A hybrid approach for automatic clause boundary identification in Hindi

Rahul Sharma, Soma Paul
Language Technology Research Centre, IIIT-Hyderabad, India
rahul.sharma@research.iiit.ac.in, soma@iiit.ac.in

Abstract

A complex sentence, divided into clauses, can be analyzed more easily than the complex sentence itself. We present here, the task of clauses identification in Hindi text. To the best of our knowledge, not much work has been done on clause boundary identification for Hindi, which makes this task more important. We have built a Hybrid system which gives 90.804% F1-scores and 94.697% F1-scores for identification of clauses' start and end respectively.

1 Introduction

Clause is the minimal grammatical unit which can express a proposition. It is a sequential group of words, containing a verb or a verb group(verb and its auxiliary), and its arguments which can be explicit or implicit in nature (Ram and Devi, 2008) . This makes clause an important unit in language grammars and emphasis the need to identify and classify them as part of linguistic studies.
Analysis and processing of complex sentences is a far more challenging task as compared to a simple sentence. NLP applications often perform poorly as the complexity of the sentence increases. "It is impossible, to process a complex sentence if its clauses are not properly identified and classified according to their syntactic function in the sentence" (Leffa, 1998). Further, identifying clauses, and processing them separately are known to do better in many NLP tasks. The performance of many NLP systems like Machine Translation, Parallel corpora alignment, Information Extraction, Syntactic parsing, automatic summarization and speech applications etc improves by introducing clause boundaries in a sentence (e.g., Ejerhed, 1988; Abney, 1990; Leffa, 1998; Papageorgiou, 1997; Gadde et al., 2010).
We present a hybrid method which comprises of Conditional random fields(CRFs) (Lafferty et al., 2001) based statistical learning followed by some rules to automatically determine *'clause'* boundaries (beginnings and ends) in complex or compound sentences. CRFs is a framework for building undirected probabilistic graphical models to segment and label sequence data (Lafferty et al., 2001). In past, this framework has proved to be successful for sequence labeling task (Sha and Pereira, 2003; McCallum and Li, 2003). Van Nguyen et al. (2007) used CRFs for clause splitting task with some linguistic information giving 84.09% F1-score.

Our system has minimum dependency on linguistic resources,only part of speech (POS) and chunk information of lexical items is used with a fair performance of the system. As far as we know, not much work has been done on clause boundary identification for Hindi and this makes this task more significant. This paper is structured as follows: In Section 2, we discuss the related works that has been done earlier on clause identification. Section 3 describes the creation of dataset for various system use. In Section 4, we talk about methodology of our system. Section 5 outlines the system performance. In section 6, some issues related clause identification are discussed. In Section 7, we conclude and talk about future works in this area.

Proceedings of the 5th Workshop on South and Southeast Asian NLP, 25th International Conference on Computational Linguistics, pages 43–49,
Dublin, Ireland, August 23-29 2014.

2 Related works

Studies in identifying clauses date back to (Ejerhed, 1988) work, where they showed how automatic clause boundary identification in discourse can benefit a parser's performance. However her experiments could detect only basic clauses. Later (Abney, 1990) used clause filter as part of his CASS parser. (Papageorgiou, 1997) used hand crafted rules to identify clause boundaries in a text. (Leffa, 1998) is another rule based method which was implemented in an English-Portuguese MT system.
Some more recent works in this area are: (Puscasu, 2004), in which she proposed a multilingual method of combining language independent ML 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. (Ram and Devi, 2008) proposed a hybrid based approach for detecting clause boundaries in a sentence. They have used a CRF based system which uses different linguistic cues. After identifying the clause boundaries they run an error analyzer module to find false boundary markings, which are then corrected by the rule based system, built using linguistic clues. (Ghosh et al., 2010) is another rule based system for clause boundary identification for Bengali, where they use machine learning approach for clause classification and dependency relations between verb and its argument to find clause boundaries. Dhivya et al. (2012) use dependency trees from maltparser and the dependency tag-set with 11 tags to identify clause boundaries. Similar to (Dhivya et al., 2012), Sharma et al. (2013) showed how implicit clause information present in dependency trees can be used to extract clauses in sentences. Their system have reported 94.44% accuracy for Hindi.Gadde et al. (2010) reported improvement in parser performance by introducing automatic clause information in a sentence for Hindi in 'Improving data driven dependency parsing using clausal information'. However their approach for identifying clause information has not been discussed. Thus a comparison is not possible here.

3 Dataset

In Hindi, We don't have any data available annotated with clause boundary, So to generate clause annotated corpora we have used (Sharma et al., 2013) technique where they have showed how implicit clause information present in dependency trees can be used to extract clauses in sentences. By this technique we have automatically generated 16000 sentences of Hindi treebank (Palmer et al., 2009) marked with clause boudaries. Out of which, 14500 sentences were taken as training set, 500 for development set and remaining 1000 sentences for testing set. As these sentences were generated automatically there are chances of noises in form of wrongly marked clause boundaries, so for proper evaluation of the system, we have manually corrected the wrongly marked clauses in development and testing sets.

