal. 1995. *Natural Language Processing: A Paninian Perspective*, Prentice-Hall of India, New Delhi.
- A. Bharati, S. Husain, B. Ambati, S. Jain, D. Sharma, and R. Sangal. 2008. Two semantic features make all the difference in parsing accuracy. In *Proceedings of ICON-08*.
- R. Bhatt, B. Narasimhan, M. Palmer, O. Rambow, D. M.Sharma and F.Xia. 2009. Multi-Representational and Multi-Layered Treebank for Hindi/Urdu. In *Proceedings of the Third Linguistic Annotation Workshop at 47th ACL and 4th IJCNLP*.
- W. Chen, Y. Wu, and H. Isahara. 2008. Learning reliable information for dependency parsing adaptation. In *Proceedings of the 22nd International Conference on Computational Linguistics (COLING)*.
- S. Husain. 2009. Dependency Parsers for Indian Languages. In *Proceedings of ICON09 NLP Tools Contest: Indian Language Dependency Parsing*. Hyderabad, India.
- S. Husain, P. Mannem, B. Ambati and P. Gadde. 2010. The ICON-2010 Tools Contest on Indian Language Dependency Parsing. In *Proceedings of ICON-2010 Tools Contest on Indian Language Dependency Parsing*. Kharagpur, India.
- P. Kosaraju, S. R. Kesidi, V. B. R. Ainavolu and P. Kukkadapu. 2010. Experiments on Indian Language Dependency Parsing. In *Proceedings of the ICON10 NLP Tools Contest: Indian Language Dependency Parsing*.
- P. Mannem and A. Dara. 2011. Partial Parsing from Bitext Projections. Accepted at *the 49th Annual Meeting of the Association of Computational Linguistics*.
- D. McClosky, E. Charniak and M. Johnson. 2006. Effective Self-Training for Parsing. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*.
- 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.
- I. A. Mel'čuk. 1988. *Dependency Syntax: Theory and Practice*, State University Press of New York.
- J. Nilsson and J. Nivre. 2008. Malteval:An evaluation and visualization tool for dependency parsing. In *Proceedings of the Sixth LREC*, Marrakech, Morocco.
- 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*.
- 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.
- R. Reichart, and A. Rappoport. 2007. Self-Training for Enhancement and Domain Adaptation of Statistical Parsers Trained on Small Datasets. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*.
- S. M. Shieber. 1985. Evidence against the context-freeness of natural language. In *Linguistics and Philosophy*, p. 8, 334–343.
- M. Steedman, M. Osborne, A. Sarkar, S. Clark, R. Hwa, J. Hockenmaier, P. Ruhlen, S. Baker and J. Crim. 2003. Bootstrapping Statistical Parsers from Small Datasets. In *Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics – Volume 1*.