On the Role of Morphosyntactic Features in Hindi Dependency Parsing

Bharat Ram Ambati*, Samar Husain*, Joakim Nivre† and Rajeev Sangal*
*Language Technologies Research Centre, IIIT-Hyderabad, India.
†Department of Linguistics and Philology, Uppsala University, Sweden.
{bharat,samar}@research.iiit.ac.in, joakim.nivre@lingfil.uu.se, sangal@mail.iiit.ac.in

Abstract

This paper analyzes the relative importance of different linguistic features for data-driven dependency parsing of Hindi, using a feature pool derived from two state-of-the-art parsers. The analysis shows that the greatest gain in accuracy comes from the addition of morphosyntactic features related to case, tense, aspect and modality. Combining features from the two parsers, we achieve a labeled attachment score of 76.5%, which is 2 percentage points better than the previous state of the art. We finally provide a detailed error analysis and suggest possible improvements to the parsing scheme.

1 Introduction

The dependency parsing community has since a few years shown considerable interest in parsing morphologically rich languages with flexible word order. This is partly due to the increasing availability of dependency treebanks for such languages, but it is also motivated by the observation that the performance obtained for these languages has not been very high (Nivre et al., 2007a). Attempts at handling various non-configurational aspects in these languages have pointed towards shortcomings in traditional parsing methodologies (Tsarfaty and Sima'an, 2008; Eryigit et al., 2008; Seddah et al., 2009; Husain et al., 2009; Gadde et al., 2010). Among other things, it has been pointed out that the use of language specific features may play a crucial role in improving the overall parsing performance. Different languages tend to encode syntactically relevant information in different ways, and it has been hypothesized that the integration of morphological and syntactic information could be a key to better accuracy. However, it has also been noted that incorporating these language specific features in parsing is not always straightforward and many intuitive features do not always work in expected ways.

In this paper, we are concerned with Hindi, an Indian language with moderately rich morphology and relatively free word order. There have been several previous attempts at parsing Hindi as well as other Indian languages (Bharati et al., 1995, Bharati et al., 2009b). Many techniques were tried out recently at the ICON09 dependency parsing tools contest (Husain, 2009). Both the best performing system (Ambati et al., 2009a) and the system in second place (Nivre, 2009b) used a transition-based approach to dependency parsing, as implemented in MaltParser (Nivre et al., 2007b). Other data driven parsing efforts for Indian languages in the past have been Bharati et al. (2008), Husain et al. (2009), Mannem et al. (2009b) and Gadde et al. (2010).

In this paper, we continue to explore the transition-based approach to Hindi dependency parsing, building on the state-of-the-art results of Ambati et al. (2009a) and Nivre (2009b) and exploring the common pool of features used by those systems. Through a series of experiments we select features incrementally to arrive at the best parser features. The primary purpose of this investigation is to study the role of different morphosyntactic features in Hindi dependency parsing, but we also want to improve the overall parsing accuracy. Our final results are 76.5% labeled and 91.1% unlabeled attachment score, improving previous results by 2 and 1 percent absolute, respectively. In addition to this, we also provide an error analysis, isolating specific linguistic phenomena and/or other factors that impede the overall parsing performance, and suggest possible remedies for these problems.

2 The Hindi Dependency Treebank

Hindi is a free word order language with SOV as the default order. This can be seen in (1), where (1a) shows the constituents in the default order, and the remaining examples show some of the word order variants of (1a).

(1) a. malaya ne sameer ko kitaba dii.
Malay ERG Sameer DAT book gave
"Malay gave the book to Sameer" (S-IO-DO-V)[1]
b. malaya ne kitaba sameer ko dii. (S-DO-IO-V)
c. sameer ko malaya ne kitaba dii. (IO-S-DO-V)
d. sameer ko kitaba malaya ne dii. (IO-DO-S-V)
e. kitaba malaya ne sameer ko dii. (DO-S-IO-V)
f. kitaba sameer ko malaya ne dii. (DO-IO-S-V)

Hindi also has a rich case marking system, although case marking is not obligatory. For example, in (1), while the subject and indirect object are explicitly marked for the ergative (ERG) and dative (DAT) cases, the direct object is unmarked for the accusative.

