

Empty Categories in Hindi Dependency Treebank: Analysis and Recovery

Chaitanya GSK Intl Institute of Info. Technology Hyderabad, India chaitanya.gsk @research.iiit.ac.in	Samar Husain Intl Institute of Info. Technology Hyderabad, India samar @research.iiit.ac.in	Prashanth Mannem Intl Institute of Info. Technology Hyderabad, India prashanth @research.iiit.ac.in
---	--	--

Abstract

In this paper, we first analyze and classify the empty categories in a Hindi dependency treebank and then identify various discovery procedures to automatically detect the existence of these categories in a sentence. For this we make use of lexical knowledge along with the parsed output from a constraint based parser. Through this work we show that it is possible to successfully discover certain types of empty categories while some other types are more difficult to identify. This work leads to the state-of-the-art system for automatic insertion of empty categories in the Hindi sentence.

1 Introduction

Empty categories play a crucial role in the annotation framework of the Hindi dependency treebank¹ (Begum et al., 2008; Bharati et al., 2009b). They are inserted in a sentence in case the dependency analysis does not lead to a fully connected tree. In the Hindi treebank, an empty category (denoted by a NULL node) always has at least one child. These elements have essentially the same properties (e.g. case-marking, agreement, etc.) as an overtly realized element and they provide valuable information (such as predicate-argument structure, etc.). A different kind of motivation for postulating empty categories comes from the demands of natural language processing, in particular parsing. There are several types of empty categories in the Hindi dependency

treebank serving different purposes. The presence of these elements can be crucial for correct automatic parsing. Traditional parsing algorithms do not insert empty categories and require them to be part of the input. The performance of such parser will be severely affected if one removes these elements from the input data. Statistical parsers like MaltParser (Nivre, 2003), MSTParser (McDonald, 2005), as well as Constraint Based Hybrid Parser (CBHP) (Bharati et al., 2009a) produce incorrect parse trees once the empty categories are removed from the input data. Hence there is a need for automatic detection and insertion of empty categories in the Hindi data. Additionally, it is evident that successful detection of such nodes will help the annotation process as well.

There have been many approaches for the recovery of empty categories in the treebanks like Penn treebank, both ML based (Collins, 1997; Johnson, 2002; Dienes and Dubey, 2003a,b; Higgins, 2003) and rule based (R Campbell, 2004). Some approaches such as Yang and Xue (2010) follow a post processing step of recovering empty categories after parsing the text.

In this paper we make use of lexical knowledge along with the parsed output from a constraint based parser to successfully insert empty category in the input sentence, which may further be given for parsing or other applications. Throughout this paper, we use the term recovery (of empty categories) for the insertion of different types of empty categories into the input sentence.

The paper is arranged as follows, Section 2 discusses the empty nodes in the treebank and classifies

¹The dependency treebank is part of a Multi Representational and Multi-Layered Treebank for Hindi/Urdu (Palmer et al., 2009).

NULL_NP tokens	69
NULL_VG tokens	68
NULL_CCP tokens	32
Sentences with more than one empty category in them	159

Table 1: Empty categories in Hindi Tree bank

them based on their syntactic type. In section 3 we provide an algorithm to automatically recover these elements. Section 4 shows the performance of our system and discusses the results. We conclude the paper in section 5.

2 An overview of Empty Categories in Hindi dependency Treebank

Begum et al., (2008) proposed a dependency framework in which an empty node is introduced during the annotation process only if its presence is required to build the dependency tree for the sentence (Figures 1, 2, 3)². Empty categories such as those discussed in Bhatia et al. (2010) which would be leaf nodes in the dependency tree are not part of the dependency structure and are added during Propbanking³. Consequently, the empty categories in Hindi treebank do not mark displacement as in Penn treebank (Marcus et al., 1993) rather, they represent undisplaced syntactic elements which happen to lack phonological realization. In the Hindi dependency treebank, an empty category is represented by a ‘NULL’ word. Sentences can have a missing VG or NP or CCP⁴. These are represented by ‘NULL’ token and are marked with the appropriate Part-of-speech tag along with marking the chunk tag such as NULL_NP, NULL_VGF, NULL_CCP, etc. in Table 2

²Due to space constraints, sentences in all the figures only show chunk heads. Please refer to examples 1 to 6 for entire sentences with glosses

³These empty categories are either required to correctly capture the argument structure during propbanking or are required to successfully convert the dependency structure to phrase structure (Xia et al., 2009)

⁴VG is Verb Group, NP is Noun Phrase and CCP is Conjoint Phrase.

