

# A Two Stage 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, meher\_vijay}@students.iiit.ac.in,

samar@research.iiit.ac.in

## Abstract

In this paper we present a two stage constraint Telugu dependency parsing. We define the two stages 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 give 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.

## 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 (Tsarfaty et al., 2010; Ambati et al., 2010; McDonald and Nivre, 2007; Nivre, 2009; Tsarfaty and Sima'an, 2008; Seddah et al., 2009; Goldberg and Elhadad, 2009; Husain et al., 2009; Eryigit et al., 2008). Data driven parser performance (Hall et al., 2007) 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 (Karlsson et al., 1995; Maruyama, 1990; Bharati et al., 2002, 1993; Tapanai-

nen, Järvinen, 1997; Schröder, 2002; Martins et al., 2009; and Debusmann et al., 2004).

In this paper we present a two stage constraint based hybrid approach to Telugu dependency parsing based on the parser proposed by Bharati et al., (2008, 2009a, 2009b). We begin 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 give 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.

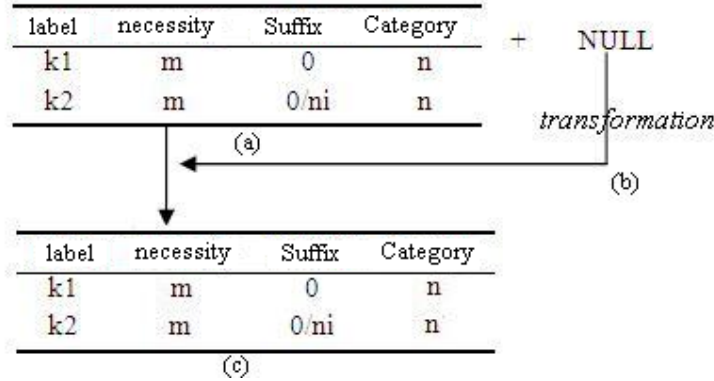
This paper is arranged as follows: section 2 explains in details how we handle different Telugu constructs in two stages using CBHP, section 3 describes the results of the core parser and makes some observations on these results. In section 4 we describe the ranking strategies, followed with results in Section 5. We conclude the paper with future directions in section 6.

## 2 CBHP for Telugu

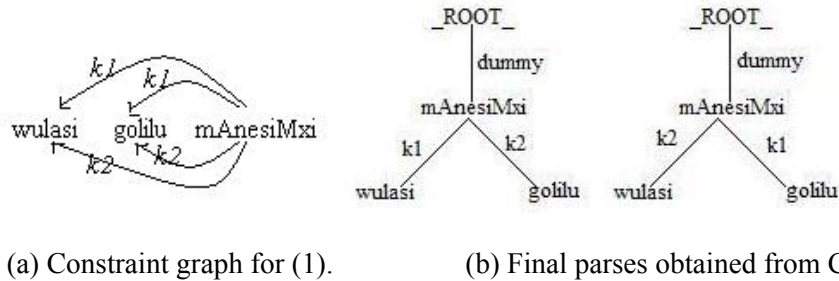
Bharati et al., (2009a, 2009b) have proposed a constraint based hybrid parser (CBHP) for Indian languages. It has however been tested only for Hindi. We use the same machinery that was used by them to handle Telugu. We first describe the different types of relations the parser currently handles in the two stages, following which we briefly mention H-constraints for Telugu CBHP.

### 2.1 Relations handled in Stage 1

Stage 1 handles mainly intra-clausal relations. These relations represent:



**Figure 1.** 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.



**Figure 2.**

- 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  $k1^1$  and  $k2$  as mandatory children with null postpositions. Figure 1 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 2(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 bi-partite graph matching (Bharati et al., 1993, 2002). Figure 2(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 un-

<sup>1</sup> $k1$  can be roughly translated to ‘agent’,  $k2$  can be roughly translated as ‘theme’. CBHP follows the syntactic-semantic dependency labels proposed in (Begum et al., 2008).

processed nodes to it. `_ROOT_` ensures that the output we get after each stage is a tree.

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

## 2.2 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

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

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

Figure 3 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 4 shows the 2<sup>nd</sup> stage CG for (3). Notice that only the two finite verbs and the conjunct are seen here. The relations identified in the 1<sup>st</sup> stage remain untouched. Figure 5 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 1<sup>st</sup> stage and 2<sup>nd</sup> stage is not problematic as the two stages are mutually exclusive.

