Automatic Clause Boundary Annotation in the Hindi Treebank

Rahul Sharma, Soma Paul, Riyaz Ahmad Bhat and Sambhav Jain
Language Technology Research Centre, IIIT-Hyderabad, India
{rahul.sharma, riyaz.bhat, sambhav.jain}@research.iiit.ac.in, soma@iiit.ac.in

Abstract

In this paper, we propose a method for automatic clause boundary annotation in the Hindi Dependency Treebank. We show that the clausal information implicitly encoded in a dependency structure can be made explicit with no or less human intervention. We exercised the proposed approach on 16,000 sentences of Hindi Dependency Treebank. Our approach gives an accuracy of 94.44% for clause boundary identification evaluated over 238 clauses. The resultant corpus has varied usages and can be utilized for developing a statistical clause boundary identifier.

1 Introduction

Clause boundary is important for various NLP systems like machine translation, parallel corpora alignment, parsing etc. (Leffa, 1998; Gadde et al., 2010; Ejerhed, 1988). This information is furnished by an automatic tool often called *clause boundary identifier*. Both data driven (Puscasu, 2004) and rule based (Leffa, 1998) approaches have been explored in past for building such a system, however recent inclination has been towards the data-driven approaches due to their robustness. In order to built a clause boundary identifier, using data driven approach, one needs to have a good clause boundary annotated corpus for training. At present, such a resource is not available in Hindi. However, the syntactic treebank with dependency relations annotated has been developed. We wish to expand this manually annotated treebank with the clause boundary annotation in this work.

Several insightful approaches, in past, have enriched existing resources by first utilizing the explicit information available to derive new implicit information (Klein and Manning, 2003; Kosaraju et al., 2012) and then explicitly annotating it back into the original resource. Conversion of a treebank from one grammatical formalism to the other serves as a good example of how an implicit information can be mapped and extracted (Xia and Palmer, 2001). Instead of starting from scratch, an already existing treebank is transformed into a new grammatical formalism. Bhatt et al. (2009) is one such effort for Hindi. They have automatically transformed dependency structures to phrase structure utilizing Hindi Dependency Treebank and Hindi PropBank (Palmer et al., 2009). Following such insights, we attempt to automatically generate the clause marked data from existing resources for Hindi. We propose that the clause information is implicitly encoded in the Hindi Dependency Treebank and thus, can be extracted and explicitly specified as an additional layer of annotation in the treebank. This paper presents a systematic approach towards incorporating clausal information in the Hindi Dependency Treebank utilizing the information (mopho-syntactic, dependency etc.) already available in the treebank.

This paper is structured as follows: In Section 2, we discuss the related works that have been done earlier on clause identification and classification. In Section 3, we talk about clause and its types. In Section 4, we discuss about Hindi-Urdu treebank. Section 5 describes our methodology and in Section 6 we discuss the results achieved and outline the issues faced. In Section 7, we conclude with some future directions.

2 Related Work

In this section, we report some of the works related to clause boundary marking. In general, for the task of clause boundary identification two kinds of resources are used: (a) typed dependency structures; (b) lexical cues such as subordinate and coordinate conjuncts. However, the works reported on Indian languages have mainly used typed de-

27th Pacific Asia Conference on Language, Information, and Computation pages 499—504

pendency structures. Ghosh et al. (2010) has developed a rule based system for identifying clause boundary for Bangla. They have defined clause as a composite construction of a verb along with its dependent chunks. The rules are designed on the basis of dependency relation in an annotated corpus. They use CRF based statistical system for labeling different clauses. Dhivya et al. (2012) reports the task of identifying clauses in Tamil. They first preprocess the input sentence using Maltparser which gives dependency tree as its output. They have proposed 11 different dependency tags. Using those dependency tags marked by Maltparser, they try to find clause boundary in a sentence. Another work on Tamil (Ram and Devi, 2008) have proposed a hybrid approach for detecting clause boundaries in a sentence. They have used CRF based system which uses different linguistic cues for the task. After identification of the clause boundaries they run error analyzer module to find the false boundaries, which are then corrected by the rule based system built using linguistic cues.