Type of empty categories	Inst-ances	Chunk tag (CPOS)
Empty subject	69	NULL_NP
Backward gapping	29	NULL_VG
Forward gapping	21	NULL_VG
Finite verb ellipses	18	NULL_VG
Conjunction ellipses (verbs)	20	NULL_CCP
Conjunction ellipses (nouns)	12	NULL_CCP
Total	169	

Table 2: Empty category types.

2.1 Empty category types

From the empty categories recovery point of view, we have divided the empty categories in the treebank into six types (Table 2).

The first type of empty category is *Empty Subject* (Figure 1), *example.1* where a clause ‘*rava ke kaaran hi manmohan singh rajaneeti me aaye*’ is dependent on the missing subject of the verb ‘*hai*’ (is).

- (1) NULL gaurtalab hai ki raao
 NULL ‘noticeable’ ‘is’ ‘that’ ‘Rao’
 ke kaaran hi manmohan singh
 ‘because’ ‘only’ ‘Manmohan’ ‘singh’
 raajaniiti me aaye
 ‘politics’ ‘in’ ‘came.’

‘it is noticeable that because of Rao, Manmohan Singh came in politics’

The second type of empty category is due to *Backward Gapping* (Figure 2), *example.2* where the verb is absent in the clause that occurs before a co-ordinating conjunct.

- (2) doosare nambara para misa roosa
 ‘second’ ‘position’ ‘on’ ‘miss’ ‘Russia’
 natasha NULL aur tiisare nambara
 ‘Natasha’ NULL ‘and’ ‘third’ ‘position’
 para misa lebanan sendra rahiim .
 ‘on’ ‘miss’ ‘Lebanan’ ‘Sandra’ ‘were’ .

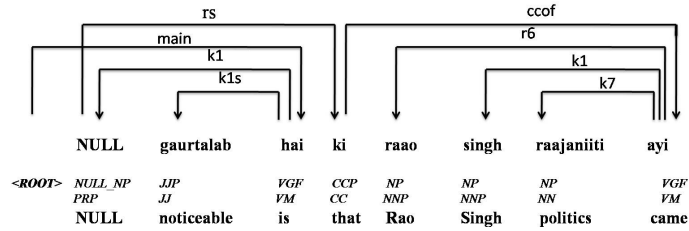


Figure 1: Empty Subject.

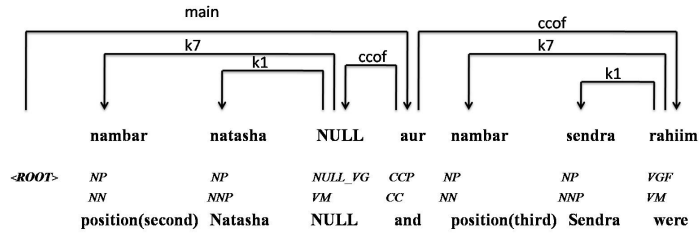


Figure 2: Backward Gapping.

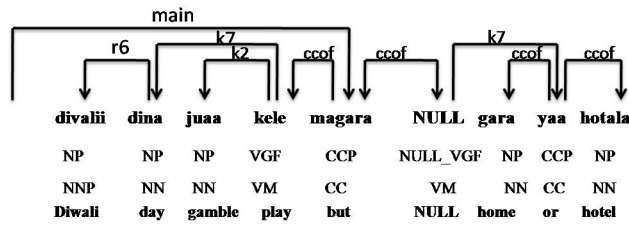


Figure 3: Forward Gapping.