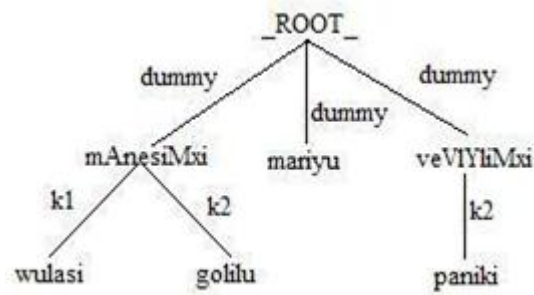


Figure 3. 1<sup>st</sup> stage parse output of (3)



Figure 4. 2<sup>nd</sup> stage constraint graph for (3)

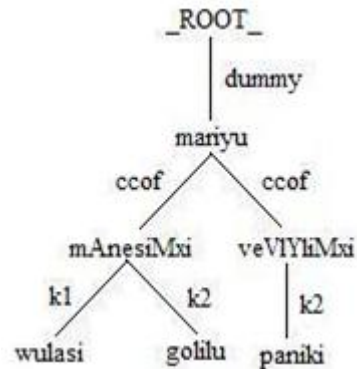


Figure 5. Final parse output of (3)

We must note here that for few cases such as (4) below, the 1<sup>st</sup> stage output is revised. We follow the same principles as described in (Bharati et al., 2008). This mainly concerns case dropping in nominal coordinations.

- (4) *wulasi mariyu rama golilu*  
 ‘Tulasi ‘and’ ‘Rama’ ‘tablets’  
*mAnesAru.*  
 ‘stopped using’  
 ‘Tulasi and Rama stopped using tablets.’

### 2.3 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.

## 3 Results

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.

### 3.1 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 (Husain et al., 2010). The training data had 1300 sentences, development and test set had 150 and 150 sentences respectively. 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.

### 3.2 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 accu-

racies for LAS (labeled attachment 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.

	LAS	UAS	LS
<b>Development</b>	68.06	84.41	70.34
<b>Testing</b>	65.33	84.14	66.60

**Table 1.** Overall parser oracle accuracy

		LAS	UAS	LS
<b>Devel</b>	<b>Intra</b>	59.83	82.82	63.15
	<b>Inter</b>	85.89	87.73	85.89
<b>Test</b>	<b>Intra</b>	57.92	85.11	59.87
	<b>Inter</b>	79.63	82.09	79.63

**Table 2.** Intra-clausal (Intra) and Inter-clausal (Inter) relation accuracy

## 4 S-constraints and Prioritization

It was clear from section 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 Weighted Constraint Grammar (Schröder, 2002). 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. This is similar to the work on ranking in phrase structure parsing (Coolins and Koo, 2005; Shen et al., 2003) and graph-based parsing (McDonald et al., 2005). We use the latter strategy. The score of a dependency parse tree  $t=(V, A)$  in most graph-based parsing system (Kubler et al., 2009) 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,  $\alpha$  is the subgraph,  $\alpha_t$  is the relevant set of subgraph in  $t$  and  $\lambda$  is a real valued parameter. If one follows the arc-factored model for scoring a dependency tree (Kubler et al., 2009) 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 ( $\Phi$ ) produced by the core parser.

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

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

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

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 (Ratnaparkhi, 1998). So,

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

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. 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 1<sup>st</sup> right sibling
- (4) Suffix, Morph category of the 1<sup>st</sup> 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

## 5 Prioritization Results and observations

Table 3 shows the result of the MaxEnt model<sup>2</sup> on the development and test data. The features used for training were mentioned in the previous section.

	Accuracy
<b>Development</b>	76.62
<b>Test</b>	76.78

**Table 3.** Accuracy of the MaxEnt labeler

The ranked parse score (Rank-P) for both development and testing data is shown in Table 4. We can see that the best UAS is very close to the oracle UAS. The difference however is wider for LAS.

<sup>2</sup><http://maxent.sourceforge.net/>

		LAS	UAS	LA
Devel.	Rank-P	59.51	81.56	63.12
Test	Rank-P	58.99	82.45	61.10

**Table 4.** Parser accuracy after Ranking

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 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 post-position. 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.

Our system's official result is much less than the results in Table 4. As noted earlier this is primarily because the parser does not handle ellipses and in those sentences fails to identify the correct heads.

	LAS	UAS	LA
Test	48.08	76.29	50.25

**Table 5.** Official Score

## 6 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. 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 (Kolachina et al., 2009). 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.

## References

- B. Ambati, S. Husain, J. Nivre and R. Sangal. 2010. On the Role of Morphosyntactic Features in Hindi Dependency Parsing. In Proc of *NAACL-HLT 2010 workshop on Statistical Parsing of Morphologically Rich Language, Los Angeles, CA*.
- R. Begum, S. Husain, A. Dhawaj, D. Sharma, L. Bai and R. Sangal. 2008. Dependency annotation scheme for Indian languages. In Proc of *IJCNLP08*
- A. Bharati, S. Husain, D. Misra and R. Sangal. 2009a. Two stage constraint based hybrid approach to free word order language dependency parsing. In Proc of *the 11th IWPT09, Paris*
- A. Bharati, S. Husain, M. Vijay, K. Deepak, D. Misra and R. Sangal. 2009b. Constraint Based Hybrid Approach to Parsing Indian Languages. In Proc of *The 23rd Pacific Asia Conference on Language, Information and Computation, Hong Kong*
- A. Bharati, S. Husain, D. Sharma and R. Sangal. 2008. A Two-Stage Constraint Based Dependency Parser for Free Word Order Languages. In Proc of *the COLIPS IALP*.

