

A Constraint Based Hybrid Dependency Parser for Telugu

Sruthilaya Reddy Kesidi, Prudhvi Kosaraju, Meher Vijay and Samar Husain

Language Technologies Research Centre, IIIT-Hyderabad, India.
{sruthilaya, prudhvi}@students.iiit.ac.in, {mehervijay.yeleti, samar}@research.iiit.ac.in

Abstract. In this paper we present a two stage constraint based approach to Telugu dependency parsing. We define the two stage and show how different Telugu constructions are parsed at appropriate stages. The division leads to selective identification and resolution of specific dependency relations at the two stages. We then discuss the ranking strategy that makes use of the S-constraints to gives us the best parse. A detailed error analysis of the output of core parser and the ranker helps us to flesh out the current issues and future directions.

Keywords: Telugu Dependency Parser, Constraint Based Parsing, Parse Ranking, Hybrid Approach, Morphologically rich free word order parsing

1 Introduction

There has been a recent surge in addressing parsing for morphologically rich free word order languages such as Czech, Turkish, Hindi, etc. These languages pose various challenges for the task of parsing mainly because the syntactic cues necessary to identify various relations are complex and distributed [14], [1], [10]. [11], [13], [12], [6], [9], [5]. Data driven parser performance [7] show that many of these complex phenomena are not being captured by the present parsers. Constraint based parsers have been known to have the power to account for difficult constructions and are well suited for handling complex phenomena. Some of the recent constraint based systems have been [20], [24], [36], [37], [31], [28], [23] and [17].

In this paper we present a two stage constraint based approach to Telugu dependency parsing based on the parser proposed by [3], [4], [35] We begin by quickly summarizing the parser design and follow it by describing how different Telugu constructions are parsed at appropriate stages. We will see that this division leads to selective identification and resolution of specific dependency relations at the two stages. We then discuss the ranking strategy that makes use of the S-constraints to gives us the best parse. A detailed error analysis of the output of core parser and the ranker helps us to flesh out the current issues and future directions.

This paper is arranged as follows: section 2 gives a quick overview of the two-stage constraint based hybrid parser (CBHP); section 3 explains in details how we handle different Telugu constructs in two stages, section 4 describes the results of the

core parser and makes some observations on these results. In section 5 we describe the ranking strategies, followed with results in Section 6. We conclude the paper with future directions in section 7.

2 Overview of the two-stage constraint based hybrid parser (CBHP)

CBHP [3], [4] adopts a hybrid approach to dependency parsing. A constraint based approach is supported by a controlled statistical strategy to achieve high performance and robustness. The overall task of dependency parsing is attacked using modularity, wherein specific tasks are broken down into smaller linguistically motivated sub-tasks. Figure 1 shows the overall design of CBHP. Below we quickly summarize CBHP:

(1) *Two stages during the sentence analysis phase*: The parser tries to analyze the given input sentence, which has already been tagged and chunked¹, in two stages; it first tries to extract intra-clausal² dependency relations. In the second stage it then tries to handle more complex inter-clausal relations such as those involved in constructions of coordination and subordination between clauses.

(2) *H-constraints*: Both 1st and 2nd stage use linguistically motivated constraints. These H- constraints (Hard constraints) reflect that aspect of the grammar which in general cannot be violated. H-constraints mainly comprised of structural and lexical knowledge of the language.

(3) *S-constraints and Prioritization for parse selection*: The sentence analysis layer (cf. figure 1) can potentially produce more than one parse. These parses are then ranked by a prioritization component using S-constraints(Soft constraints). S-constraints are the constraints that are used in the language as preferences. Unlike the H-constraints that are derived from a knowledge base and are used within the core parser during derivation, S-constraints have weights assigned to them and are used exclusively during prioritization. These weights are automatically learnt using a manually annotated dependency treebank.

3 CBHP for Telugu

[3], [4] have proposed CBHP for Indian languages. It has however been tested only for Hindi. We use the same machinery that was used by them to handle Telugu. We

¹ POS and chunk tagging scheme for Indian Languages is discussed in [38].

²A clause is a group of word such that the group contains a single finite verb and its dependents. We must note here that these dependents cannot themselves be finite verbs. Also, subordinating conjunctions and finite verb coordinating conjunctions are also not treated as part of a clause. And therefore, a sentence such as ‘*John said that He will come late*’ has 3 units; (1) *John said*, (2) *that*, and (3) *He will come late*. Similarly, ‘*John ate his food and he went shopping*’ has 3 units; (1) *John ate his food*, (2) *and*, and (3) *he went shopping*.

first describe the different types of relations the parser currently handles in the two stages, following which we briefly mention H-constraints for Telugu CBHP.