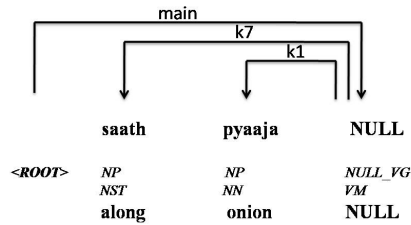


Figure 4: Finite verb ellipses.

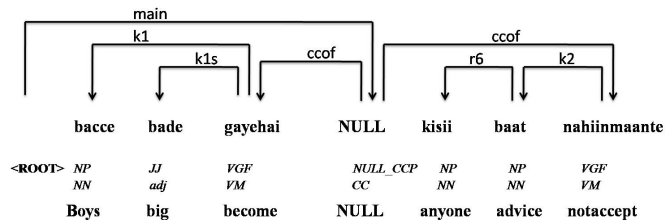


Figure 5: Conjunction ellipses (verbs).

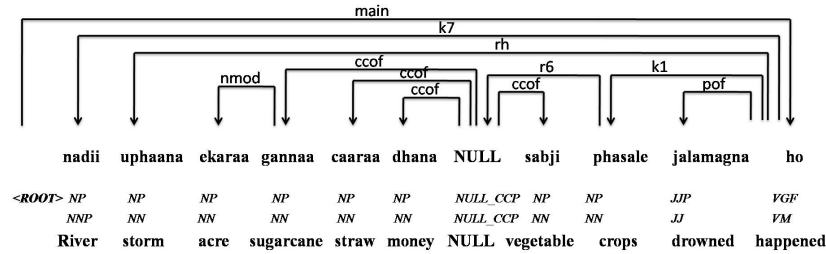


Figure 6: Conjunction ellipses (nouns).

‘Miss Russia stood second and Miss Lebanan was third’

The third type of empty category is *Forward Gapping* (Figure 3), example 3, which is similar to the second type but with the clause with the missing verb occurring after the conjunct rather than before. The reason for a separate class for forward gapping is explained in the next section.

(3) divaalii ke dina jua Kele magara
 ‘Diwali’ ‘GEN’ ‘day’ ‘gamble’ ‘play’ ‘but’
 NULL gar me yaa hotala me
 ‘NULL’ ‘home’ ‘in’ ‘or’ ‘hotel’ ‘in’

‘Played gamble on Diwali day but was it at home or hotel’

The fourth type of empty category is due to *Finite verb ellipses* (Figure 4), example 4, where the main verb for a sentence is missing.

(4) saath me vahii phevareta khadaa pyaaja
 ‘along’ ‘in’ ‘that’ ‘favorite’ ‘raw’ ‘onion’
 NULL.
 NULL

‘Along with this, the same favorite semi-cooked onion’

The fifth type of empty category is *Conjunction ellipses (Verbs)*, example 5 (Figure 5).

(5) bacce bare ho-ga-ye-hai NULL
 ‘children’ ‘big’ ‘become’ ‘NULL’
 kisii ki baat nahiin maante
 ‘anyone’ ‘gen’ ‘advice’ ‘not’ ‘accept’

‘The children have grown big (and) do not listen to anyone’

The sixth type of empty category is the *Conjunction ellipses (for nouns)*, example 6 (Figure 6).

(6) yamunaa nadii me uphaana se
 ‘Yamuna’ ‘river’ ‘in’ ‘storm’ ‘INST’
 sekado ekara gannaa, caaraa,
 ‘thousands’ ‘acre’ ‘sugarcane’ ‘straw’
 dhana, NULL sabjii kii phasale
 ‘money’ ‘NULL’ ‘vegetable’ ‘GEN’ ‘crops’
 jala-magna ho-gai-hai .
 ‘drowned’ ‘happened’

‘Because of the storm in the Yamuna river, thousand acres of sugarcane, straw, money, vegetable crops got submerged’

3 Empty categories recovery Algorithm

Given the limited amount of data available (only 159 sentences with at least one empty category in them out of 2973 sentences in the Hindi treebank, Table 12), we follow a rule based approach rather than using ML to recover the empty categories discussed in the previous section. Interestingly, a rule-based approach was followed by R Campbell, (2004) that recovered empty categories in English resulting in better performance than previous empirical approaches. This work can be extended for ML once more data becomes available.

