Linguistically Rich Graph Based Data Driven Parsing For Hindi

Samar Husain, Pujitha Gade and Rajeev Sangal
Language Technologies Research Centre, IIIT-Hyderabad, India.
{samar, pujitha.gade}@research.iiit.ac.in, sangal@mail.iiit.ac.in

Abstract

In this paper we show how linguistic knowledge can be incorporated during graph based parsing. We use MSTParser and show that the use of a constraint graph, instead of a complete graph, to extract a spanning tree improves parsing accuracy. A constraint graph is formed by using linguistic knowledge of a constraint based parsing system. Through a series of experiments we formulate the optimal constraint graph that gives us the best accuracy. These experiments show that some of the previous MSTParser errors can be corrected consistently. It also shows the limitations of the proposed approach.

1 Introduction

In MSTParser (McDonald et al., 2005a, 2005b; a graph based data driven parser), a complete graph is used to extract a spanning tree during derivation. MSTParser's learning model uses large-margin algorithm, which optimizes the parameters of the model to maximize the score margin between the correct dependency graph and all incorrect dependency graphs for every sentence in a training set. The learning procedure is global. Unlike, MaltParser (and other transition based systems, see Kubler et al., 2009), MSTParser considers limited history of parser decisions during training. McDonald and Nivre (2007) characterize in detail the specific error patterns in MSTParser. Recent works such as Sagae and Lavie (2006), Nivre and McDonald (2008), Zhang and Clark (2008), Koo and Collins (2010), have tried to improve the parsing accuracy either by integrating the two parsers via stacking, etc. or by introducing better learning models.

In this work we try to investigate if parsing accuracy using MSTParser can be improved by providing it a constraint graph instead of a complete graph during the derivation step. Our work is related to Bergsma and Cherry, (2010), where they try something similar to increase parser speed. We modify the Chu-Liu-Edmonds algorithm such that it would start with the modified graph instead of a complete graph. This algorithm was chosen over the Eisner's algorithm as the Hindi treebank contains ~14% non-projective arcs (Mannem et al., 2009). While we do not change the learning phase, it will be interesting to see what effect certain linguistic knowledge alone can have on the overall accuracy. A constraint graph is formed by using linguistic knowledge of a constraint based parsing system (Bharati et al., 2009). Through a series of experiments we formulate the optimal constraint graph that gives us the best accuracy. These experiments show that some of the previous MSTParser errors can be corrected consistently. It also shows the limitations of the proposed approach.

The paper is arranged as follows, in section 2 we briefly discuss the notion of a constraint graph for a sentence; Section 3 describes the experimental setup. The experiments are discussed in section 4, followed by the results and observations in section 5. We finally conclude the paper along with future directions in section 6.

2 Constraint Graph

Bharati et al., (2009) proposed a two-stage constraint based hybrid dependency parsing approach for free word order languages. They divide the task of parsing into intra-clausal and inter-clausal stages. At each stage valency frames for various heads (mainly for verbs and conjunctions) are used to construct a constraint graph. The parser currently uses close to 536 manually annotated valency frames. The constraint graph is then converted into an integer programming problem to get the parse at each stage. Let us look at a sample constraint graph for a Hindi sentence.

Proceedings of the 2nd Workshop on Statistical Parsing of Morphologically-Rich Languages (SPMRL 2011), pages 56–61,
Dublin, Ireland, October 6, 2011.