The Hindi dependency treebank (Begum et al., 2008) used for the experiment was released as part of the ICON09 dependency parsing tools contest (Husain, 2009). The dependency framework (Bharati et al., 1995) used in the treebank is inspired by Panini's grammar of Sanskrit. The core labels, called *karakas*, are syntactico-semantic relations that identify the participant in the action denoted by the verb. For example, in (1), 'Malay' is the agent, 'book' is the theme, and 'Sameer' is the beneficiary in the activity of 'give'. In the treebank, these three labels are marked as k1, k2, and k4 respectively. Note, however, that the notion of *karaka* does not capture the 'global' semantics of thematic roles; rather it captures the elements of the 'local semantics' of a verb, while also taking cues from the surface level morpho-syntactic information (Vaidya et al., 2009). The syntactic relational cues (such as case markers) help identify many of the karakas. In general, the highest available karaka,[2] if not case-marked, agrees with the verb in an active sentence. In addition, the tense, aspect and modality (TAM) marker can many a times control the case markers that appear on k1. For example, in (1) 'Malay' takes an ergative case because of the past perfective TAM marker (that appears as a suffix in this case) of the main verb 'gave'. Many dependency relations other than karakas are purely syntactic. These include relations such as noun modifier (nmod), verb modifier (vmod), conjunct relation (ccof), etc.

Each sentence is manually chunked and then annotated for dependency relations. A chunk is a minimal, non-recursive structure consisting of correlated groups of words (Bharati et al., 2006). A node in a dependency tree represents a chunk head. Each lexical item in a sentence is also annotated with its part-of-speech (POS). For all the experiments described in this paper we use gold POS and chunk tags. Together, a group of lexical items with some POS tags within a chunk can be utilized to automatically compute coarse grained morphosyntactic information. For example, such information can represent the postposition/case-marking in the case of noun chunks, or it may represent the TAM information in the case of verb chunks. In the experiments conducted for this paper this local information is automatically computed and incorporated as a feature of the head of a chunk. As we will see later, such information proves to be extremely crucial during dependency parsing.

For all the experiments discussed in section 4, the training and development data size was 1500 and 150 sentences respectively. The training and development data consisted of ~22k and ~1.7k words respectively. The test data consisted of 150 sentences (~1.6k words). The average sentence length is 19.85.

3 Transition-Based Dependency Parsing

A transition-based dependency parser is built of two essential components (Nivre, 2008):

- A transition system for mapping sentences to dependency trees
- A classifier for predicting the next transition for every possible system configuration

[1] S=Subject; IO=Indirect Object; DO=Direct Object; V=Verb; ERG=Ergative; DAT=Dative
[2] These are the *karta karaka* (k1) and *karma karaka* (k2). k1 and k2 can be roughly translated as 'agent' and 'theme' respectively. For a complete description of the tagset and the dependency scheme, see Begum et al. (2008) and Bharati et al. (2009a).

	PTAG	CTAG	FORM	LEMMA	DEPREL	CTAM	OTHERS
Stack: *top*	1	5	1	7		9	
Input: *next*	1	5	1	7		9	
Input: *next*+1	2	5	6	7			
Input: *next*+2	2						
Input: *next*+3	2						
Stack: *top*-1	3						
String: predecessor of *top*	3						
Tree: head of *top*	4						
Tree: leftmost dep of *next*	4	5	6				
Tree: rightmost dep of *top*					8		
Tree: left sibling of rightmost dep of *top*					8		
Merge: PTAG of *top* and *next*							10
Merge: CTAM and DEPREL of *top*							10

Table 1. Feature pool based on selection from Ambati et al. (2009a) and Nivre (2009b).

Given these two components, dependency parsing can be realized as deterministic search through the transition system, guided by the classifier. With this technique, parsing can be performed in linear time for projective dependency trees. Like Ambati et al. (2009a) and Nivre (2009b), we use MaltParser, an open-source implementation of transition-based dependency parsing with a variety of transition systems and customizable classifiers.[3]

3.1 Transition System

Previous work has shown that the arc-eager projective transition system first described in Nivre (2003) works well for Hindi (Ambati et al., 2009a; Nivre, 2009b). A parser configuration in this system contains a stack holding partially processed tokens, an input buffer containing the remaining tokens, and a set of arcs representing the partially built dependency tree. There are four possible transitions (where *top* is the token on top of the stack and *next* is the next token in the input buffer):

- **Left-Arc(*r*)**: Add an arc labeled *r* from *next* to *top*; pop the stack.
- **Right-Arc(*r*)**: Add an arc labeled *r* from *top* to *next*; push *next* onto the stack.
- **Reduce**: Pop the stack.
- **Shift**: Push *next* onto the stack.