The techniques that are used for recovering empty categories in the Penn treebank (Collins, 1997; Johnson, 2002;) might not be suitable since the Penn treebank has all the empty categories as leaf nodes in the tree unlike the Hindi dependency treebank where

```

for each sentence in the input data
  try in Empty Subject
  try in Forward Gapping
  try in Finite Verb ellipses
for each tree in CBHP parse output
  try in Backward Gapping
  try in Forward Gapping
  try in Finite Verb ellipses
  try in Conjunction ellipses (for Verbs)

```

Table 3: Empty categories Recovery Algorithm.

the empty categories are always internal nodes in the dependency trees (Figure 2).

In this section we describe an algorithm which recovers empty categories given an input sentence. Our method makes use of both the lexical cues as well as the output of the Constraint Based Hybrid Parser (CBHP). Table 3 presents the recovery algorithm which first runs on the input sentence and then on the output of the CBHP.

3.1 Empty Subject

Framing rule 1 requires the formation of a set (*Cue-Set*) based on our analysis discussed in the previous section. It contains all the linguistic cues (lexical items such as *gaurtalab* ‘noticeable’, *maloom* ‘known’, etc). We then scan the input sentence searching for the cue and insert an empty category (NULL_NP)⁵ if the cue is found. Table 4 illustrates the process where we search for ‘CueSet *he ki*’ or ‘CueSet *ho ki*’ phrases. In Table 4, W+1 represents word next to W, W+2 represents word next to W+1.

3.2 Backward Gapping

To handle backward gapping cases, we take the intermediate parse output from CBHP⁶ for the whole data. The reason behind choosing CBHP lies in its rule based approach. CBHP fails (or rather gives a visibly different parse) for sentences with missing verbs. And when it fails to find a verb, CBHP

⁵We insert a token ‘NULL’ with NULL_NP as CPOS

⁶CBHP is a two-stage parser. In the 1st stage it parses intra-clausal relations and inter-clausal relations in the 2nd stage. The 1st stage parse is an intermediate parse.

```

for each word W in the Sentence
if W  $\in$  CueSet
  if W+1 & W+2 = he or ho & ki
    Insert NULL with PRP as POS,
    NULL_NP as CPOS

```

Table 4: Rule for identifying Empty Subject.

```

for each node N in tree T
  if head of N =  $\phi$ 
    insert N in unattached_subtrees[]
  for each node X in unattached_subtrees[]
    while POS(X) is not VG
      traverse in the array of unattached_subtrees
      if  $\exists$  a conjunct, then recovery=1
    if recovery = 1
      insert NULL, with VM as POS,
      NULL_VG as CPOS
      Head of NULL =  $\phi$ 

```

Table 5: Rule for identifying Backward Gapping using CBHP.

gives unattached subtrees⁷ (Figure 7, 8, 9 illustrates the unattached subtrees where the parser is unable to find a relation between the heads of each unattached subtree). Similarly whenever the parser expects a conjunction and the conjunction is absent in the sentence, CBHP again gives the unattached subtrees.

We analyze these unattached sub-trees to see whether there is a possibility for empty category. The array, in Table 5 represents all the nodes having no heads. POS represents part of speech and CPOS represents chunk part of speech and ϕ represents empty set.