Leffa (1998) has proposed a rule based system for English. This system uses lexical cues such as subordination conjuncts, coordination conjuncts etc. for indentification of clause boundaries and the type of a clause. Puscasu (2004) proposed a multilingual method of combining language independent machine learning techniques with language specific rules to detect clause boundaries in unrestricted texts. The rules identify the finite verbs and clause boundaries not included in learning process. Gadde et al. (2010) used some heuristic rules for clause boundary marking in Hindi. Their aim was to see the impact of clausal information on parser performance.

3 Clause and its Classification

A clause is a group of words consisting of a verb (or a verb group) and its arguments (explicit and implicit). Depending on the type of the verb, a clause is classified either as finite or non-finite based on the finiteness of the head verb. For example:

(1) raam khana khaakar ghar gayaa.
Ram food eat+do home go+past.
'Ram went home after eating.'

In this example (1), *khana khaakar* is a non-finite clause since 'khaakar' is a non-finite verb. Similarly, *raam ghar gayaa* is a finite clause as 'gayaa' is a finite verb. A sentence can have more than one clauses in it. These clauses are classified in to two types as:

1. Main clause, which is an independent clause, is also called Superordinate clause,
2. Subordinate clause, which is dependent on the main clause.

Clauses can also be classified based on their function in a sentence such as complement clause, adverbial clause, relative clause etc. (discussed shortly). Based on the relative position of clauses with respect to each other, clauses can either be nested or non-nested. Nested here means one clause is embedded in another clause, while non-nested means they lie adjacent to each other. For example,

(2) raam jo khela , ghar gayaa
Ram who play+past , home go+past
'Ram who played , went home.'

In example (2) the two clauses are: 1) *raam ghar gayaa* (a non-embedded clause) 2) *jo khela* (an embedded clause), which is embedded in *raam ghar gayaa*.

Below, we discuss some of the clause types mentioned earlier.

(a) **Complement Clause**

These clauses are introduced by complementizer 'ki' (that) and generally follow the verb of main clause (Koul, 2009).

(3) yaha sach hai ki mohan bimaara hai
It true is that Mohan sick is
'It is true that Mohan is sick'

In example (3), *ki mohan bimaar hai* is a Complement clause and 'ki' is a complementizer.

It must be noted that 'complement clause' may also act an argument of the main clause verb. So, in example (3), the main clause is *yaha sach hai ki mohan bimaara hai*, which contains the complement clause *ki mohan bimaara hai*, in it. This is considered to be a special case where a clause comes as an argument of a verb and becomes a part of the main clause. We have handled this type of construction separately (discussed in section 5).

27th Pacific Asia Conference on Language, Information and Computation pages 499－504

(b) **Relative Clause**

Finite relative clauses occur as a modifier of verb's argument and contain a relative pronoun (Koul, 2009). Such clauses can be either nested or non-nested. For example:

(4) vaha ladkaa jo khel rahaa thaa ghar gayaa
that boy who play+past+conti. home go+past
'That boy who was playing went home'

In example (4), the nested relative clause is *jo khel rahaa thaa* (who was playing) with 'jo' as a relative marker. 'jo' modifies 'vaha', the argument of the verb 'gayaa'.

Consider another example:

(5) vaha ladkaa ghar gayaa jo khel rahaa thaa
that boy home go+past who play+past+conti.
'That boy who was playing went home'

In example (5) relative clause *jo khel rahaa thaa* is an example of an extraposed relative clause.

(c) **Coordinate Clause**

It is one of the independent clauses in a sentence belonging to a series of two or more independent clauses co-ordinated by a coordinating conjunction (Koul, 2009). For example:

(6) main ghar jaaungaa aur raam dillii jaayegaa
I home go+fut. and Ram delhi go+fut
'I will go home and Raam will go to Delhi'

mai ghar jaaungaa and *raam dillii jaayegaa* are two independent clauses with the same status in example (6). In our work, we consider both clause as coordinate clauses, and the coordinating conjunct is not taken to be part of any of the two clauses. There is thus no hierarchy in these clauses.