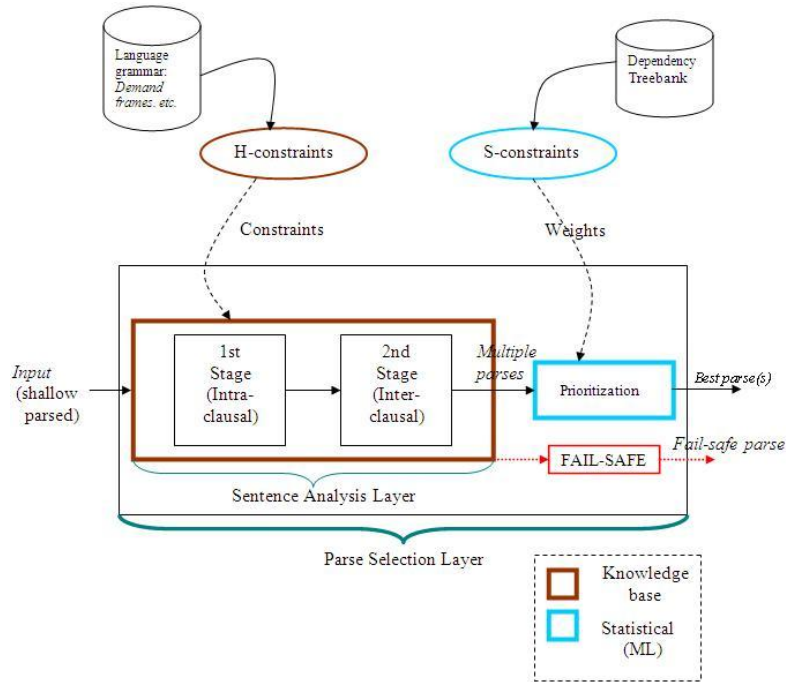


Fig 1. Design of CBHP

3.1 Relations handled in Stage 1

In stage 1 we handle mainly intra-clausal relations. These relations represent:

- i. Argument structure of the finite verb
- ii. Argument structure of the non-finite verb
- iii. Adjuncts of finite and the non-finite verb
- iv. Noun modifications
- v. Adjectival modifications

Using example (1) below we describe how the parser handles a simple declarative sentence.

- (1) *wulasi golilu mAnesiMxi*
 ‘Tulasi’ ‘tablets’ ‘stopped using’
 ‘Tulasi stopped using the tablets’

CBHP begins by constructing the constraint graph (CG) for the above sentence. A constraint graph shows all the potential arcs that can exist between the heads and their

corresponding dependents. This information is obtained from the demand frame of the head. According to the frame the verb can take $k1^3$ and $k2$ as mandatory children with null postpositions. Figure 2 shows that the demand frame for *mAnesiMxi* is obtained from basic demand frame of root verb *mAnu* and the null frame of suffix *-esiMxi* (which represent the tense, aspect and modality (TAM)). The basic demand frame for a verb or a class of verbs specifies the suffix, category, etc. permitted for the allowed relations for a verb when the verb has the basic TAM label. In Figure 3(a) we show the constraint graph that is constructed using the demand frame. The final parses are obtained by converting the CG into integer programming equations and using bipartite graph matching [36], [37], [4]. Figure 3(b) shows the final parses obtained from the CG. Notice that we get two parses. This indicates the ambiguity in the sentence if only the limited knowledge base is considered. Stated another way, H-constraints (in the form of demand frames) are insufficient to restrict multiple analysis of a given sentence and that more knowledge (semantics, other preferences, etc.) is required to curtail the ambiguities. We will see in Section 5 how we can obtain the best parse from these multiple parses. Notice also the presence of the `_ROOT_` node. By introducing `_ROOT_` we are able to attach all unprocessed nodes to it. `_ROOT_` ensures that the output we get after each stage is a tree.

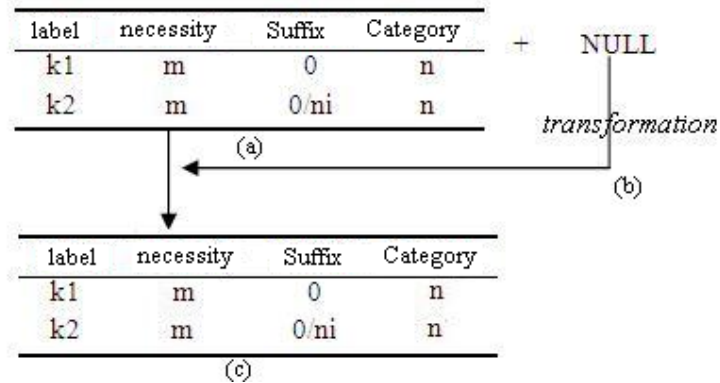


Fig 2. Final demand frame for *mAnesiMxi* shown in (c) is obtained from the basic demand frame of *manu* (a) and the transformation (b) which is NULL here.