4 Methodology

We propose a hybrid system which identifies the clause(s) in the input sentence and marks the *'clause start position'* (**CSP**) and *'clause end position'* (**CEP**) with brackets.
Hindi usually follows the SOV word order, so ends of the clauses can be found by just using verb information, in most of the cases. The language also has explicit relative pronouns, subordinating conjuncts, coordinate conjunctions etc. which serve as cues that help to identify clause boundaries of the clauses. Apart from the lexical cues we have also used POS tag and chunk information to built our system.
Our system comprise of two main modules; first modules is stochastic model which have been trained on 14500 sentences, and second module which is built using hand crafted rules.

4.1 Stochastic Models

We have used two techniques to built two different models; 1) step-by-step model and 2) merged model, using CRF machine learning approach. Both the models take word, word's POS tag and its suffix as word's features for training. Table (1) shows the common features used for training models. These feature are syntactic in nature, and relative pronoun, verb, conjunctions etc. plays important role in identifying boundaries. suffixes help to learn morphological feature of the word.

Present word's Lexicon, POS tag, last character, last two character, and last three character
Previous four words' Lexicon and POS tags
Next four words' Lexicon and POS tags
Next three words' last character, last two character, last three character

Table 1: Features

4.1.1 step-by-step model

This model comprises of two models; end model and start model. First one identifies the end of a clause and then later one takes the output of former model as input and identifies the start of the clause. In this technique we can notice that both models have to only mark whether a word is a boundary of a clause or not. For example 'end model' has to check whether a given word is a end(boundary) of a clause or not. Below example (1) explains this further.

(1) raam jisne khaanaa khaayaa ghar gayaaa
Ram who food eat+past home go+past
'Raam who ate food, went home'

In example (1), end model first marks 'gayaa' and 'khaayaa' as the end of clause. Then start model takes this additional information also as the feature, and marks 'raam' and 'jisne' as the start of clause.

4.1.2 Merged Model

This model marks the clauses' start and end in one go. Unlike the step-by-step model, it check whether a word is clause's start, clause's end or none. For above example (1), it will mark 'gayaa' and 'khaayaa' as the end of clause, and 'raam' and 'jisne' as the start of clause respectively in one go.

-- Keeping post-processing module(discussed below) same, we have evaluated our system using both stochastic models separately, and observed, system with step-by-step model gives high F1-score value than the system with merged model.

4.2 Post-processing Module

This module processes the output from stochastic model, and mark the boundaries of clauses in sentences. As we know, in a sentence CSPs should always be equal to CEPs. So on the basis of difference between CSPs and CEPs, we have formalized our rules. Below is the description of rules used in this module.

1. Rules, when CSPs are greater than CEPs are:
   (a) Check for 'ki' complement clause: The verb in a sentence which contain 'ki' compliment clause is not the end of its clause whereas its end is same as of end of 'ki' complement clause. Below example (2) will make this rule more clearer.

   (2) raam ne kahaa ki vaha ghar gayaa
   Ram+arg say+past that he home go+past
   'Raam said that he went home'

   In this example (2), Stochastic models will mark 'raam' and 'ki' as the start of clause, and 'gayaa' as the end of clause, making CSPs more than the CEPs. We can notice that 'gayaa' is the end for both the clauses in a sentence, so using this rule, we will add one more end of clause to 'gayaa' word. The resultant sentence with clauses marked will be:
   (raam ne kahaa (ki vaha ghar gayaa))
   (b) Check for finite verb: If a verb is finite and does not have any 'ki' complement clause in it then that verb should be marked as the end of clause. So if this type verb is unmarked by the stochastic model then this rule will handle this.
   (c) Check for non-finite verb: If a non-finite verb is present in a sentence and word next to it does not mark start of another clause then this rule will mark that word as the start of that clause.

–It should be noted that rules are applied in specific order, and once the number of CSPs and CEPs become same at any point of rule we stop applying more rules from this type where CSPs and CEPs are not same.

2. Rules, When CEPs are greater than CSPs are:
   (a) If there is a 'non-finite' verb in a sentence then we check for its start and mark them using regular expressions if not marked by stochastic models. for example:

   (3) raam khaanaa khakara ghar gayaa
   Ram+arg food having eaten home go+past
   'having eaten food, Ram wen home'

   In example (3), if stochastic models does not able to mark 'khaanaa' as the start of non-finite clause 'khaanaa khakara'. Then this rules will capture these type of situations and add a new CSP in a sentence.
   (b) If a word before conjunction, not a verb, is marked as end of a clause then this rule will remove that end, reducing number of CEP.

3. Rules, when CSPs and CEPs are same:
   (a) If there are more than one clauses in one single 'ki' complement clause than this rules marks one bigger boundary as clause which will contain all the embedded clauses. For example:

   (4) raam ne kahaa ki shaam ne khaanaa khaayaa aur paani piyaa
   Ram+arg say+past that Shaam+arg food eat+past and water drink+past
   'Raam said that Shaam ate food and drank water'

   The situation discussed in this rule can be observed in example (4). The system output before this rule may be,
   "(raam ne kahaa (ki shaam ne khaanaa khaayaa) aur (paani piyaa))", Which this rule will convert to
   "(raam ne kahaa (ki (shaam ne khaanaa khaayaa) aur (paani piyaa)))"

– Having these rules applied, the output sentence will contain start and end of clauses in a sentence.