3.3 Forward gapping

The main reason for handling the forward gapping as a separate case rather than considering it along with backward gapping is the prototypical SOV word-order of Hindi, i.e. the verb occurs after subject and object in a clause or sentence. We take the intermediate parse output from the CBHP for the whole data and when ever a verb is absent in a clause occurring immediately after a conjunct, we search for a VG af-

⁷CBHP gives fully connected trees in both the stages. We have modified the parser so that it gives unattached subtrees when it fails.

```

for each node N in tree T
  if head of N =  $\phi$ 
    insert N in unattached_subtrees[]
  for each node X in unattached_subtrees[]
    if  $\exists$  a verb between two conjuncts
      if those conjuncts belongs to conjunct_set
        insert insert NULL with VM as POS,
        NULL_VG as CPOS

```

Table 6: Rule for identifying Forward Gapping using CBHP.

```

for each word W in the sentence S
  if  $W \in \text{CueSet\_FG}$ 
    insert NULL with NULL_VG as POS
    and CPOS
  if W = Conjunct
    if POS(W-1) = VG
      if  $\exists$  a VG in S-W
        insert NULL with VM as POS,
        NULL_VG as CPOS

```

Table 7: Rule for identifying Forward Gapping .

ter the conjunct and insert an empty category if the VG is absent (an example of such cases can be seen in Figure 7). This procedure is given in Table 6. In addition, we use the lexical cues (such as *ya nahii* ‘or not’, *ya* ‘or’) for recovering certain types of empty categories. *CueSet_FG* is the set that contains the lexical cues and *conjunct_set* contains lexical cues like (*ki* and *ya*). This procedure is shown in Table 7.

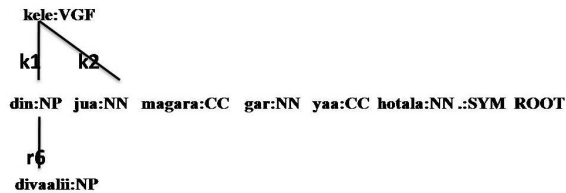


Figure 7: Unattached sub trees in CBHP parse output of an input sentence (forward gapping).

3.4 Finite Verb ellipses

In the cases where there is no VG at all in the sentence, we insert a NULL VG before the EOS (End-Of-Sentence) in the input sentence. For this case, finite verb ellipses can be recovered directly from

```

if  $\exists$  a VG in S-W
  insert NULL with VM as POS,
  NULL_VG as CPOS

```

Table 8: Rule for identifying Finite Verb ellipses in sentence.

```

for each node N in tree T
  if head of N =  $\phi$ 
    insert N in unattached_subtrees[]
  if  $\exists$  a verb in unattached_subtrees[]
    if those conjuncts belongs to conjunct_set
      insert insert NULL with VM as POS,
      NULL_VG as CPOS

```

Table 9: Rule for identifying Finite Verb ellipses using CBHP.

the input sentence using the rule in Table 8 .Also, in a sentence with a VG, we use CBHP to ascertain if this VG is the root of the sentence. If its not, we insert an additional NULL_VG. This algorithm will correctly recover VG in the sentence but the position can be different from the gold input at times not because the recovery algorithm is wrong, but there is no strict rule that says the exact position of empty category in this case of finite verb ellipse and annotators might choose to insert an empty category at any position. For example, in Figure 8, we can insert an empty category either after first NP sub tree or second or the third etc, all these possibilities are accepted syntactically. For simplicity purposes, we insert the empty category just before the EOS. This procedure is shown in Table 9.

3.5 Conjunction ellipses (for verbs)

We again use the intermediate parsed output of CBHP for this type. Whenever there is a missing conjunction between the two finite clauses, the clausal sub trees are disconnected from each other as shown in Figure 9. Hence the rule that should be applied is to insert a NULL_CCP between two sub trees with VG heads and insert NULL CCP immediately after the first verb in the input sentence. Table 10 shows this procedure.