- A. Bharati, D. Sharma, L. Bai and R. Sangal. 2006. AnnCorra: Annotating Corpora Guidelines for POS and Chunk Annotation for Indian Languages. *LTRC-Technical Report TR31*.
- A. Bharati and R. Sangal, T.P. Reddy. 2002. A Constraint Based Parser Using Integer Programming, In Proc of *ICON*.
- A. Bharati and R. Sangal. 1993. Parsing Free Word Order Languages in the Paninian Framework. In Proc of *ACL: 93*
- M. Collins, and T. Koo. 2005. Discriminative reranking for natural language parsing. In *CL p.25-70 March05*.
- R. Debusmann, D. Duchier and G. Kruijff. 2004. Extensible dependency grammar: A new methodology. In Proc of *Workshop on Recent Advances in Dependency Grammar*, pp. 78–85.
- G. Eryigit, J. Nivre and K. Oflazer. 2008. Dependency Parsing of Turkish. *Computational Linguistics 34(3)*, 357-389.
- Y. Goldberg and M. Elhadad. 2009. Hebrew Dependency Parsing: Initial Results. In Proc of *the 11th IWPT09*.
- J. Hall, J. Nilsson, J. Nivre, G. Eryigit, B. Megyesi, M. Nilsson and M. Saers. 2007. Single Malt or Blended? A Study in Multilingual Parser Optimization. In Proc of *EMNLP-CoNLL shared task*
- S. Husain, P. Mannem, B. Ambati and P. Gadde. 2010. The ICON-2010 tools contest on Indian language dependency parsing. In Proc of *ICON-2010 tools contest on Indian language dependency parsing*. Kharagpur, India to appear
- S. Husain, P. Gadde, B. Ambati, D. Sharma and R. Sangal. 2009. A modular cascaded approach to complete parsing. In Proc of *The COLIPS International Conference on Asian Language Processing (IALP) Singapore*
- F. Karlsson, A. Voutilainen, J. Heikkilä and A. Anttila. (eds). 1995. *Constraint Grammar: A language-independent system for parsing unrestricted text*. Mouton de Gruyter.
- P. Kolachina, S. Kolachina, A.K. Singh, V. Naidu, S. Husain, R. Sangal and A. Bharati. 2009. Grammar Extraction from Treebanks for Hindi and Telugu. In Proceedings of *The 7th LREC. Valleta. Malta*.
- S. Kubler, R. McDonald and J. Nivre. 2009. *Dependency parsing*. Morgan and Claypool.
- A. Martins, N. Smith and E. Xing. 2009. Concise Integer Linear Programming Formulations for Dependency Parsing. In Proc of the *ACL-IJCNLP09*.
- H. Maruyama. 1990. Structural disambiguation with constraint propagation. In Proc of *ACL:90*
- R. McDonald, F. Pereira, K. Ribarov, and J. Hajic. 2005. 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 EMNLP and CoNLL*.
- J. Nivre. 2009. Non-Projective Dependency Parsing in Expected Linear Time. In Proc of *ACL-IJCNLP*.
- A. Ratnaparkhi. 1998. *Maximum entropy models for natural language ambiguity resolution*. Ph.D. Dissertation, University of Pennsylvania. IRCS Tech Report IRCS-98-15.
- I. Schröder. 2002 *Natural Language Parsing with Graded Constraints*. PhD thesis, Hamburg Univ
- D. Seddah, M. Candito and B. Crabbé. 2009. Cross parser evaluation : a French Treebanks study. In proc of *the 11th IWPT09*. Paris
- L. Shen, A. Sarkar and A.K. Joshi. 2003. Using LTAG Based Features in Parse Reranking. In Proc of *EMNLP*.
- P. Tapanainen and T. Järvinen. 1997. A non-projective dependency parser. In Proc of *the 5th Conference on Applied Natural Language Processing*, pp. 64–71.
- R. Tsarfaty, D. Seddah, Y. Goldberg, S. Kuebler, Y. Versley, M. Candito, J. Foster, I. Rehbein and L. Tounsi. 2010. Statistical Parsing of Morphologically Rich Languages (SPMRL) What, How and Wither. In Proc of *NAACL-HLT 2010 workshop on SPMRL, Los Angeles, CA*.
- R. Tsarfaty and K. Sima'an. 2008. Relational-Realizational Parsing. In Proc of *the 22nd CoLing*