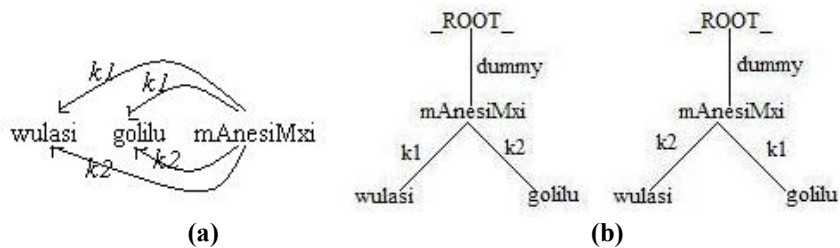


Fig 3. (a) Constraint graph for (1). (b) Final parses obtained from CG

³ $k1$ can be roughly translated to ‘agent’, $k2$ can be roughly translated as ‘theme’. CBHP follows the syntactic-semantic dependency labels proposed in [3].

Example (2) is a slightly complex construction containing a non-finite verb. Such sentences are also handled in the 1st stage.

- (2) *wulasi rogaM waggiMxani golilu mAnesiMxi.*
 ‘Tulasi’ ‘disease’ ‘having reduced-so’ ‘tablets’ ‘stopped using’.
 ‘As the disease reduced, Tulasi stopped using the tablets’

We have already seen through (1) how a sentence with a finite verb like *mAnesiMxi* can be parsed. In (2) in addition to a finite verb we have a non-finite verb *waggiMxani*. Like last time we first get the basic demand frame of the non-finite verb *waggiMxani*. Basic demand frames always represent the argument structure of a verb with its default TAM (present, indefinite). When a new TAM appears with the verb, we transform the original frame to get the final frame that is accessed during the construction of the CG. Figure 4 shows this process clearly. Figure 5 shows the most appropriate final parse.

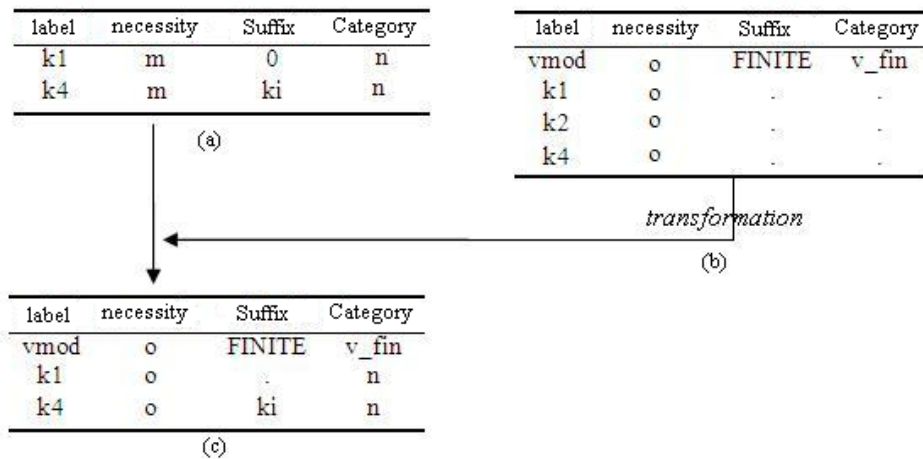


Fig 4. Basic Demand frame of *waggiMxani* (a) is transformed by the transformation frame *A_ani* (b) giving us the final frame for *waggiMxani* shown in (c).

3.2 Relations handled in Stage 2

Stage 2 handles more complex inter-clausal relations such as those involved in constructions of coordination and subordination between clauses. Stage 2 handles the following relations:

- i. Clausal coordination
- ii. Clausal subordination
- iii. Non-clausal coordination
- iv. Clausal complement

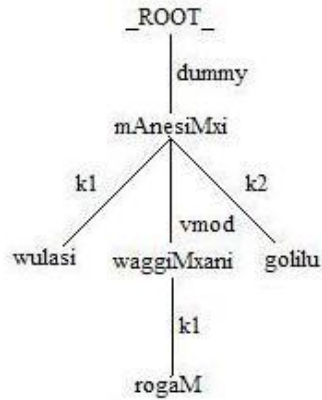


Fig 5. Appropriate final parse

2nd stage functions similar to the 1st stage, except that in the 2nd stage the CG has very few nodes. This is because the 1st stage parse subtrees are mostly not modified in the 2nd stage and only those nodes that are relevant for 2nd stage relations are considered. Take (3) as a case in point.

- (3) *wulasi golilu mAnesiMxi mariyu paniki velYliMxi*
 ‘Tulasi’ ‘tablets’ ‘stopped using’ ‘and’ ‘work’ ‘went’
 ‘Tulasi stopped using tablets and went to work’

Figure 7(a) shows the first stage parse for this sentence. We can see that the analysis of the two clauses is complete. The root of these subtrees and the conjunct *mariyu* are seen attached to the *_ROOT_*. Figure 6 shows the 2nd stage CG for (3). Notice that only the two finite verbs and the conjunct are seen here. The relations identified in the 1st stage remain untouched. Figure 7(b) also shows the final parse for this sentence; notice that the outputs of the two stages are combined to get the final parse. Merging the 1st stage and 2nd stage is not problematic as the two stages are mutually exclusive.