Although this system can only derive projective dependency trees, the fact that the trees are labeled allows non-projective dependencies to be captured using the pseudo-projective parsing technique proposed in Nivre and Nilsson (2005).

3.2 Classifiers

Classifiers can be induced from treebank data using a wide variety of different machine learning methods, but all experiments reported below use support vector machines with a polynomial kernel, as implemented in the LIBSVM package (Chang and Lin, 2001) included in MaltParser. The task of the classifier is to map a high-dimensional feature vector representation of a parser configuration to the optimal transition out of that configuration. The features used in our experiments represent the following attributes of input tokens:

- PTAG: POS tag of chunk head.
- CTAG: Chunk tag.
- FORM: Word form of chunk head.
- LEMMA: Lemma of chunk head.
- DEPREL: Dependency relation of chunk.
- CTAM: Case and TAM markers of chunk.

The PTAG corresponds to the POS tag associated with the head of the chunk, whereas the CTAG represent the chunk tag. The FORM is the word form of the chunk head, and the LEMMA is automatically computed with the help of a morphological analyzer. CTAM gives the local morphosyntactic features such as case markers (postpositions/suffixes) for nominals and TAM markers for verbs (cf. Section 2).

[3] MaltParser is available at http://maltparser.org.

The pool of features used in the experiments are shown in Table 1, where rows denote tokens in a parser configuration – defined relative to the stack, the input buffer, the partially built dependency tree and the input string – and columns correspond to attributes. Each non-empty cell represents a feature, and features are numbered for easy reference.

4 Feature Selection Experiments

Starting from the union of the feature sets used by Ambati et al. (2009a and by Nivre (2009b), we first used 5-fold cross-validation on the combined training and development sets from the ICON09 tools contest to select the pool of features depicted in Table 1, keeping all features that had a positive effect on both labeled and unlabeled accuracy. We then grouped the features into 10 groups (indicated by numbers 1–10 in Table 1) and reran the cross-validation, incrementally adding different feature groups in order to analyze their impact on parsing accuracy. The result is shown in Figure 1.

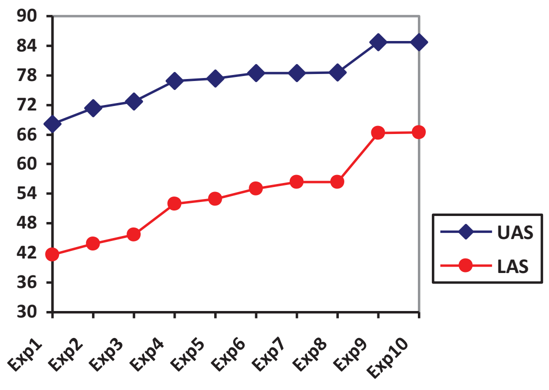


Figure 1. UAS and LAS of experiments 1-10; 5-fold cross-validation on training and development data of the ICON09 tools contest.

Experiment 1: Experiment 1 uses a baseline model with only four basic features: PTAG and FORM of *top* and *next*. This results in a labeled attachment score (LAS) of 41.7% and an unlabeled attachment score (UAS) of 68.2%.

Experiments 2–3: In experiments 2 and 3, the PTAG of contextual words of *next* and *top* are added. Of all the contextual words, *next+1*, *next+2, next+3, top-1* and predecessor of *top* were found to be useful.[4] Adding these contextual features gave a modest improvement to 45.7% LAS and 72.7% UAS.

Experiment 4: In experiment 4, we used the PTAG information of nodes in the partially built tree, more specifically the syntactic head of *top* and the leftmost dependent of *next*. Using these features gave a large jump in accuracy to 52% LAS and 76.8% UAS. This is because partial information is helpful in making future decisions. For example, a coordinating conjunction can have a node of any PTAG category as its child. But all the children should be of same category. Knowing the PTAG of one child therefore helps in identifying other children as well.

