# Two-stage Approach for Hindi Dependency Parsing Using MaltParser

*Karan Singla, Aniruddha Tammewar, Naman Jain, Sambhav Jain*

LTRC IIIT Hyderabad

{karan.singla, uttam.tammewar, naman.jain}@students.iiit.ac.in,
sambhav.jain@research.iiit.ac.in

## Abstract

In this paper, we present our approach towards dependency parsing of Hindi language as a part of Hindi Shared Task on Parsing, **COLING 2012**. Our approach includes the effect of using different settings available in Malt Parser following the two-step parsing strategy i.e. splitting the data into interChunks and intraChunks to obtain the best possible **LAS**[1], **UAS**[2] and **LA**[3] accuracy. Our system achieved best LAS of **90.99%** for **Gold Standard** track and second best LAS of **83.91%** for **Automated** data.

Keywords : Hindi dependency parsing, two-stage parsing, MaltParser, interChunk, intraChunk

1 Labeled Attachment Score
2 Unlabeled Attachment Score
3 Labeled Score

## 1 Introduction

Hindi is a morphologically rich and relatively free-word order language(MoR-FWO). Parsing is a challenging task for such MoR-FWO languages like Turkish, Basque, Czech, Arabic, etc. because of their non-configurability. It has been suggested that these kind of languages can be represented better using dependency framework rather than constituent framework (Hudson, 1984; Shieber, 1985; Mel'čuk, 1988, Bharati et al., 1995).

Previous efforts on parsing MoR-FWO languages includes Nivre et al., 2007b; Hall et al. 2007; McDonald and Nivre, 2007 etc. In ICON 2010, best results were obtained by (Kosaraju et al., 2010), which uses Malt parser with SVM classifier for labeling and using local morph-syntactic, chunk and automatic semantic information as features. Ambati et al. (2010) has explored two-stage approach of parsing Hindi. It divided the data into two parts namely, interChunks and intraChunks. The inter chunk part of the data contains only dependency relations between chunk heads of the sentences while the intra chunk data has the dependency relations between the tokens of a chunk. The dependency relation labels for interChunk and intraChunk are disjoint. This approach helps in avoiding intraChunk relations to be marked as interChunk relations and vice-versa. Following this approach, we explored different parsing algorithm parameters and learner algorithm settings of Malt Parser.

The rest of the paper is divided into four sections. In section 2, we briefly discuss about the training and testing data, accompanied by a few statistics from the treebank, which guides the parameter selection for our experiments. Section 3 contains the details of our experiments along with the results. In section 4 we present error analysis. Finally, we conclude the paper in section 5 with a summary and the future work.

## 2 Data

A subset of the dependency annotated Hindi Treebank (HTB ver-0.5) is released as part of the Hindi Parsing Shared Task-2012 (HPST-2012). Morphological analysis, however, has not been validated for errors and inconsistencies. It was released for two evaluation tracks (gold standard and automatic). In the gold standard track, the input to the system consists of lexical items with gold standard morphological analysis, part-of-speech tags, chunks and the additional features listed above. In the automatic track, the input to the system contains only the lexical items and the part-of-speech tags from an automatic tagger. Some sentences have been discarded due to presence of errors in the data. Table 1 shows the training, development and testing data sizes for Hindi. For the testing phase of the contest, the parser was trained on the entire released data(training + development).

| Type | Sent Count | Token Count | Avg. Sentence Length |
|---|---|---|---|
| **Training** | 12041 | 268,093 | 22.27 |
| **Development** | 1233 | 26,416 | 21.42 |
| **Testing** | 1828 | 39,975 | 21.87 |

**Table 1.** Treebank Statistics

## 3 Experiments and results

In our experiments we have used freely available Malt Parser (version 1.6.1) (Nivre et al., 2007). In this section we give an account of experiments performed in a series. Each experiment focuses on choosing the best option for a certain parameter/feature keeping the other parameter/feature fixed. In the subsequently following experiments the best parameter chosen from previous experiment is retained.

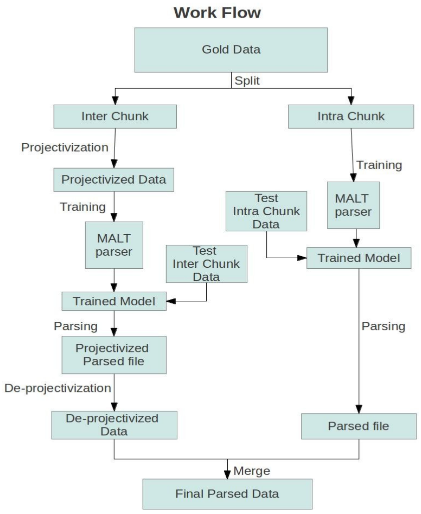

Figure 1.