We must note here that for few cases such as (4) below, the 1st stage output is revised. We follow the same principles as described in [35]. This mainly concerns case dropping in nominal coordinations. Example 5 shows an example of clausal complement where the verb *ceVppiMxi* takes the clause headed by a complementizer *ani*, the complementizer in turn takes a clause headed by *mAnesiMxi*. Figure 8(a) and 8(b) shows the 1st and 2nd stage output of (5) respectively. The 2nd stage CG can be seen in Figure 9.

- (4) *wulasi mariyu rama golilu mAnesAru.*
 ‘Tulasi’ ‘and’ ‘Rama’ ‘tablets’ ‘stopped using’
 ‘Tulasi and Rama stopped using tablets.’
- (5) *wulasi golilu mAnesiMxi ani rama ceVppiMxi*
 ‘tulasi’ ‘tablets’ ‘stopped using’ ‘’ ‘rama’ ‘told’
 ‘Rama told that Tulasi stopped using tablets.’

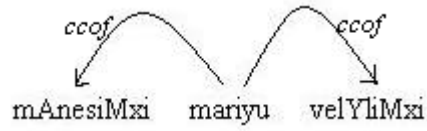


Fig 6. 2nd stage constraint graph for (3)

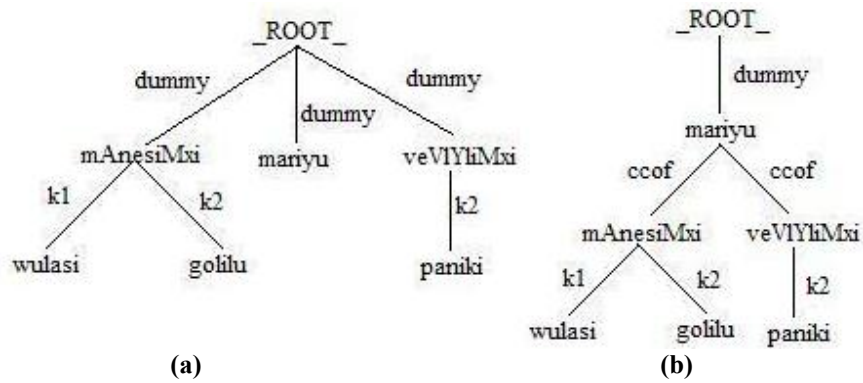


Fig 7. (a) Stage 1 and (b) Stage 2 parse output for (3)

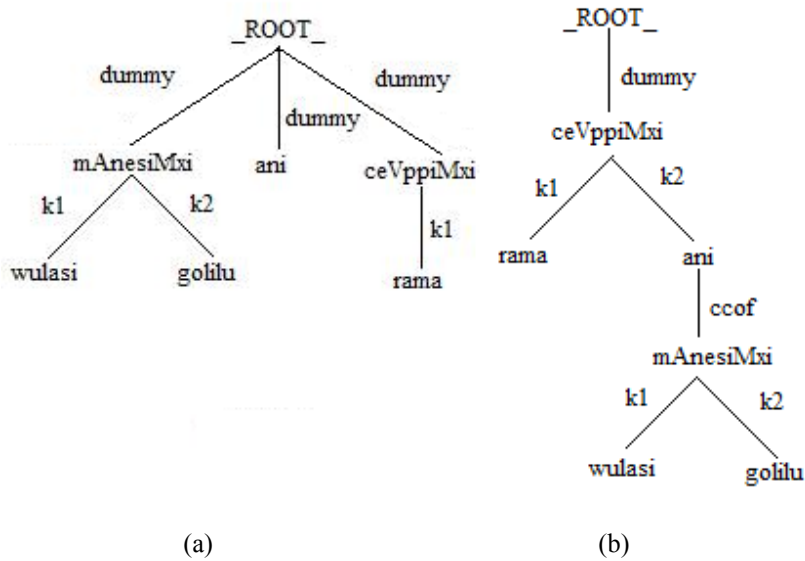


Fig 8. Stage 1 and Stage 2 outputs for sentence (5)

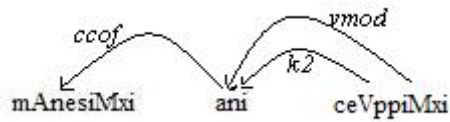


Fig 9. 2nd stage CG for sentence (5)

3.3 H-constraints

As mentioned earlier H-constraints in CBHP mainly comprises of the lexical constraints. This as we saw in section 3.1 corresponds to the basic demand frame and the transformation frame. For Telugu CBHP we manually prepared around 460 verb frames and 95 transformation frames. The transformation frames handles various alternations that are brought in by non-finite and passive TAMs. High frequency verbs and tense, aspect and modality markers were extracted from the training data to prepare the frames. Similarly, other heads such as conjuncts were also extracted. Preparation of these frames took around 30 days.