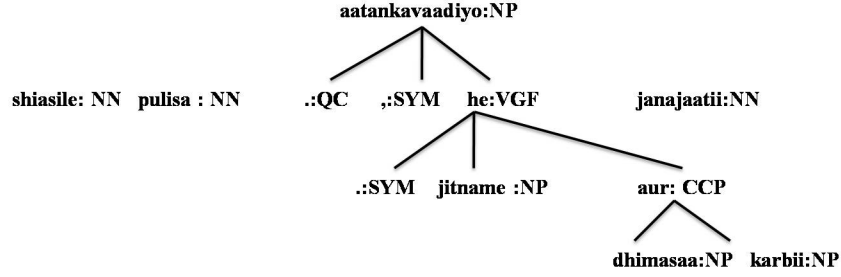


Figure 8: Unattached Subtrees (Finite verb ellipses).

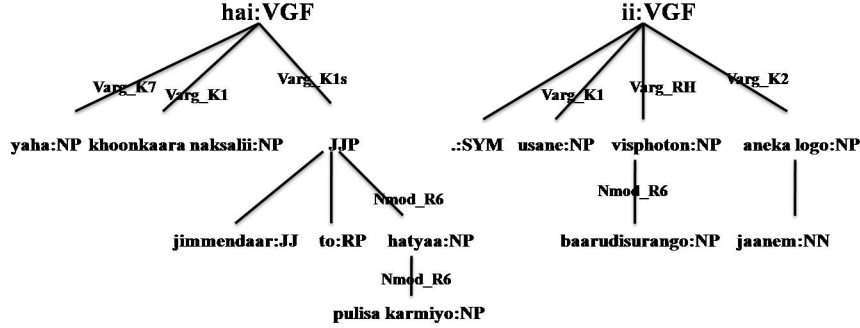


Figure 9: Unattached Subtrees in the case of conjunction ellipses.

```

for each node N in tree T
  if head of N =  $\phi$ 
    insert N in unattached_subtrees[]
  for each node X in unattached_subtrees[]
    if X and X+1 are VG's
      insert insert NULL with CC as POS,
      NULL_CCP as CPOS

```

Table 10: Rule for identifying Finite Verb ellipses using CBHP.

4 Results and Discussion

We have presented two sets of results, the overall empty categories detection along with the accuracies of individual types of empty categories in Table 11 and Table 12.

The results in Table 12 show that the precision in recovering many empty categories is close to 90%. A high precision value of 89.8 for recovery of Empty subject type is due to the strong lexical cues that were found during our analysis. CBHP parse output proved helpful in most of the remaining types. Few cases such as backward gapping and conjunc-

Type of empty categories	Inst-ances	Prec-ision	Recall
Empty subject	69	89.8	89.8
Backward gapping	29	77.7	48.3
Forward gapping	21	88.8	72.7
Finite verb ellipses	18	78.5	61.1
Conjunction ellipses (verbs)	20	88.2	75
Conjunction ellipses (nouns)	12	0	0
Total	169	91.4	69.8

Table 11: Recovery of empty categories in Hindi tree-bank.

tion ellipses (for nouns) are very difficult to handle. We see that although CBHP helps in the recovery process by providing unattached subtrees in many instances, there are cases such as those of backward gapping and nominal conjunction ellipses where it does not help. It is not difficult to see why this is so. The presence of the 2nd verb in the case of backward gapping fools CBHP into treating it as the main verb of a normal finite clause. In such a case, the

Type of empty categories	Inst-ances	Prec-ision	Recall
NULL_NP tokens	69	89.8	89.8
NULL_VG tokens	68	82	60.2
NULL_CCP tokens	32	88.2	46.8
Total	159	91.4	69.8

Table 12: Empty categories in Hindi Tree bank

parser ends up producing a fully formed tree (which of course is a wrong analysis) that is of no use for us.

Similar problem is faced while handling conjunction ellipses (for nouns). Here as in the previous case, CBHP is fooled into treating two coordinating nominals as independent nouns. We note here that both the cases are in fact notoriously difficult to automatically detect because of the presence (or absence) of any robust linguistic pattern.

These results show that our system can be used to supplement the annotators effort during treebanking. We plan to use our system during the ongoing Hindi treebanking to ascertain its effect. As mentioned earlier, automatic detection of empty categories/nodes will prove to be indispensable for parsing a sentence. We also intend to see the effect of our system during the task of parsing.