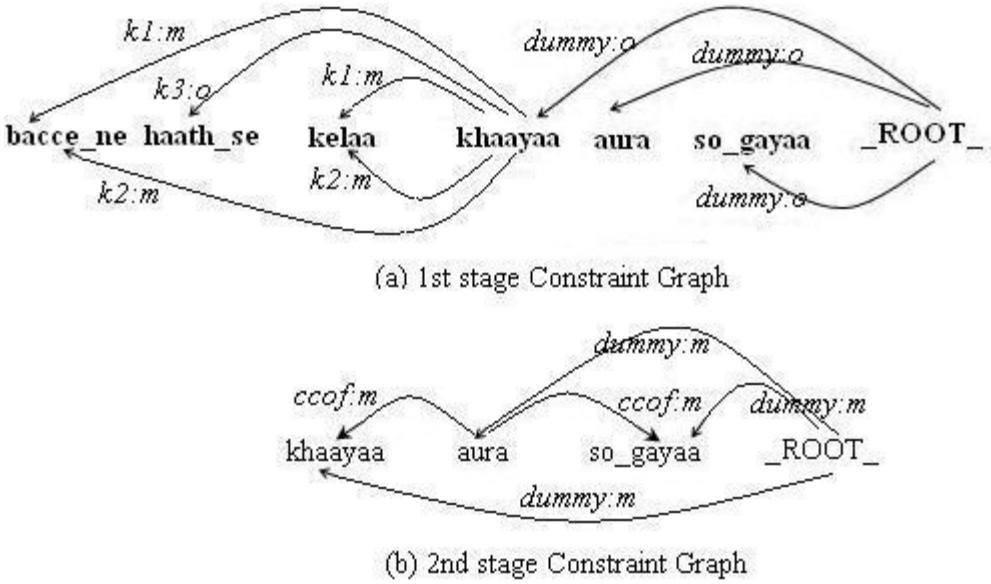

Figure 1. Constraint graph for sentence 1.
(Although a constraint graph has arc labels, they are not used in our experiments.)

(1) *bacce ne haath_se kelaa khaayaa*
'child' ERG hand_with 'banana' 'eat'
aura so gayaa
'and' 'sleep' PAST
'The child ate the banana with his hand and slept'

Figure 1 shows the 1st stage and the 2nd stage Constraint graph (CG) for Example 1. Note that the arcs in 1st stage CG are localized to individual clauses. The _ROOT_ node is required in order to get the partial parse at the end of the 1st stage. Also note that in the 2nd stage only the inter-clausal relations are considered (here finite verbs and a conjunctions). In such a scenario 1st stage and 2nd stage CGs are distinct and vary drastically in size. This can be clearly seen in Figure 1.

The CG for each sentence provides the linguistic knowledge that we will use in various experiments in this paper. We can use this information in two ways:

a) Complete CG or stage specific CG can be directly used instead of a complete graph during the derivation.
b) Specific information from CG can be used to prune out certain arcs in the complete graph while retaining others.

For the experiments discussed in this paper we use the latter. We note that although CG also provides arc labels, for all our experiments we are only concerned with the attachment information. This is because the spanning tree extraction algorithm in MSTParser uses unlabeled graph. MSTParser uses a separate classifier to label the trees.

3 Experimental Setup

All the experiments are conducted on Hindi. We use the dependency treebank released as part of the ICON2010 tools contest (Husain et al., 2010). The training data had 2,973 sentences. Development and testing had 543 and 321 sentences respectively. MSTParser[1] was modified so that it can use CG during derivation. We use the non-projective algorithm, order=1 and training k=5. We use the feature set optimized for Hindi by Bharati et al. (2008). Experiments were first conducted using training and development data. Once the experimental design was frozen, only then the test data was used.

4 Experiments

For an input S = w_0, w_1, ….w_n, i.e. the set of all words in a sentence, let G_S be the complete graph, and CG_S be the constraint graph provided by the constraint parser. Let N = {w_0, w_1, ….w_n} be the

[1] MST Version 0.4b

set of vertices in G_S. $A_G = N \times N$ and $A_{CG} \subseteq N \times N$ is the set of arcs in the two graphs. An arc between w_i and w_j, shown as (w_i,w_j) signifies w_i as the parent of w_j. X is the set of all the nodes which occurs as a child in A_{CG}. Also, let C be the set of all vertices which are conjunctions, V be the set of all vertices which are verbs, K be the set of all vertices which are nouns, P be the set of all vertices that have a case-marker/post-position and J be the set of adjectives.

The set of arcs which will be pruned from the complete graph in experiment 1 is shown in Table 1. This means that all the arcs in G will be pruned except the ones present in CG.

```
For y in X:
    For x in S:
        If ∄ (x,y) in A_CG:
            Remove (x,y) from A_G
```