### 3.1 Feature model

Feature model is the template, which governs the learning from the given training data. We explored various configuration for feature model using insight from previously used feature models from similar tasks. We observed feature model used by (Kosaraju et al., 2010) performs best.

### 3.2 Two-stage(inter-intra chunk) approach:

Every sentence in data is divided in chunks.

e.g. (राम ने) (सीता को) (एक लाल किताब) (दी) ।

(raam *erg.)* (sita *dat.)* (one red book) (gave).

Ram gave a red book to Sita.

In the above example, the sentence is divided in 4 chunks marked by brackets. The dependency labels for intraChunk relations are different from that of interChunk, forming two disjoint sets of dependency labels : interChunk labels(k1, k2, k7, etc.) and intraChunk labels(nmod_adj, lwg_psp, mod, etc.).

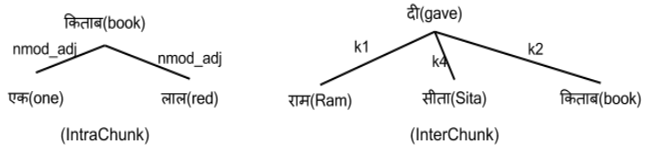

Figure 2.

Theoretically, the parent of a non-chunkhead token should be a chunkhead or non-chunkhead token of the same chunk and the relation should be labeled with an intraChunk label. (in the example, the non-chunkHead tokens एक and लाल are connected with the chunkHead token किताब also the relation label *nmod_adj* is an intraChunk label). The parent of the chunkhead must also be a chunkhead from another chunk. The relation should be marked with interChunk label.(in the given example, the chunkHead tokens दी ,राम,सीता and किताब are attached with a chunkHead token also the relation labels are interChunk labels). However, in the training data, there are few noisy cases where intraChunk relations are marked as interChunk and vice-versa. The cases were very less in number and hence were ignored.

Dividing the data into inter and intra chunk, all the above constraints will be automatically handled. In the resulting intraChunk data, the chunks formed will behave as an individual sentence therefore the parser would not be able to make an inappropriate arc.

## 3.3 Experiment with projectivity

MaltPaser has a default constraint to give only projectiive output. However, in the training data we find approximately 1.1% arcs to be non-projective. To address the non-projectivity in data, we use pseudo-projective algorithm as proposed by Nivre et al, 2005. We only incorporated the pseudo-projective algorithm in case of interChunk data as in intraChunk we found the arcs to be always projective. There are three options available with the pseudo-projective algorithm in MaltParser. We performed intermediary[6] experiments on all of these and got some interesting results.

Pseudo-projective algorithm replaces all the non-projective arcs in the input data to projective arcs by applying a lifting operation. The lifts are encoded in the dependency labels of the lifted arcs. In order to apply an inverse transformation to recover the underlying (non projective) dependency graph, there is a need to encode information about lifting operations in arc-labels. The encoding scheme can be varied according to **marking_strategy** and there are currently five of them: **none**, **baseline**, **head**, **path** and **head+path**(Nivre et al. 2005). We performed intermediary experiments separately on each of them and observed that **head** option gives the best result. This option projectivizes input data with head encoding for labels.

Secondly, there is an option called **covered_root**, which is mainly used for handling dangling punctuation. This option has five values: **none**, **ignore**, **left**, **right** and **head**. In our intermediary experiments, we found that **ignore** gave better results than others.

On the basis of lifting order, there are two ways to lift the non-projective arcs namely, **shortest** and **deepest**. In the **deepest** lifting order, most deeply nested non-projective arc is lifted first, not the shortest one. In our experiments we found that deepest has no effect in increasing the parsing accuracy rather there is a slight decrease in the accuracy as compared to shortest**.**

## 3.4 Experiment with features

We tried to experiment with the types of features that can be used in FEATS column in the CoNLL-X format. We considered four ways: 1)without any information in FEATS column 2) only tam[4] and vib[5] information, 3)tam and vib along with chunkType and 4)with all the information present by default. The best results were obtained using all information. All this could only be done for the Gold track as such information about the features is not provided for Automatic track. This is the major reason for the difference in the parsing accuracies between gold and auto data.

## 3.5 Experiment with algorithms

Kolachina et al., 2010 has shown that **nivre_eager** algorithm gives the best accuracy for Hindi. Our intermediary experiments also support the same. We also explored the **root-handling** option which can be **normal**, **strict** and **relaxed**. Our experiments showed that relaxed option gives the best accuracy. In relaxed option**,** root dependents are not attached during parsing (attached with default label afterwards) and reduction of unattached tokens is permissible.