Experiments 5–7: In experiments 5, 6 and 7, we explored the usefulness of CTAG, FORM, and LEMMA attributes. These features gave small incremental improvements in accuracy; increasing LAS to 56.4% and UAS to 78.5%. It is worth noting in particular that the addition of LEMMA attributes only had a marginal effect on accuracy, given that it is generally believed that this type of information should be beneficial for richly inflected languages.

Experiment 8: In experiment 8, the DEPREL of nodes in the partially formed tree is used. The rightmost child and the left sibling of the rightmost child of *top* were found to be useful. This is because, if we know the dependency label of one of the children, then the search space for other children gets reduced. For example, a verb cannot have more than one k1 or k2. If we know that the parser has assigned k1 to one of its children, then it should use different labels for the other children. The overall effect on parsing accuracy is nevertheless very marginal, bringing LAS to 56.5% and UAS to 78.6%.

Experiment 9: In experiment 9, the CTAM attribute of *top* and *next* is used. This gave by far the greatest improvement in accuracy with a huge jump of around 10% in LAS (to 66.3%) and slightly less in UAS (to 84.7%). Recall that CTAM consists of two important morphosyntactic features, namely, case markers (as suffixes or postpositions) and TAM markers. These feature help because (a) case markers are important surface

[4] The predecessor of *top* is the word occurring immediately before *top* in the input string, as opposed to *top-1*, which is the word immediately below *top* in the current stack.

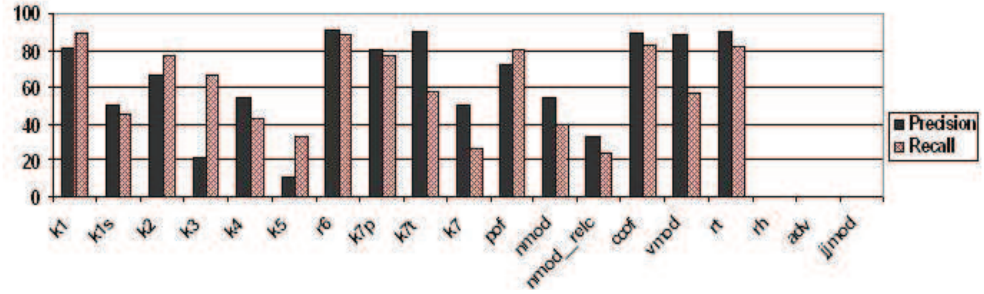


Figure 2. Precision and Recall of some important dependency labels.

cues that help identify various dependency relations, and (b) there exists a direct mapping between many TAM labels and the nominal case markers because TAMs control the case markers of some nominals. As expected, our experiments show that the parsing decisions are certainly more accurate after using these features. In particular, (a) and (b) are incorporated easily in the parsing process.

In a separate experiment we also added some other morphological features such as gender, number and person for each node. Through these features we expected to capture the agreement in Hindi. The verb agrees in gender, number and person with the highest available karaka. However, incorporating these features did not improve parsing accuracy and hence these features were not used in the final setting. We will have more to say about agreement in section 5.

Experiment 10: In experiment 10, finally, we added conjoined features, where the conjunction of POS of *next* and *top* and of CTAM and DEPREL of *top* gave slight improvements. This is because a child-parent pair type can only take certain labels. For example, if the child is a noun and the parent is a verb, then all the dependency labels reflecting noun, adverb and adjective modifications are not relevant. Similarly, as noted earlier, certain case-TAM combinations demand a particular set of labels only. This can be captured by the combination tried in this experiment.

Experiment 10 gave the best results in the cross-validation experiments. The settings from this experiment were used to get the final performance on the test data. Table 2 shows the final results along with the results of the first and second best performing systems in the ICON09 tools contest. We see that our system achieved an improvement of 2 percentage points in LAS and 1 percentage point in UAS over the previous state of the art reported in Ambati et al. (2009a).

System	**LAS**	**UAS**
Ambati et al. (2009a)	74.5	90.1
Nivre (2009b)	73.4	89.8
Our system	76.5	91.1

Table 2. Final results on the test data from the ICON09 tools contest.

5 Error Analysis

In this section we provide a detailed error analysis on the test data and suggest possible remedies for problems noted. We note here that other than the reasons mentioned in this section, small treebank size could be another reason for low accuracy of the parser. The training data used for the experiments only had ~28.5k words. With recent work on Hindi Treebanking (Bhatt et al., 2009) we expect to get more annotated data in the near future.