4 Hindi Dependency Treebank

In this section, we give an overview of Hindi Treebank (HTB ver-0.51) a part of which was released for Hindi Dependency Parsing shared task, MTPIL, COLING 2012 (Sharma et al., 2012). It is a multi-layered dependency treebank with morphological, part-of-speech and dependency annotations based on the Computational Pāṇinian Grammatical (CPG) framework. In the dependency annotation, relations are mainly verb-centric. The relation that holds between a verb and its arguments is called a $kaṛaka$ relation. Besides $kaṛaka$ relations, dependency relations also exist between nouns (genitives), between nouns and their modifiers (adjectival modification, relativization), between verbs and their modifiers (adverbial modification including subordination). CPG provides an essentially syntactico-semantic dependency annotation, incorporating $kaṛaka$ (e.g., agent, theme, etc.), non-$kaṛaka$ (e.g. possession, purpose) and other (part of) relations. A complete tagset of dependency relations based on CPG can be found in (Bharati et al., 2009), the ones starting with 'k' are largely Pāṇinian $kaṛaka$ relations, and are assigned to the arguments of a verb. Example (7) shows the three levels of information discussed above encoded in the SSF format.

(7) raam ne khaanaa khaayaa aur paani piyaa.
Ram+erg food ea+past and water drink+past
'Raam who ate food and drank water, went home'

Offset	Token	Tag	Feature structure
1	((	NP	<fs name=NP drel=k1:VGF>
1.1	raama	NNP	<fs af='raama,n,m,sg,3,d,0,0'>
1.2	ne	PSP	<fs af='ne,psp,,,,,,'>
	))		
1 2	((	NP	<fs name=NP2 drel=k2:VGF>
2.1	khaanaa	NN	<fs af='khaanaa,n,m,sg,3,d,0,0' name="khaanaa">
	))		
3	((	VGF	<¡fs name=VGF drel=ccof:CCP>
3.1	khaayaa	VM	<fs af='KA,v,m,sg,any,,yA,yA' name="khaayaa">
	))		
4	((	CCP	<¡fs name=CCP>
4.1	aur	CC	<fs af='Ora,avy,,,,,,' name="aur">
	))		
5	((	NP	<¡fs name=NP3 drel=k2:VGF2>
5.1	paani	NN	<fs af='pAnI,n,m,sg,3,d,0,0' name="paani">
	))		
6	((	VGF	<¡fs name=VGF2 drel=ccof:CCP>
6.1	piyaa	VM	<fs af='pIyA,unk,,,,,,' name="piyaa">
	))		

Figure 1: SSF representation for example 7

In figure 1, the preterminal node is a part of speech (POS) of a lexical item. These parts of speech are grouped together to form chunks (eg. NP, VGF, CCP, VGNF etc.) as a part of sentence analysis. The dependency relations are marked at chunk level, marked with drel in above SSF format. $k1$ is the agent of the action and $k2$ is the object of the verb. There are two $k2's$ for two different verbs, $khaanaa$ 'food' is $k2$ for $khaayaa$ 'eat' verb and $paani$ 'water' is $k2$ for $piyaa$ 'drink' verb.

5 Method

As we discussed earlier, we use dependency attachments and dependency relations annotated in the treebank to automatically mark the clause boundaries. The assumption is that the left most and the right most projections (dependents) of a verb are the extremes of a clause it heads.

Our approach is composed of two steps which execute sequentially to identify boundaries of a clause. Step 1 identifies the clause boundary in general, while Step 2 is a post-processing step which do adjustments specifically to handle 'ki' (that) complement clauses.

STEP 1: In this step, we first extract all verbs in a sentence using POS tag and chunk information and then traverse the dependency tree to extract their dependents recursively one by one. For each verb in the list, we stop traversing if either we exhaust the nodes dominated by the verb or find another verb in its dominance. However, when a complement clause introduced by complementizer 'ki' is annotated as an argument of a verb we will continue traversing till we exhaust all the nodes dominated by the complementizer 'ki'. This will ensure that the complement clause be treated as part of the main clause, more like an embedded clause. Once verb and its dependents are obtained, we sort them by their offsets. The lowest offset is considered as the start of a clause and the highest offset marks its end. This way we determine boundaries of each clause in a sentence.