5 Conclusion

In this paper we presented an empty category recovery algorithm by analyzing the empty categories in the Hindi treebank. This, we noticed, uses lexical cues and parsed output of a constraint based parser. The results show that our system performs considerably high (90%) for many types of empty categories. Few types, on the other hand, such as backward gapping and nominal coordinating conjunctions were very difficult to handle. Our approach and analysis will be useful in automatic insertion of empty nodes during dependency annotation. It will also benefit data-driven/statistical approaches either as a post-processing tool or in recovering empty categories by helping in feature selection for various machine learning techniques.

Acknowledgments

We would like to thank Prof. Rajeev Sangal for providing valuable inputs throughout the work.

References

- R. Begum, S. Husain, A. Dhvaj, D. Sharma, L. Bai, and R. Sangal. Dependency annotation scheme for Indian languages. 2008. In proceedings of Third International Joint Conference on Natural Language Processing (IJCNLP), Hyderabad, India
- A. Bharati, S. Husain, D. Misra, and R. Sangal. Two stage constraint based hybrid approach to free word order language dependency parsing. 2009a. In Proceedings of the 11th International Conference on Parsing Technologies (IWPT). Paris.
- A. Bharati, D. Sharma, S. Husain, L. Bai, R. Begam, and R. Sangal. Anncorra: Treebanks for indian languages, guidelines for annotating hindi treebank. 2009b. <http://ltrc.iiit.ac.in/MachineTrans/research/tb/DS-guidelines/DS-guidelines-ver2-28-05-09.pdf>
- A. Bhatia, R. Bhatt, B. Narasimhan, M. Palmer, O. Rambow, D. Sharma, M. Tepper, A. Vaidya, and F. Xia. Empty Categories in a Hindi Treebank. 2010. In the Proceedings of the 7th International Conference on Language Resources and Evaluation (LREC).
- R. Campbell. Using linguistic principles to recover empty categories. 2004. In Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics
- A. Chanev. Portability of dependency parsing algorithms—an application for Italian. 2005. In Proc. of the fourth workshop on Treebanks and Linguistic Theories (TLT). Citeseer.
- M. Collins. Three generative, lexicalised models for statistical parsing. 1997. In Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics.
- P. Dienes and A. Dubey. Antecedent recovery: Experiments with a trace tagger. 2003a. In Proceedings of the 2003 conference on Empirical methods in natural language processing.

- P. Dienes and A. Dubey. Deep syntactic processing by combining shallow methods. 2003b. In Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1.
- D. Higgins. A machine-learning approach to the identification of WH gaps. 2003. In Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics-Volume 2.
- X. Fei, O. Rambow, R. Bhatt, M. Palmer, and D. Sharma. Towards a multi-representational treebank. 2008. Proc. of the 7th Int'l Workshop on Treebanks and Linguistic Theories (TLT-7)
- M. Johnson. A simple pattern-matching algorithm for recovering empty nodes and their antecedents. 2002. In Proceedings of the 40th Annual Meeting on Association for Computational Linguistics.
- M. Marcus, M. Marcinkiewicz, and B. Santorini. Building a large annotated corpus of English: The Penn Treebank. 1993. Computational linguistics.
- R. McDonald, F. Pereira, K. Ribarov, and J. Hajič. Non-projective dependency parsing using spanning tree algorithms. 2005. In Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing.
- J. Nivre. An efficient algorithm for projective dependency parsing. 2003. In Proceedings of the 8th International Workshop on Parsing Technologies (IWPT).
- M. Palmer, R. Bhatt, B. Narasimhan, O. Rambow, D. Sharma, and F. Xia. Hindi Syntax: Annotating Dependency, Lexical Predicate-Argument Structure, and Phrase Structure. 2009. In The 7th International Conference on Natural Language Processing.
- Y. Yang and N. Xue. Chasing the ghost: recovering empty categories in the Chinese Treebank. 2010. In Proceedings of the 23rd International Conference on Computational Linguistics: Posters.