4 Results and Observations

In this section we describe the data that was used for evaluation. We then give the oracle result of the core parser on this data, following which we discuss the results and the error analysis. The oracle parse for a sentence is the best available parse amongst all the parses of that sentence. It is obtained by selecting the parse closest to the gold parse. The oracle accuracy gives the upper-bound of the parser accuracy and gives some idea about its coverage.

4.1 Data

All the results in this paper are reported for Telugu. We use the Telugu data set that was released as part of the ICON Tools Contest 2010 [8]. The training data had 1300 sentences, development and test set had 150 and 150 sentences respectively. Since the released data is a preliminary version of the treebank it had few errors. Certain relations related to experiencer verbs and verbs of movement have been corrected to report the results. The current parser does not handle ellipses and therefore all the sentences with NULL nodes have been removed to report the results. This data has 1119 training, 133 development and 127 test sentences.

Table 1. Overall parser oracle accuracy

	LAS	UAS	LS
Development	68.06	84.41	70.34
Testing	65.33	84.14	66.60

4.2 Results

Table 1 below shows the oracle accuracies of the parser for the development and testing data. We see that the UAS (unlabeled attachment score) for both test and development is very good; the accuracies for LAS (labeled attachment score) and LS (labeled score) however are not. In Table 2 we show the breakup of the results into intra-clausal and inter-clausal relations. We see that on an average the inter-clausal

relations are being identified successfully, and the low LAS of Table 1 can be attributed mainly to the intra-clausal relations.

Table 2. Intra-clausal and Inter-clausal relation accuracy

		LAS	UAS	LS
Development	Intra-clausal	59.83	82.82	63.15
	Inter-clausal	85.89	87.73	85.89
Test	Intra-clausal	57.92	85.11	59.87
	Inter-clausal	79.63	82.09	79.63

Further analysis of the results showed why the oracle LAS is not very high:

1. *Coverage of H-constraints:* As mentioned earlier we are currently using around 460 frames in the parser. Close to 30% of all the verbs in the test and development data were unseen. The parser uses a default strategy for unseen verbs; not surprisingly, this does not always work well. Similar observation has been reported in the literature for all the parsing approaches in general [22].
2. *Unhandled Relations:* There are still some relations that the parser doesn't handle. Complex predicate is one such case. Automatic identification of such predicates is a challenging task as most diagnostics proposed in the literature are behavioral [39], [32]. There has been some work in automatically identifying complex predicates for Hindi [25], [30]; we need to try and see if these methods can help us too. Ellipses is another phenomena that the parser doesn't handle. In Telugu, sometimes even the main arguments of the verb might go missing and in such cases the parser might assign this relation to some other word with the same property.
3. *Morphological errors and ambiguous TAMs:* A small portion of errors are caused when the morphological analyzer gets the root form of a verb wrong. In such a case, CBHP will pick incorrect verb frame. Also, in Telugu certain TAM (tense, aspect and modality) labels are ambiguous and will affect transformations.

5 S-constraints and Prioritization

It was clear from section 2 and 3 that the core parser that uses H-constraints produces multiple parses. S-constraints are those constraints that are used in a language as preferences and hence can be used to rank the multiple parses. These S-constraints can be used for ranking by penalizing a parse for the constraint that it violates and finally choosing the parse that gets least penalized. This strategy is similar to the one used in Optimality Theory [33], [34]. The other way is to use these S-constraints as

features, associate weight with them, use them to score the output parses and select the parse with the best score. We use the latter strategy. The score of a dependency parse tree $t=(V, A)$ in most graph-based parsing system [26] is

$$\text{Score}(t) = \text{Score}(V, A) \in \mathbf{R} \quad (\text{I})$$

where, V and A are the set of vertices and arcs. This score signifies how likely it is that a particular tree is the correct analysis of a sentence S . Many systems assume the above score to factor through the scores of subgraph of t . Thus, the above score becomes

$$\text{Score}(t) = \sum_{\alpha \in at} \lambda_{\alpha} \quad (\text{II})$$

Where, α is the subgraph, α_t is the relevant set of subgraph in t and λ is a real valued parameter. If one follows the arc-factored model for scoring a dependency tree [26] like we do, the above score become

$$\text{Score}(t) = \sum_{(i, r, j) \in A} \lambda_{(i, r, j)} \quad (\text{III})$$

In (III) the score is parameterized over the arcs of the dependency tree. Since we are interested in using this scoring function for ranking, our ranking function (R) should therefore select the parse that has the maximum score amongst all the parses (Φ) produced by the core parser.

$$R(\Phi, \lambda) = \text{argmax}_{(t=(V,A)) \in T} \text{Score}(t) = \text{argmax}_{(t=(V,A)) \in T} \sum_{(i, r, j) \in A} \lambda_{(i, r, j)} \quad (\text{IV})$$