Figure 2 shows the precision and recall of some important dependency labels in the test data. The labels in the treebank are syntacto-semantic in nature. Morph-syntactic features such as case markers and/or TAM labels help in identifying these labels correctly. But lack of nominal postpositions can pose problems. Recall that many case markings in Hindi are optional. Also recall that the verb agrees with the highest available karaka. Since agreement features do not seem to help, if both k1 and k2 lack case markers, k1-k2 disambiguation becomes difficult (considering that word order information cannot help in this disambiguation). In the case of k1 and k2, error rates for instances that lack post-position markers are 60.9% (14/23) and 65.8% (25/38), respectively.

	Correct	Incorrect							
		k1	k1s	k2	pof	k7p	k7t	k7	others
k1	184	5	3	8	3		1		3
k1s	12	6		1	6				1
k2	126	14		1	7	5			11
pof	54	1	8	4					
k7p	54	3		7			1	2	3
k7t	27	3		3	3		1		10
k7	3	2			2	4			

Table 3. Confusion matrix for important labels. The diagonal under 'Incorrect' represents attachment errors.

Table 3 shows the confusion matrix for some important labels in the test data. As the present information available for disambiguation is not sufficient, we can make use of some semantics to resolve these ambiguities. Bharati et al. (2008) and Ambati et al. (2009b) have shown that this ambiguity can be reduced using minimal semantics. They used six semantic features: human, non-human, in-animate, time, place and abstract. Using these features they showed that k1-k2 and k7p-k7t ambiguities can be resolved to a great extent. Of course, automatically extracting these semantic features is in itself a challenging task, although Øvrelid (2008) has shown that animacy features can be induced automatically from data.

In section 4 we mentioned that a separate experiment explored the effectiveness of morphological features like gender, number and person. Counter to our intuitions, these features did not improve the overall accuracy. Accuracies on cross-validated data while using these features were less than the best results with 66.2% LAS and 84.6% UAS. Agreement patterns in Hindi are not straightforward. For example, the verb agrees with k2 if the k1 has a post-position; it may also sometimes take the default features. In a passive sentence, the verb agrees only with k2. The agreement problem worsens when there is coordination or when there is a complex verb. It is understandable then that the parser is unable to learn the selective agreement pattern which needs to be followed. Similar problems with agreement features have also been noted by Goldberg and Elhadad (2009).

In the following sections, we analyze the errors due to different constructions and suggest possible remedies.

5.1 Simple Sentences

A simple sentence is one that has only one main verb. In these sentences, the root of the dependency tree is the main verb, which is easily identified by the parser. The main problem is the correct identification of the argument structure. Although the attachments are mostly correct, the dependency labels are error prone. Unlike in English and other more configurational languages, one of the main cues that help us identify the arguments is to be found in the nominal postpositions. Also, as noted earlier these postpositions are many times controlled by the TAM labels that appear on the verb. There are four major reasons for label errors in simple sentences: (a) absence of postpositions, (b) ambiguous postpositions, (c) ambiguous TAMs, and (d) inability of the parser to exploit agreement features. For example in (2), *raama* and *phala* are arguments of the verb *khaata*. Neither of them has any explicit case marker. This makes it difficult for the parser to identify the correct label for these nodes. In (3a) and (3b) the case marker *se* is ambiguous. It signifies 'instrument' in (3b) and 'agent' in (3a).

(2) raama phala khaata hai
'Ram' 'fruit' 'eat' 'is'
'Ram eats a fruit'

(3) a. raama se phala khaayaa nahi gaya
'Ram' INST 'fruit' 'eat' 'not' 'PAST'
'Ram could not eat the fruit'
b. raama chamach se phala khaata hai
'Ram' 'spoon' INST 'fruit' 'eat' 'is'
'Ram eats fruit with spoon'

5.2 Embedded Clauses

Two major types of embedded constructions involve participles and relative clause constructions. Participles in Hindi are identified through a set of TAM markers. In the case of participle embeddings, a sentence will have more than one verb, i.e., at least one participle and the matrix verb. Both the matrix (finite) verb and the participle can take their own arguments that can be identified via the case-TAM mapping discussed earlier. However, there are certain syntactic constraints that limit the type of arguments a participle can take. There are two sources of errors here: (a) argument sharing, and (b) ambiguous attachment sites.