Example (8) illustrates STEP 1:

(8) raam ghar gayaa aur khaanaa khaayaa.
Ram home go+past and food eat+past
'Ram went home and ate food.'

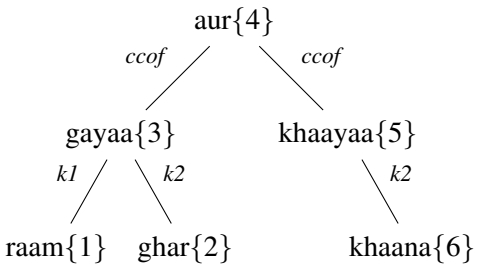

Figure 2: Dependency Tree

Figure 2 shows the dependency tree of example (8). Relations (k1, k2 etc.) are marked on edges and Offsets of different chunks are shown in brackets with the words. Following STEP 1, a verb list containing two verbs–'gayaa' and 'khaayaa' is formed. Then, after traversing the dependency tree of example (8) for verb 'gayaa', a list, containing verb 'gayaa' and its arguments–'raam' and 'ghar' is built. This list is, then, sorted by the offsets of words contained in it. After sorting, the words corresponding to the lower and higher offsets are treated as the boundaries of the clause headed by the verb 'gayaa'. Similarly for 'khaayaa' verb, words at offset 5 and offset 6 mark the boundaries. Thus, the clause boundaries for example (8) will be marked as:

$(raam\ ghar\ gayaa)\ aur\ (khaanaa\ khaayaa.)$

STEP 2: This step, as a postprocessing step, handles the exceptional case of 'ki' (that) complex complement clauses. As mentioned earlier, 'ki' complement clause may occur as an argument of a verb and could be thus a part of its clause. Although, in STEP 1 we will accurately include the complement clause as a part of the main clause, we don't mark the scope of complement clause itself, if it is complex i.e., made of more than one clause. This step marks the scope of complex complement clauses based on the output of STEP 1. Example (9) explains this further.

(9) raam ne kahaa ki tum ghar jaao or
ram+erg say+past that you home go and
aaraam karloo
rest do
'Ram said that you go home and take rest.'

After STEP 1, the clause boundaries for the sentence (9) would be like:

$(raam\ ne\ kahaa\ ki\ (tum\ ghar\ jaao)\ or\ (aaraam\ karloo))$

In STEP 2, we iterate over the output of STEP 1 and mark the boundaries of the complement clause starting from the word imediately following the 'ki' complementizer and the ending with the end of main clause of which complement clause is a part. The modified boundaries will be:

$(raam\ ne\ kahaa\ ki\ ((tum\ ghar\ jaao)\ or\ (aaraam\ karloo)))$

6 Results and Discussion

A testing set of 100 sentences containing 288 clauses randomly selected from a section of the Hindi Dependency Treebank is used to evaluate the performance of our approach. The accuracies are calculated on the basis of the following aspects of a clause:

- Start of clause
- End of clause
- Whole clause
- Finite clause
- Non-finite clause
- Embedded clause
- Non-embedded clause

Table 1 shows the accuracies of our approach for different aspects of clause marking.

Different aspects	Accuracy%
Start of clause	97.91
End of Clause	94.44
Whole clause	94.44
Finite clause	93.88
Non-Finite clause	98.30
Embedded clause	94.32
Non-Embedded clause	94.55

Table 1: Results of different aspect of clause

While evaluating our approach, we come across some constructions which were not handled by it. They are:

1. **Topicalisation**: Extraction of a constituent from in its canonical position to clause initial position may sometimes affect the representation of actual clause boundaries. Extraction from subordinate clause to sentence initial position provides such an example:

   (10) raam$_i$ maine kahaa ki t_i ghar gayaa.
   Ram I+Erg say+past that home go+past
   'I said that raam went home.'

   In example (10) 'raam' moved from its default position t_i to the sentence initial position. The overlap in the constituents of main and subordinate clauses in (10) makes the representation of clause boundaries in such sentences difficult.