Since, in our case $\lambda_{(i, r, j)}$ represent probabilities therefore it is more natural to multiply the arc parameters instead of summing them.

$$R(\Phi, \lambda) = \text{argmax}_{(t=(V,A)) \in T} \text{Score}(t) = \text{argmax}_{(t=(V,A)) \in T} \prod_{(i, r, j) \in A} \lambda_{(i, r, j)} \quad (\text{V})$$

For us $\lambda_{(i, r, j)}$ is simply the probability of relation r on arc $i \rightarrow j$ given some S-constraints (Sc). This probability is obtained using the MaxEnt model [40]. So,

$$\lambda_{(i, r, j)} = p(r_{ij} | Sc) \quad (\text{VI})$$

If A denotes the set of all dependency labels and B denotes the set of all S-constraints then MaxEnt ensures that p maximizes the entropy

$$H(p) = - \sum_{x \in E} p(x) \log p(x) \quad (\text{VII})$$

Where $x = (a,b)$, $a \in A$, $b \in B$ and $E = A \times B$. Note that, since we are not parsing but prioritizing, unlike the arc-factored model where the feature function associated with the arc parameter consists only of the features associated with that specific arc, our features can have wider context. Figure 10 shows the context over which various S-constraints can be specified to create the MaxEnt model. Some of the S-constraints

that have been tried out are: (1) *Order of the arguments*, (2) *Relative position of arguments with respect to the verb*, (3) *Agreement*, (4) *General graph properties*.

These S-constraints get reflected as features that are used in MaxEnt. The features for which the model gave the best performance are given below. Note that the actual feature pool was much larger, and some features like that for agreement did not get selected.

- (1) Root, POS tag, Chunk tag, suffix of the current node and its parent
- (2) Suffix of the grandparent, Conjoined suffix of current node and head
- (3) Root, Chunk Tag, Suffix, Morph category of the 1st right sibling
- (4) Suffix, Morph category of the 1st left sibling
- (5) Dependency relations between the first two, right and left sibling and the head
- (6) Dependency relation between the grandparent and head
- (7) Dependency relation between the current node and its child
- (8) A binary feature to signify if a k1 already exist for this head
- (9) A binary feature to signify if a k2 already exist for this head
- (10) Distance from a non-finite head

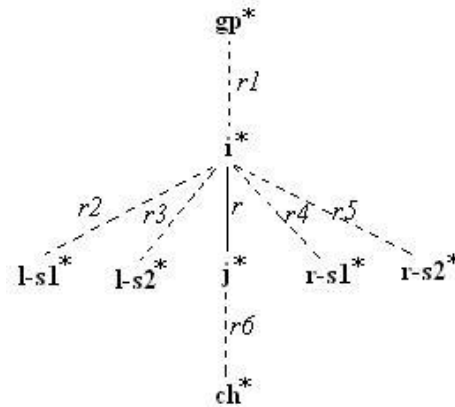


Fig 10. Context over which S-constraints can be specified. Node i is the parent of node j . $l-s1$ corresponds to 1st left sibling, $r-s1$ corresponds to 1st right sibling, gp is grandparent of node j , ch is child of node j . $r1-r6$ are dependency relations

The ranking function shown in (V) can differ based on how one gets the probability of relation on arc $i \rightarrow j$. Since we are ranking labeled dependency tree the first way (as shown in VI) is to use the probability of the label r in the labeled dependency parse. But we can also use the probability of the label given by the MaxEnt model. Considering this, the third obvious way is to take the weighted average the two method. (VIII) and (IX) show these other two options.

$$\lambda_{(i, r, j)} = p(\text{rm}_{i,j} | \text{Sc}) \quad (\text{VIII})$$

where, rm is the relation on arc $i \rightarrow j$ predicted by the model.

$$\lambda_{(i, r, j)} = (p(r_{i,j} | \text{Sc}) + p(\text{rm}_{i,j} | \text{Sc})) / 2 \quad (\text{IX})$$

When the ranker uses (VI) we call it ‘Ranking with Parser Relation probability’ (Rank-PR), the other two are called ‘Ranking with Model Relation probability’ (Rank-MR) and ‘Ranking with Weighted Relation probability’ (Rank-WR).

6 Prioritization Results and observations

Table 3 shows the result of the MaxEnt model⁴ on the development and test data. The features used for training were mentioned in the previous section.

Table 3. Accuracy of the MaxEnt labeler

	Accuracy
Development	76.62
Test	76.78

The result for Rank-PR, Rank-MR, and Rank-WR on both development and testing data is shown in Table 4. It is interesting to note that the best system turns out to be Rank-WR. One should not be surprised with this as this strategy combines the advantage of both the parser labels and the MaxEnt predicted labels. We can see that the best UAS is very close to the oracle UAS. The difference however is wider for LAS.

