ISI-Kolkata at MTPIL-2012

Arjun Das, Arabinda Shee and Utpal Garain
INDIAN STATISTICAL INSTITUTE, 203, B. T. Road, Kolkata 700108, India.
{arjundas|arabinda|utpal}@isical.ac.in

ABSTRACT

In this paper we present our work in the MTPIL-2012 dependency parsing task on Hindi using MaltParser. Here we have experimented with MaltParser by selecting different parsing algorithms and different features selection. Finally, we have achieved unlabeled attachment score (UAS) of 91.80%, labeled attachment score (LAS) 86.51% and labeled accuracy (LA) 88.47% respectively.

KEYWORDS: Dependency parser, MaltParser, Hindi.

Proceedings of the Workshop on Machine Translation and Parsing in Indian Languages (MTPIL-2012), pages 185–190,
COLING 2012, Mumbai, December 2012.

1 Introduction

Dependency parsing is one of the core applications of Natural Language Processing. Dependency parsing is useful in other NLP application like Question Answering, Machine Translation, word Sense Disambiguation etc. Dependency parsing can be divided into grammar-driven dependency parsing and data-driven dependency parsing. In the grammar-driven dependency parsing the grammars or set of rules are extracted from a corpus by linguist. Where as in the data-driven approach a large manually annotated training data is required.

In recent past ICON has organized a shared task competition in dependency parsing in Indian languages, namely Hindi, Bengali and Telugu [ICON 2009, 2010]. The ICON task consisted in training and evaluating of dependency parsers. Each shared task 2009 and 2010 had much lesser data to work with (20,000 words). Similarly, the MTPIL-2012 dependency parsing task also consisted in training and evaluating dependency parsers for Hindi. We have participated in this task using the freely available MaltParser [Nivre et al., 2006a] which follows the data-driven approach. In this experiment we have trained and evaluate Maltparser with default properties. And then step-by-step we tried to optimize those features for which the parsing accuracy increases.

2 MaltParser for Hindi

In this experiment we have customized MaltParser [Nivre et al., 2006a]. During MaltParser optimization we follow same approach described by Nivre, (2009). MaltParser comes with several parsing algorithms. We experimented with different parsing algorithms. The result shows that arc-standard projective system gave the highest accuracy for Hindi. Moving further we try to optimize those features for which parser accuracy increases. For this we first added all possible features. Then we discarded those features for which the parsing accuracy increases. Finally, we end up with following features:

- Features 2 and 9, the top and next for lemma.
- Features 3 and 10, the coarse-grained part of speech of top and next.
- Feature 5 and 12, the top and next of morphological features.
- Features 21, 25, 28 and 31, the part of speech features are added.
- Features 27 and 30, the form of leftmost dependencies of next and predecessor of top.
- The conjoined features (1&4, 1&8, 4&11) i.e. part of speech and form of stack top, form of top and next, part of speech of top and next was also added.

We used LIBSVM package [Chang and Lin, 2001] for classification task.

3 Training Data

A set of training data and development data has been provided to all the participants. The training set contains 12041 sentences (268,093 words) and the development set contains 1233 sentences (26416 words). We combined both data to one training set.

4 Evaluation

There are two evaluation tracks (gold standard and automatic) in the shared task and all the participating systems must participate in both the tracks. In the gold standard track, the input to the system consists of sentence tokens 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 sentence token and the part-of-speech tags from an automatic tagger. In both the tracks, the parser must output the head and the corresponding dependency relation for each token in the input sentence.

Table 1 Performance on MTPIL-2012 Data

	Baseline			Optimized(Final)		
	LAS	**UAS**	**LA**	**LAS**	**UAS**	**LA**
Hindi-Gold	80.84	89.32	83.17	86.51	91.80	88.47
Hindi-Auto	-	-	-	32.34	38.25	32.93

Table 1 shows the results for the final optimized model and the baseline model using MTPIL test data. We have found the largest improvement in LAS, with 5.67 percent, while the improvement in UAS and LA is 2.48 percent and 5.3 percent respectively for the gold track. We want to see the parser performance with minimal numbers of features in the training data. So we left the auto-track training data as it was. As it was expected the parsers performs poorly with UAS of 38.25 percentages.

5 Error Analysis

A primary goal of this experiment is to point out the errors made by MaltParser. We have performed a number of experiments to find out most possible errors with respect to sentence length factor. We have also shown dependency relation wise performance for the gold track.

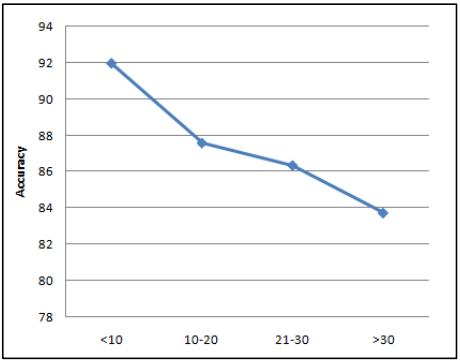

Figure 1 Parser Accuracy (LAS) and Sentence Length

5.1 Length Factor

In this experiment we have performed several experiments on the gold track data to find out the parser performance with different sentence length i.e., number of tokens. It is a well known fact that dependency parsers tends to perform well on shorter sentences than longer ones. Figure 1 shows the accuracy i.e. the labeled attachment score (LAS) for the parser with respect to different sentence length. From the figure it is clear Malt Parser tends to perform better on shorter sentences.

5.2 Dependency Relation Wise Evaluation

Table 2 presents more detailed analysis of results by reporting lower recall and precision for 15 dependencies in the gold track evaluation. The lowest accuracy is reported for the label "rs" with recall 13.97 percent.

Table 2 Dependency Relation-wise Performance Evaluation

Deprel	Gold	Correct	System	Recall (%)	Precision (%)
k1s	328	210	274	64.02	76.64
k2	1957	1461	1947	74.66	75.04
k3	56	21	41	37.5	51.22
k4	283	192	269	67.84	71.38
k4a	67	28	49	41.79	57.14
k5	156	108	214	69.23	50.47
k7a	84	67	87	79.76	77.01
k7p	566	397	541	70.14	73.38
nmod__k1inv	161	119	169	73.91	70.41
r6-k1	68	19	42	27.94	45.24
r6-k2	306	224	283	73.2	79.15
ras-k1	74	36	61	48.65	59.02
rh	152	107	141	70.39	75.89
rs	179	25	41	13.97	60.98
vmod	493	345	472	69.98	73.09

6 Conclusions

This paper presents optimization and evaluation of MaltParser for Hindi. Due to large amount of training data the parser was able to achieve such high accuracy with default properties. It is interesting to see using different feature selection how parser performance can be improved further. The evaluation results reported here will be useful for future research in this area.

Acknowledgments

We would like to thank the organizers of MTPIL for their effort from starting to the end.

References

Chih-Chung Chang and Chih-Jen Lin, (2001). LIBSVM: A Library for Support Vector Machines. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

ICON (2009). *NLP Tool Contest: Parsing*, In *7th International Conference on Natural Language Processing*, Hyderabad, India.

ICON (2010). *NLP Tool Contest: Parsing*, In *8th International Conference on Natural Language Processing*, Khragpur, India

Nivre, J., J. Hall, and J. Nilssion. (2006a). MaltParser: A data-driven parser-generator for dependency parsing. *In Proceedings of LREC*, 2216-2219.

Nivre, J. (2009). Parsing Indian Languages with MaltParser. In *Proceedings of the ICON09 NLP Tools Contest: Indian Language Dependency Parsing*, 12-18.