2. **Inconsistencies in the treebank**: Since we rely on manually annotated dependency structures to identify the clause boundaries, any inconsistency in the structure would affect the accurate marking of such information. We spotted some errors which were due to the inconsistencies in the annotation in the treebank like part of speech and attachment errors.

7 Conclusion and future work

In this paper, we showed how implicit clausal information captured in a dependency tree can be extracted and added back to the original resource. We worked with the Hindi Dependency Treebank and automatically added the clausal information using the dependencies between constituents in the treebank. We discussed some of the issues in identifying clause boundaries using our approach. In the future, we plan to use the clause boundary annotated corpus furnished in this work for the task of clause boundary identification in raw text using machine learning.

Acknowledgments

The work reported in this paper is supported by the NSF grant (Award Number: CNS 0751202; CFDA Number: 47.070). [1]

References

Akshar Bharati, Dipti Misra Sharma, Samar Husain, Lakshmi Bai, Rafiya Begum, and Rajeev Sangal. 2009. Anncorra: Treebanks for indian languages guidelines for annotating hindi treebank (version–2.0).

Rajesh Bhatt, Bhuvana Narasimhan, Martha Palmer, Owen Rambow, Dipti Misra Sharma, and Fei Xia. 2009. A multi-representational and multi-layered treebank for hindi/urdu. In *Proceedings of the Third Linguistic Annotation Workshop*, pages 186–189. Association for Computational Linguistics.

R Dhivya, V Dhanalakshmi, M Anand Kumar, and KP Soman. 2012. Clause boundary identification for tamil language using dependency parsing. In *Signal Processing and Information Technology*, pages 195–197. Springer.

Eva I Ejerhed. 1988. Finding clauses in unrestricted text by finitary and stochastic methods. In *Proceedings of the second conference on Applied natural language processing*, pages 219–227. Association for Computational Linguistics.

Phani Gadde, Karan Jindal, Samar Husain, Dipti Misra Sharma, and Rajeev Sangal. 2010. Improving data driven dependency parsing using clausal information. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 657–660. Association for Computational Linguistics.

[1] Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

Aniruddha Ghosh, Amitava Das, and Sivaji Bandyopadhyay. 2010. Clause identification and classification in bengali. In *23rd International Conference on Computational Linguistics*, page 17.

Dan Klein and Christopher D Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 423–430. Association for Computational Linguistics.

Prudhvi Kosaraju, Samar Husain, Bharat Ram Ambati, Dipti Misra Sharma, and Rajeev Sangal. 2012. Intra-chunk dependency annotation: expanding hindi inter-chunk annotated treebank. In *Proceedings of the Sixth Linguistic Annotation Workshop*, pages 49–56. Association for Computational Linguistics.

Omkar Nath Koul. 2009. *Modern Hindi Grammar*. Indian Institute of Language Studies.

Vilson J Leffa. 1998. Clause processing in complex sentences. In *Proceedings of the First International Conference on Language Resources and Evaluation*, volume 1, pages 937–943.

Martha Palmer, Rajesh Bhatt, Bhuvana Narasimhan, Owen Rambow, Dipti Misra Sharma, and Fei Xia. 2009. Hindi syntax: Annotating dependency, lexical predicate-argument structure, and phrase structure. In *The 7th International Conference on Natural Language Processing*, pages 14–17.

Georgiana Puscasu. 2004. A multilingual method for clause splitting. In *Proceedings of the 7th Annual Colloquium for the UK Special Interest Group for Computational Linguistics*.

R Vijay Sundar Ram and Sobha Lalitha Devi. 2008. Clause boundary identification using conditional random fields. In *Computational Linguistics and Intelligent Text Processing*, pages 140–150. Springer.

Dipti Misra Sharma, Prashanth Mannem, Joseph van-Genabith, Sobha Lalitha Devi, Radhika Mamidi, and Ranjani Parthasarathi, editors. 2012. *Proceedings of the Workshop on Machine Translation and Parsing in Indian Languages*. The COLING 2012 Organizing Committee, Mumbai, India, December.

Fei Xia and Martha Palmer. 2001. Converting dependency structures to phrase structures. In *Proceedings of the first international conference on Human language technology research*, pages 1–5. Association for Computational Linguistics.