Table 4. Parser accuracy after Ranking

		LAS	UAS	LA
Development	Rank-PR	59.51	81.56	63.12
	Rank-MR	57.03	82.13	61.03
	Rank-WR	59.51	81.94	63.31
Test	Rank-PR	58.99	82.45	61.10
	Rank-MR	55.18	81.82	57.72
	Rank-WR	59.83	82.45	61.52

The average number of output parses for each sentence is around 10. It was noticed that the differences between these parses were very minimal and this makes ranking them a non-trivial task. The closeness between parses is quite expected from a constraint based parser whose output parses are only those that do not violate any of the H-constraints. In other words most of the output parses are linguistically very sound. Of course, linguistic soundness is only restricted to morpho-syntax and does not consider any semantics. This is because the H-constraints do not incorporate any

⁴<http://maxent.sourceforge.net/>

semantics in the parser as of now. Considering this, the error analysis doesn't throw up any big surprises. The main reasons why the LAS suffers can be attributed to:

- i. *Lack of explicit post-positions or presence of ambiguous one*: Errors because of this, manifest themselves at different places. This can lead to attachment error. Few common cases are finite and non-finite argument sharing, confusion between finite and non-finite argument, adjectival participle, appositions, etc. Also, it was noted that the most frequent errors are for those arguments of the verb, that have no postposition. Consequently, relations such as 'k1', 'k2', 'k7' and 'vmod' have very high confusion. The other major error caused by lack of postposition is the selection of parses with argument ordering errors.
- ii. *Multiple parses with the same score*: It is possible that more than one parse finally gets the same score. This is partly caused due the above reason but it also reflects the accuracy of the labeler. As the accuracy of the labeler increases this problem will lessen. Currently, we select only the first parse amongst all the parses with equal score.

7 Conclusion and Future directions

In this paper we successfully adapted a constraint based hybrid parser for Telugu. We showed that the parser is broad coverage and handles various syntactic phenomena. We motivated the analysis in two stages and showed that a finite clause can be a basis of such a division. The oracle accuracies of the parser on the development and the test data set shows that the parser performs well, however there is lot of room for improvement in LAS. The deficit in LAS, as showed, was due to reasons that can be resolved. Apart from incorporating more H-constraints, handling more constructions, we also plan to try and induce the H-constraints automatically from a treebank. For Hindi and Telugu, this has recently been successfully shown by [21]. Along with the base parser, we also discussed the ranking strategy to get the best parse. We noticed that the best selected parse comes very close to the oracle UAS but lags behind in LAS. The error analysis shows that this is mainly because of lack of any explicit cues in the sentence. One of the things that we plan to do to help improve the final selection is to use an OT style filter [34] to compliment the present ranker. Of course, the ranker also benefits from any improvement in the core parser.

References

1. Ambati, B.R., Husain, S., Nivre, J., Sangal, R.: On the Role of Morphosyntactic Features in Hindi Dependency Parsing. In: NAACL-HLT 2010 workshop on Statistical Parsing of Morphologically Rich Language, Los Angeles, CA. (2010)
2. Begum, R., Husain, S., Dhvaj, A., Sharma, D., Bai, L., Sangal, R.: Dependency annotation scheme for Indian languages. In: IJCNLP08 (2008)