Some arguments such as place/time nominals can be shared. Shared arguments are assigned to only one verb in the dependency tree. So the task of identifying the shared arguments, if any, and attaching them to the correct parent is a complex task. Note that the dependency labels can be identified based on the morphosyntactic features. The task becomes more complex if there is more than one participle in a sentence. 12 out of 130 instances (9.23%) of shared arguments has an incorrect attachment.

Many participles are ambiguous and making the correct attachment choice is difficult. Similar participles, depending on the context, can behave as adverbials and attach to a verb, or can behave as adjectives and attach to a noun. Take (4) as a case in point.

(4)	maine	daurte	hue	kutte	ko	dekhaa
	'I'-ERG	(while)	running	dog	ACC	'saw'

In (4) based on how one interprets 'daurte hue', one gets either the reading that 'I saw a running dog' or that 'I saw a dog while running'. In case of the adjectival participle construction (VJJ), 2 out of 3 errors are due to wrong attachment.

5.3 Coordination

Coordination poses problems as it often gives rise to long-distance dependencies. Moreover, the treebank annotation treats the coordinating conjunction as the head of the coordinated structure. Therefore, a coordinating conjunction can potentially become the root of the entire dependency tree. This is similar to Prague style dependency annotation (Hajicova, 1998). Coordinating conjunctions pose additional problems in such a scenario as they can appear as the child of different heads. A coordinating conjunction takes children of similar POS category, but the parent of the conjunction depends on the type of the children.

(5) a.	raama	aur	shyaama	ne	khaana	khaayaa
	'Ram'	'and'	'Shyam'	'ERG'	'food'	'ate'

'Ram and Shyam ate the food.'

b.	raama	ne	khaanaa	khaayaa	aur	paanii	piyaa
	'Ram'	'ERG'	'food'	'ate'	'and'	'water'	'drank'

'Ram ate food and drank water.'

In (5a), *raama* and *shyaama* are children of the coordinating conjunction *aur*, which gets attached to the main verb *khaayaa* with the label k1. In effect, syntactically *aur* becomes the argument of the main verb. In (5b), however, the verbs *khaayaa* and *piyaa* are the children of *aur*. In this case, *aur* becomes the root of the sentence. Identifying the nature of the conjunction and its children becomes a challenging task for the parser. Note that the number of children that a coordinating conjunction can take is not fixed either. The parser could identify the correct head of the conjunctions with an accuracy of 75.7% and the correct children with an accuracy of 85.7%.

The nature of the conjunction will also affect the dependency relation it has with its head. For example, if the children are nouns, then the conjunction behaves as a noun and can potentially be an argument of a verb. By contrast, if the children are finite verbs, then it behaves as a finite verb and can become the root of the dependency tree. Unlike nouns and verbs, however, conjunctions do not have morphological features. So a child-to-head feature percolation should help make a coordinating node more transparent. For example, in (5a) the Ergative case *ne* is a strong cue for the dependency label k1. If we copy this information from one of its children (here *shyaama*) to the conjunct, then the parser can possibly make use of this information.

5.4 Complex Predicates

Complex predicates are formed by combining a noun or an adjective with a verbalizer *kar* or *ho*. For instance, in *taariif karanaa* 'to praise', *taariif* 'praise' is a noun and *karanaa* 'to do' is a verb. Together they form the main verb. Complex predicates are highly productive in Hindi. Combination of the light verb and the noun/adjective is dependent on not only syntax but also semantics and therefore its automatic identification is not always straightforward (Butt, 1995). A noun-verb complex predicate in the treebank is linked via the dependency label *pof*. The parser makes mistakes in identifying pof or misclassifies other labels as pof. In particular, the confusion is with k2 and k1s which are object/theme and noun complements of k1, respectively. These labels share similar contextual features like the nominal element in the verb complex. Table 3 includes the confusion matrix for pof errors.