## 3.6 Experiment with prediction strategy

There are three types of **prediction strategies** available in MaltParser :

1. **combined**(default): Combines the prediction of the transition and the arc label .
2. **sequential**: Predicts the transition and continues to predict the arc label if the transition requires an arc label.
3. **branching**: Predicts the transition and if the transition does not require any arc label then the non determinism is resolved, but if the predicted transition requires an arc label then the parser continues to predict the arc label.

We performed experiments with the above options and found that using **branching** there is an increase in the parsing accuracy.

4 tense aspect modality
5 Vibhakti (post-postion)

## 3.7 Experiment with SVM settings

In our experiments, we used the LIBSVM learner algorithm following the SVM settings(s0t1d2g0.12r0.3n0.5m100c0.7e0.5) in experiments reported by Kolachina et al.(Kolachina et al., 2010) for Hindi. These settings gave a better result over the default SVM settings.

## 3.8 Results

We have trained MaltParser separately using Gold and Auto training data. For gold data, we trained two models, one for interChunk data with all settings obtained in the above experiments and other for intraChunk data with all the above settings except branching and projectivization. For both we used the same algorithm **"nivre_eager"** and learner **"LIBSVM"**. The final evaluation, the system demonstrated LAS is **90.99%,** UAS is **95.87%** and LA is **92.58%** respectively. For Automatic data, we didn't split the data in two parts as the information on which the data is divided is missing in the testing files. Except this all the other settings are exactly similar as for the gold data. The final LAS is **83.91%**, UAS is **91.70%** and LA is **86.77%** respectively.

## 4 Error analysis

| correct label | system output label | frequency |
|---|---|---|
| pof | k2 | 139 |
| k1 | k2 | 123 |
| k2 | pof | 112 |
| k7 | k7p | 95 |
| k7p | k7 | 88 |

Table 2. Top 5 most frequent errors

The most frequent errors that the parser made contained the confusion between marking of k2, k1 and pof dependencies. The confusion between k1 and k2 is because of the absence of the case markers for disambiguation. As pof is the verbal form of noun, it is even difficult for humans to disambiguate between pof and k2. The confusion between k7 and k7p is also frequent because of their closeness. Some of these errors can be handled by post-processing.

## 5 Conclusions and future work

In this paper we experimented with different parameters of data-driven Malt Parser along with the two-stage preprocessing approach to build a high quality dependency parser for Hindi. In future, we would like to explore other data-driven parsers like MST. Further experiments on combining parsers by stacking can also be performed.

## References

S. Husain, P. Mannem, B. Ambati and P. Gadde.2010. The ICON-2010 tools contest on Indian language dependency parsing. In Proc of *ICON-2010 tools contest on Indian language dependency parsing*. Hyderabad, India.

Bharat Ambati,Samar Husain,Sambhav Jain,Dipti Misra Sharma,Rajeev Sangal, 2010.Two methods to incorporate local morphosyntactic features in Hindi dependency parsing, *NAACL 2010: Human Language Technologies: The 11th Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL2010 2010)*

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

Sudheer Kolachina,Prasanth Kolachina,Manish Agarwal,Samar Husain, 2010.Experiments with Malt Parser for parsing Indian Languages, *NLP Tools Contest in ICON-2010: 8th International Conference on Natural Language Processing (NLP Tools Contest: ICON-2010 2010)*

A. Bharati, R. Sangal and D. M. Sharma. 2006a. SSF:Shakti Standard Format Guide. *LTRC R33.*http://ltrc.iiit.ac.in/MachineTrans/publications/technicalReports/tr033/ SSF.pdf

A. Bharati, V. Chaitanya and R. Sangal. 1995. *NaturalLanguage Processing: A Paninian Perspective*, Pren-tice-Hall of India, New Delhi, pp.65- 06.ltrc.iiit.ac.in/downloads/nlpbo -ok/ nlp-panini.pdf S. M. Shieber. 1985. Evidence against the context-freeness of natural language. In Linguistics and Philosophy, p. 8, 334–343.

R. McDonald and J. Nivre. 2007. Characterizing the Errors of Data-Driven Dependency Parsing Models. In Proc of *Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*

J. Hall, J. Nilsson, J. Nivre, G. Eryigit, B. Megyesi,M. Nilsson and M. Saers. 2007. Single Malt or Blended? A Study in Multilingual Parser Optimization. In Proceedings of the *CoNLL Shared Task Session of EMNLP-CoNLL 2007,* 933—939

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 Proc 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.

Nivre, J. and J. Nilsson (2005) Pseudo-Projective Dependency Parsing. In Proceedings of the *43rd Annual Meeting of the Association for Computational Linguistics*, pp. 99-106

I. A. Mel'čuk. 1988. Dependency Syntax: Theory and Practice, State University, Press of

New York.

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