Table1. Experiment 1 valid arcs

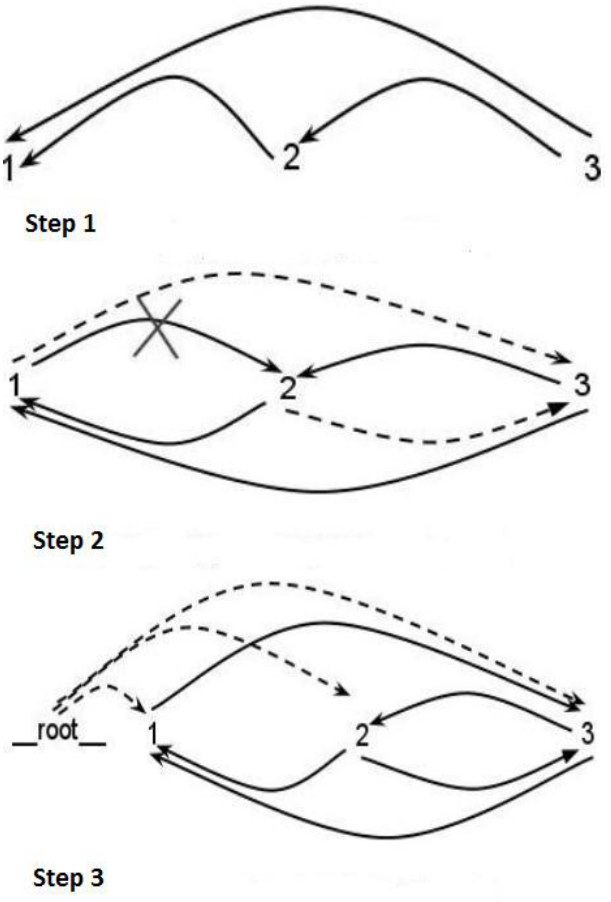

Figure 2. Illustration of Experiment 1

Experiment 1 is illustrated in Figure 2. Step 1 is a sample constraint graph for a sentence with three words which are represented as 1, 2 and 3. In step 2 we prune arcs according to the Constraint graph. For nodes which have no parents in CG we keep all their incoming arcs intact as shown in Figure 2 with dashed arcs. In step 3 we add arcs from __ROOT__ to all the nodes. The final graph is used by MSTParser to extract the parse tree during derivation.

The parser in experiment 1 (E1) outperformed the baseline UAS (more details in section 5). Further analysis showed that the pruning based on E1, although useful, also had some negative effect, i.e. it also prunes out many potentially valid arcs that would have been originally considered by MSTParser. Through experiments 2-8 we explore if we can minimize such invalid pruning. We do this by systematically considering parts of the CG and using only those parts for pruning G.

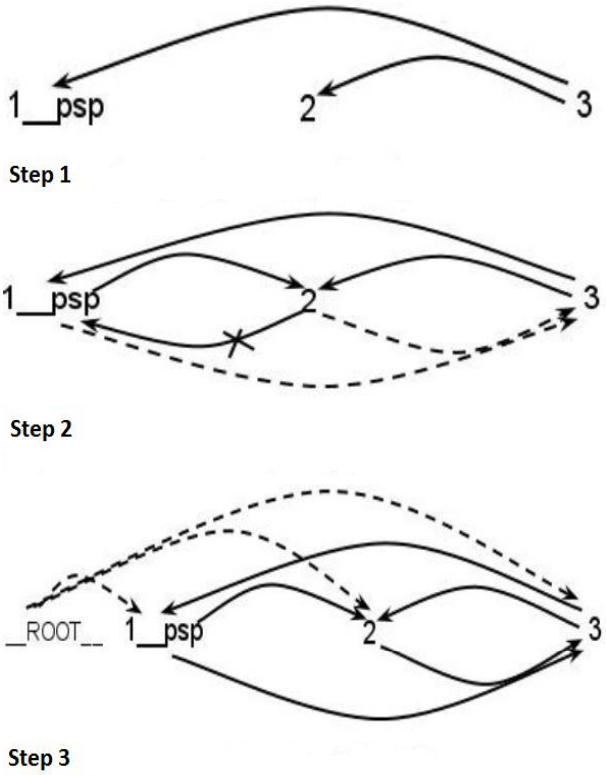

Figure 3. Illustration of Experiment 2

Experiment 2 (Table 2, 1st row) begins with focusing on child nodes with post-positions. Also incorporated are the conjunction heads. Since a CG is formed based on explicit linguistic cues, it makes

sense to base our decision where concrete information is available. Experiment 2 is illustrated in Figure 3. Experiment 3 (Table 2, 2nd row) uses similar conditions, except the constraint of nodes with post-position is only on noun children. By doing this we are trying to explore the most appropriate information in the CG.

```
For y in X:
    For x in S:
        If ∄ (x,y) in A_CG:
            Remove (x,y) from A_G
        If ∃ (x,y) in A_CG and y∉P and x∉C:
            Remove(x,y) from A_G
```

```
For y in X:
    For x in S:
        If ∄ (x,y) in A_CG:
            Remove (x,y) from A_G
        If ∃ (x,y) in A_CG and y∈K and y∉P and x∉C:
            Remove(x,y) from A_G
```

Table 2. Experiment 2 and 3 valid arcs

```
For y in X:
    For x in S:
        If ∄ (x,y) in A_CG:
            Remove (x,y) from A_G
        If ∃ (x,y) in A_CG and y∈K and y∉P and x∈V:
            Remove(x,y) from A_G
```

Table 3. Experiment 4 valid arcs

It is interesting to note that in experiment 4 we are trying to prune out invalid arcs related to the argument structure information of a verb (x∈V) available in a CG. Using CG only for verbal arguments with case-marker captures various verbal alternations manifested via case-markings.