5.5 Non-Projectivity

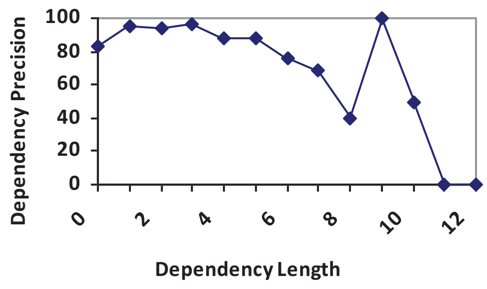
As noted earlier, MaltParser's arc-eager parsing algorithm can be combined with the pseudo-projective parsing techniques proposed in Nivre and Nilsson (2005), which potentially helps in identifying non-projective arcs. The Hindi treebank has ~14% non-projective arcs (Mannem et al., 2009a). In the test set, there were a total of 11 non-projective arcs, but the parser did not find any of them. This is consistent with earlier results showing that pseudo-projective parsing has high precision but low recall, especially when the percentage of non-projective relations is small (Nilsson et al, 2007).

Non-projectivity has proven to be one of the major problems in dependency parsing, especially for free word-order languages. In Hindi, the majority of non-projective arcs are inter-clausal (Mannem et al., 2009a), involving conjunctions and relative clauses. There have been some attempts at handling inter-clausal non-projectivity in Hindi. Husain et al. (2009) proposed a two-stage approach that can handle some of the inter-clausal non-projective structures.

5.6 Long-Distance Dependencies

Previous results on parsing other languages have shown that MaltParser has lower accuracy on long-distance dependencies. Our results confirm this. Errors in the case of relative clauses and coordination can mainly be explained in this way. For example, there are 8 instances of relative clauses in the test data. The system could identify only 2 of them correctly. These two are at a distance of 1 from its parent. For the remaining 6 instances the distance to the parent of the relative clause ranges from 4 to 12.

Figure 3 shows how parser performance decreases with increasing distance between the head and the dependent. Recently, Husain et al. (2009) have proposed a two-stage setup to parse inter-clausal and intra-clausal dependencies separately. They have shown that most long distance relations are inter-clausal, and therefore, using such a clause motivated parsing setup helps in maximizing both short distance and long distance dependency accuracy. In a similar spirit, Gadde et al. (2010) showed that using clausal features helps in identifying long distance dependencies. They have shown that providing clause information in the form of clause boundaries and clausal heads can help a parser make better predictions about long distance dependencies.



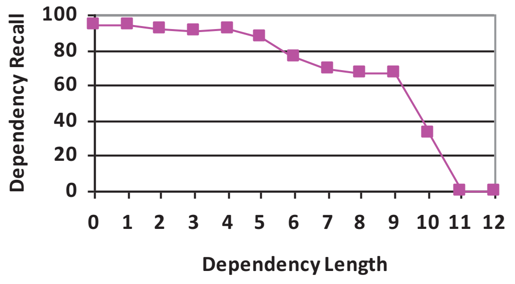


Figure 3. Dependency arc precision/recall relative to dependency length, where the length of a dependency from w_i to w_j is $|i-j|$ and roots are assumed to have distance 0 to their head.

6 Conclusion

In this paper we have analyzed the importance of different linguistic features in data-driven parsing of Hindi and at the same time improved the state of the art. Our main finding is that the combination of case markers on nominals with TAM markers on verbs is crucially important for syntactic disambiguation, while the inclusion of features such as person, number gender that help in agreement has not yet resulted in any improvement. We have also presented a detailed error analysis and discussed possible techniques targeting different error classes. We plan to use these techniques to improve our results in the near future.

References

B. R. Ambati, P. Gadde, and K. Jindal. 2009a. Experiments in Indian Language Dependency Parsing. *Proc. of ICON09 NLP Tools Contest: Indian Language Dependency Parsing*, 32-37.

B. R. Ambati, P. Gade, C. GSK and S. Husain. 2009b. Effect of Minimal Semantics on Dependency Parsing**.** *Proc. of RANLP Student Research Workshop.*

R. Begum, S. Husain, A. Dhwaj, D. Sharma, L. Bai, and R. Sangal. 2008. Dependency annotation scheme for Indian languages. *Proc. of IJCNLP.*

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. *Proc. of ICON.*

A. Bharati, R. Sangal, D. M. Sharma and L. Bai. 2006. AnnCorra: Annotating Corpora Guidelines for POS and Chunk Annotation for Indian Languages. *Technical Report (TR-LTRC-31), LTRC, IIIT-Hyderabad.*

A. Bharati, D. M. Sharma, S. Husain, L. Bai, R. Begam and R. Sangal. 2009a. AnnCorra: TreeBanks for Indian Languages, Guidelines for Annotating Hindi TreeBank**.** http://ltrc.iiit.ac.in/MachineTrans/research/tb/DS-guidelines/DS-guidelines-ver2-28-05-09.pdf

A. Bharati, S. Husain, D. M. Sharma and R. Sangal. 2009b. Two stage constraint based hybrid approach to free word order language dependency parsing. *In Proc. of IWPT.*

R. Bhatt, B. Narasimhan, M. Palmer, O. Rambow, D. M. Sharma and F. Xia. 2009. Multi-Representational and Multi-Layered Treebank for Hindi/Urdu. *Proc. of the Third LAW at ACL-IJCNLP,* 186-189.