3. Bharati, A., Husain, S., Misra, D., Sangal, R.: Two stage constraint based hybrid approach to free word order language dependency parsing. In: The 11th IWPT09. Paris (2009)
4. Bharati, A., Husain, S., Vijay, M., Deepak, K., Misra, D., Sangal, R.: Constraint Based Hybrid Approach to Parsing Indian Languages. In: The 23rd Pacific Asia Conference on Language, Information and Computation. Hong Kong (2009)
5. Eryigit, G., Nivre, J., Oflazer, K.: Dependency Parsing of Turkish. *Computational Linguistics* 34(3), 357-389 (2008)
6. Goldberg, Y., Elhadad, M.: Hebrew Dependency Parsing: Initial Results. In: The 11th IWPT09. Paris (2009)
7. Hall, J., Nilsson, J., Nivre, J., Eryigit, G., Megyesi, B., Nilsson M., Saers, M.: Single Malt or Blended? A Study in Multilingual Parser Optimization. In: EMNLP-CoNLL shared task (2007)
8. Husain, S., Mannem, P., Ambati, B., Gadde, P.: The ICON-2010 tools contest on Indian language dependency parsing. In: ICON-2010 tools contest on Indian language dependency parsing. Kharagpur, India (2010) *to appear*
9. Husain, S., Gadde, P., Ambati, B., Sharma, D., Sangal, R.: A modular cascaded approach to complete parsing. In: The COLIPS International Conference on Asian Language Processing (IALP) Singapore (2009)
10. McDonald, R., Nivre, J.: Characterizing the Errors of Data-Driven Dependency Parsing Models. In: Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (2007)
11. Nivre, J.: Non-Projective Dependency Parsing in Expected Linear Time. In: ACL-IJCNLP (2009)
12. Seddah, D., Candito, M., Crabbé, B.: Cross parser evaluation : a French Treebanks study. In: The 11th IWPT09. Paris (2009)
13. Tsarfaty, R., Sima'an, K.: Relational-Realizational Parsing. In: The 22nd CoLing. Manchester, UK. (2008)
14. Tsarfaty, R., Seddah, D., Goldberg, Y., Kuebler, S., Versley, Y., Candito, M., Foster, J., Rehbein, I., Tounsi, L.: Statistical Parsing of Morphologically Rich Languages (SPMRL) What, How and Wither. In: NAACL-HLT 2010 workshop on Statistical Parsing of Morphologically Rich Languages (SPMRL 2010), Los Angeles, CA. (2010)
15. Begum, R., Husain, S., Sharma, D., Bai, L.: Developing Verb Frames in Hindi. In: LREC (2008)
16. Collins, M., Koo, T.: Discriminative reranking for natural language parsing. In: CL p.25-70 March05 (2005)
17. Debusmann, R., Duchier, D., Kruijff, G.: Extensible dependency grammar: A new methodology. In: Workshop on Recent Advances in Dependency Grammar, pp. 78-85 (2004)
18. Foth, K. A., Menzel, W.: Hybrid parsing: Using probabilistic models as predictors for a symbolic parser. In: COLING-ACL06 (2006)
19. Goldberg, Y., Elhadad, M.: Hebrew Dependency Parsing: Initial Results. In: the 11th IWPT09. (2009)
20. Karlsson, F., Voutilainen, A., Heikkilä, J., Anttila, A. (eds): *Constraint Grammar: A language-independent system for parsing unrestricted text.* Mouton de Gruyter. (1995)
21. Kolachina, P., Kolachina, S., Singh, A.K., Naidu, V., Husain, S., Sangal, R., Bharati, A.: Grammar Extraction from Treebanks for Hindi and Telugu. (2009)
22. Manning, C.D., Schütze, H.: *Foundations of statistical natural language processing.* MIT Press, pp. 272, 299 (2002)
23. Martins, A., Smith, N., Xing, E.: Concise Integer Linear Programming Formulations for Dependency Parsing. In: the ACL-IJCNLP09 (2009)
24. Maruyama, H.: Structural disambiguation with constraint propagation. In: ACL:90 (1990)

25. Mukerjee, A., Soni, A., Raina, A.M.: Detecting Complex Predicates in Hindi using POS Projection across Parallel Corpora. In: the COLING-ACL Workshop on Multiword Expressions: Identifying and Exploiting Underlying Properties, Sydney. (2006)
26. Kubler, S., McDonald, R., Nivre, J.: Dependency parsing. Morgan and Claypool. (2009)
28. Schröder, I.: Natural Language Parsing with Graded Constraints. PhD thesis, Hamburg Univ (2002)
29. Shen, L., Sarkar, A., Joshi, A.K.: Using LTAG Based Features in Parse Reranking. In: EMNLP (2003)
30. Sinha, R. M. K.: Mining Complex Predicates In Hindi Using A Parallel Hindi-English Corpus. In: MWE09, ACL-IJCNLP (2009).
31. Tapanainen, P., Järvinen, T.: A non-projective dependency parser. In: the 5th Conference on Applied Natural Language Processing, pp. 64–71. (1997)
32. Verma, M. K. (ed.): Complex Predicates in South Asian Languages. Manohar Publications. New Delhi (1993)
33. Prince, A., Smolensky, P.: Optimality Theory: constraint interaction in generative grammar. In: Technical Report, Rutgers Center for Cognitive Science. (1993)
34. Aissen, J.: Markedness and subject choice in Optimality Theory. *Natural Language and Linguistic Theory* 17:673–711. (1999)
35. Bharati, A., Husain, S, Sharma, D.M., Sangal, R.: A Two-Stage Constraint Based Dependency Parser for Free Word Order Languages. In: the COLIPS IALP. (2008)
36. Bharati, A., Sangal, R., Reddy, T.P.: A Constraint Based Parser Using Integer Programming. In: ICON. (2002)
37. Bharati, A. Sangal, R.: Parsing Free Word Order Languages in the Paninian Framework. In: ACL: 93 (1993)
38. Bharati, A., Sharma, D. M., Bai, L, Sangal, R.: AnnCorra: Annotating Corpora Guidelines for POS and Chunk Annotation for Indian Languages. LTRC-TR31. (2006)
39. Butt. M.: The Structure of Complex Predicates in Urdu. CSLI Publications (1995)
40. Ratnaparkhi, A.: Maximum entropy models for natural language ambiguity resolution. Ph.D. Dissertation, University of Pennsylvania. IRCS Tech Report IRCS-98-15. (1998)