Experiment 5 and 6 extends experiments 4 and 3 respectively by introducing an exception where a noun child *y* with no case-marker is considered only if there exists other potential conjunction/adjectival head for *y*. Owing to the free-word order property of Hindi, identifying the head of a noun with no case-marker is a rather difficult task. In spite of their availability many robust generalizations (that help disambiguate relations with nouns with no case-markings) such as agreement remain unexploited during training (Ambati et al., 2010). In this experiment therefore, we are trying to ensure that the ambiguity of correct heads for nouns with no post-position is not resolved by CG.

```
For y in X:
    For x in S:
        If ∄ (x,y) in A_CG:
            Remove (x,y) from A_G
        If ∃ (x,y) in A_CG and y∈K and y∉P and x∈V:
            If ∄ (z,y) in A_CG and (z∈C or z∈J)
                Remove(x,y) from A_G
```

```
For y in X:
    For x in S:
        If ∄ (x,y) in A_CG:
            Remove (x,y) from A_G
        If ∃(x,y) in A_CG and y∈K and y∉P and x∉C:
            If ∄ (z,y) in A_CG and (z∈C or z∈J)
                Remove(x,y) from A_G
```

Table 4. Experiment 5 and 6 valid arcs

Experiments 2-6 only catered to verbal, conjunction or adjectival head. Experiment 7 and 8 extend 5 and 6 to handle nominal predicate heads. We note here that this information is not obtained from the CG and is being treated as a heuristic rather than having some linguistic validity. The constraint parser has very limited coverage for nominal predicates and therefore we cannot rely on it for this kind of information. The heuristic considers a possibility of an attachment between two consecutive nouns and does not remove such arcs from the G.

5 Results and Discussion

Figure 4 shows the results for all the experiment. The baseline UAS was 88.66 and the best result was obtained from experiment 8 with the UAS of 89.31. This is an increase of 0.65%. There was also an increase of 0.45% in the LAS. All the improvements in the results were statistically significant using McNemar's test ($p<0.01$ for improvement in UAS).

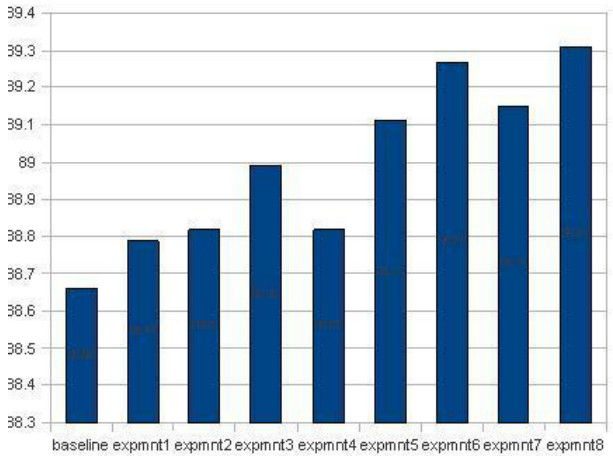

Figure 4. UAS of all the experiments.

Table 5 shows that the improvement in the accuracies is spread across different kinds of relations.

Relations	Baseline		Experiment 8	
	UAS	LAS	UAS	LAS
Intra-clausal	87.86	73.63	88.30	74.00
Verbal Args* (Intra-clausal)	89.46	72.28	90.02	72.68
Non-args (Intra-clausal)	86.25	75.00	86.57	75.32
Inter-clausal	91.85	91.53	93.29	92.33

Table 5. Comparison of baseline and Experiment 8 accuracies for various relations.
(*Args: arguments)

Both inter-clausal as well as intra-clausal relations benefit. Within a clause, both argument structure of a verb and other relations are better identified in Experiment 8 when compared to the baseline. There was a consistent improvement in the analysis of certain phenomenon. These were:

a) Intra-clausal coordinating conjunction as dependents. These may appear either as arguments of the verb or as children of non-verbal heads.
b) Better handling of arguments of non-finite verbs and gerunds
c) Better handling of clausal arguments and relative clauses.

A similar pattern is seen from Table 6 where there is an increase for almost all the POS tags in the head attachment accuracy, except for adjectival attachments.