M. Butt. 1995. *The Structure of Complex Predicates in Urdu.* CSLI Publications.

G. Eryigit, J. Nivre, and K. Oflazer. 2008. Dependency Parsing of Turkish. *Computational Linguistics* 34(3), 357-389.

P. Gadde, K. Jindal, S. Husain, D. M. Sharma, and R. Sangal. 2010. Improving Data Driven Dependency Parsing using Clausal Information. *Proc. of NAACL-HLT.*

Y. Goldberg and M. Elhadad. 2009. Hebrew Dependency Parsing: Initial Results. *Proc. of IWPT,* 129-133.

E. Hajicova. 1998. Prague Dependency Treebank: From Analytic to Tectogrammatical Annotation. *Proc. of TSD*.

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. *Proc. of EMNLP-CoNLL,* 933-939.

S. Husain. 2009. Dependency Parsers for Indian Languages. *Proc. of ICON09 NLP Tools Contest: Indian Language Dependency Parsing.*

S. Husain, P. Gadde, B. Ambati, D. M. Sharma and R. Sangal. 2009. A modular cascaded approach to complete parsing. *Proc. of the COLIPS International Conference on Asian Language Processing.*

P. Mannem, H. Chaudhry, and A. Bharati. 2009a. Insights into non-projectivity in Hindi. *Proc. of ACL-IJCNLP Student Research Workshop.*

P. Mannem, A. Abhilash and A. Bharati. 2009b. LTAG-spinal Treebank and Parser for Hindi. *Proceedings of International Conference on NLP, Hyderabad. 2009*.

R. McDonald, K. Lerman, and F. Pereira. 2006. Multilingual dependency analysis with a two-stage discriminative parser. *Proc. of CoNLL*, 216-220.

R. McDonald and J. Nivre. 2007. Characterizing the errors of data-driven dependency parsing models. *Proc. of EMNLP-CoNLL,* 122-131.

I. A. Mel'Cuk. 1988. *Dependency Syntax: Theory and Practice*, State University Press of New York.

J. Nilsson, J. Nivre and J. Hall. 2007. Generalizing Tree Transformations for Inductive Dependency Parsing. *Proc. of ACL*, 968-975.

J. Nivre. 2008. Algorithms for Deterministic Incremental Dependency Parsing. *Computational Linguistics* 34(4), 513-553.

J. Nivre and R. McDonald. 2008. Integrating graph-based and transition-based dependency parsers. In *Proceedings of ACL-HLT*, pp. 950-958.

J. Nivre. 2009a. Non-Projective Dependency Parsing in Expected Linear Time. *Proc. of ACL-IJCNLP*, 351-359.

J. Nivre. 2009b. Parsing Indian Languages with MaltParser. *Proc. of ICON09 NLP Tools Contest: Indian Language Dependency Parsing*, 12-18.

J. Nivre, J. Hall, S. Kubler, R. McDonald, J. Nilsson, S. Riedel and D. Yuret. 2007a. The CoNLL 2007 Shared Task on Dependency Parsing. *Proc. of EMNLP/CoNLL,* 915-932.

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. *NLE*, 13(2), 95-135.

L. Øvrelid. 2008. *Argument Differentiation. Soft constraints and data-driven models.* PhD Thesis, University of Gothenburg.

D. Seddah, M. Candito and B. Crabbé. 2009. Cross parser evaluation: a French Treebanks study. *Proc. of IWPT,* 150-161.

R. Tsarfaty and K. Sima'an. 2008. Relational-Realizational Parsing. *Proc. of CoLing,* 889-896.

A. Vaidya, S. Husain, P. Mannem, and D. M. Sharma. 2009. A karaka-based dependency annotation scheme for English**.** *Proc. of CICLing,* 41-52.