As mentioned earlier, the constraint graph is originally formed using the linguistic knowledge of the constraint based system. It is clear that for our experiments the coverage of this knowledge is very crucial. Our experiments show that while the coverage of verbal and conjunction heads is good, knowledge of other heads such as predicative nouns and adjectives is lacking. As mentioned earlier the constraint parser currently uses close to 536 valence frames. It would be interesting to see how grammar extraction methods for Hindi (Kolachina et al., 2010) can be combined with our approach to boost the knowledge base being currently used.

POS tag	%Instance	Accuracy	
		Baseline	E8
Noun	64%	89%	90%
Finite verb	15%	94%	95%
Non-finite verb	3%	83%	83%
Gerund	5%	86%	88%
Conjunction	7%	75%	77%
Adjective	4%	98%	95%

Table 6. Head attachment accuracy distribution over POS

6 Conclusion and Future directions

In this paper we successfully integrated linguistically driven constraint graph in MSTParser. Through a series of experiments we showed that selective use of such information leads to improvement in parser accuracy. Through analyzing the results we fleshed out the areas of improvement. We also pointed out where our approach harms the parsing accuracy.

The linguistic knowledge in all our experiments are currently being used as hard constraints, but they can also be used as soft constraints similar to parser stacking approaches explored by Nivre and McDonald, (2008) and Husain et al., (2011). Use of grammar extraction techniques in the future to automatically construct the constraint graph seems to present a very attractive strategy which can complement these experiments.

Acknowledgement

We would like to thank the three anonymous reviewers for their extensive and insightful comments that helped us improve the paper.

References

B. R. Ambati, S. Husain, J. Nivre and R. Sangal. 2010. On the Role of Morphosyntactic Features in Hindi Dependency Parsing. *In Proceedings of NAACL-HLT 2010 workshop on Statistical Parsing of Morphologically Rich Languages (SPMRL 2010), Los Angeles, CA.*

A. Bharati, S. Husain, D. M. Sharma and R. Sangal. 2009. Two stage constraint based hybrid approach to free word order language dependency parsing. *In Proceedings of the 11th International Conference on Parsing Technologies (IWPT). Paris.*

A. Bharati, S. Husain, B. Ambati, S. Jain, D. M. Sharma and R. Sangal. 2008. Two semantic features make all the difference in Parsing accuracy. *In Proceedings of the 6th International Conference on Natural Language Processing (ICON-08), CDAC Pune, India.*

S. Husain, P. Gadde, J. Nivre and R. Sangal. 2011. Clausal parsing helps data-driven dependency parsing: Experiments with Hindi. *In Proceedings of IJCNLP 2011.*

S. Husain, P. Mannem, B. R. 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. Kolachina, S. Kolachina, A. K. Singh, V. Naidu, S. Husain, R. Sangal and A. Bharati. 2010. Grammar Extraction from Treebanks for Hindi and Telugu. *In Proceedings of The 7th International Conference on Language Resources and Evaluation (LREC). Valleta. Malta. 2010.*

T. Koo and M. Collins. 2010. Efficient Third-order Dependency Parsers. In *Proc of ACL2010.*

S. Kubler, R. McDonald and J. Nivre. 2009. *Dependency parsing.* Morgan and Claypool.

P. Mannem, H. Chaudhry and A.Bharati. 2009. Insights into Non-projectivity in Hindi. *Proceedings of ACL-IJCNLP Student Research Workshop. Singapore.*

R. McDonald, K. Crammer, and F. Pereira. 2005a. Online large-margin training of dependency parsers. In *Proceedings of ACL 2005*. pp. 91–98.

R. McDonald, F. Pereira, K. Ribarov, and J. Hajic. 2005b. Non-projective dependency parsing using spanning tree algorithms. *Proceedings of HLT/EMNLP*, pp. 523–530.

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. Nilsson and J. Nivre. 2008. Malteval: An evaluation and visualization tool for dependency parsing. In t*he Proc of Sixth International Language Resources and Evaluation,* Marrakech, Morocco.

J. Nivre and R. McDonald. 2008. Integrating graph-based and transition-based dependency parsers. In *Proc. of ACL-HLT.*

J. Nivre. 2006. *Inductive Dependency Parsing*. Springer.

K. Sagae and A. Lavie. 2006. Parser combination by reparsing. In Proc. HLT/NAACL.

S. Bergsma and C. Cherry. 2010. Fast and Accurate Arc Filtering for Dependency Parsing. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010).*

Y. Zhang and S. Clark. 2008. A tale of two parsers: Investigating and combining graph-based and transition-based dependency parsing. *In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 562-571.*