5 Evaluation and Results

As mentioned earlier we have used (Sharma et al., 2013) technique to automatically generate 16000 sentences of Hindi treebank marked with clause boundaries. Out of these 16000 sentences, a set of 1500 sentences with average length of 16 words was randomly selected. This set was then manually corrected at the level of clause boundary for accurate evaluation of the system. It should be noted that this set was not used in training of the models. Further we have divided this set into two set; development set which consist of 500 sentences and testing set which consist of 1000 sentences. We have evaluated the system with both models (step-by-step and merged) along with post-processing module, and we have noticed system with step-by-step model performs better than the system with merged model. Table (2) and Table (3) show the results on development set and testing set respectively.

Model Type	**Start of clause**			**End of clause**		
	Precision	Recall	F1-measure	Precision	Recall	F1-measure
Step-by-step model	91.493	89.816	90.646	95.129	93.482	94.298
Merged Model	92.171	89.918	91.030	90.927	92.871	91.888

Table 2: Results on development set.

Model Type	Start of clause			End of clause		
	Precision	Recall	F1-measure	Precision	Recall	F1-measure
Step-by-step model	92.051	89.590	90.804	95.969	93.458	94.697
Merged Model	91.779	88.907	90.320	90.919	92.263	91.586

Table 3: Results on testing set.

6 Error Analysis and Discussion

While evaluating our both systems (system with step-by-step model and system with merged model), we come across some constructions which were not handled by them. which are:

1. Ellipses of verb: when a verb is omitted in a sentence then it is not possible for our system to mark boundaries correctly. For example:

   (5) raam ne kitaab <V> aur maine kavitaa padhii
   Ram+erg book <read+past> and I+erg poem read+past
   'Ram read a book and I read a poem'

   In example (5), there is an ellipses of the verb 'padhi' in the clause 'raam ne kitaab'. Thus, though the sentence has two clauses–'raam ne kitaab' and 'maine kavitaa padhii', our system incorrectly identifies the whole sentence as one clause due to the ellipses of the verb (denoted by <V>).

2. Scrambling in the usual word order, which is SOV in Hindi, is likely to induce incorrect identification of the clauses in our system. For Example:

   (6) ghar gayaa raam, vaha bolaa.
   home go+past Ram, he say+past
   'He said Ram went home'

   In example (6), Our system is unable to identify the clause boundaries correctly for any of the two clauses, 'ghar gayaa raam' and 'ghar gayaa raam,vaha bolaa', due to scrambling in the word order. Its output for the sentence is '(ghar) (gayaa raam, vaha bolaa)', though the output should be '((ghar (gayaa raam,) vaha bolaa)'.

3. Missing subordinate conjunction 'ki' in a sentence also leads to incorrect identification of clause boundaries by our system. For example:

   (7) raam ne kahaa tum ghar jaao
   Ram+erg say+past you home go
   'Ram said you go home'

   The missing subordinate conjunction 'ki' in example (7) leads to incorrect marking of the clause boundaries as: '(raam ne kahaa) (tum ghar jaao)'. The correct clause boundaries for the sentence are '(raam ne kahaa (tum ghar jaao))'.

4. **Start of non-finite clause**: As we don't find any syntactic cues for start of non-finite clause, our systems does not perform much efficiently in finding start of non-finite clauses. For example:

   (8) ab hum alag maslon para khulkara baatchit kar rahe hain
   now we different matters/topics on openly discussion do+conti+present
   'Now we are discussing openly on different matters'

Both system marks 'khulkara' and 'kar rahe hain' verbs as the end of clauses accurately but start of non-finite clause which is 'alag' is not identified correctly. Output by the systems is, '(ab hum alag maslon para khulkara) (baatchit kar rahe hain)' , where as the correct output is, '(ab hum (alag maslon para khulkara) baatchit kar rahe hain)'

-- Overall we observed that the system with step-by-step model which statistically first identifies end and then start, followed by rules performs better than the system with merged model.

7 Conclusion and Future Work

We have discussed our work on clause boundary identification in Hindi and the issues pertaining to them, in the course of this paper. Clausal information in a sentence is known to improve the performance of many NLP systems, thus the need for this task. We observed that the system with step-by-step model which statistically, first identifies end and then start of clauses, followed by rules, performs better than the system with merged model. The step-by-step model system, showing a satisfactory performance in terms of F1 scores of 91.53% for clause boundary identification, and the merged model system showing 80.63% for the same. Since this task is a promising resource for NLP systems such as Machine Translation, Text-to-Speech and so on, and can contribute to their better performance, applying this system for betterment of NLP tools seems quite a favorable prospect as a future work. (Gadde et al., 2010) report that even minimal clause boundary identification information leverages the performance of their system. We would like to test the performance of our system in terms of leveraging the performance of other NLP systems

References

Steven Abney. 1990. Rapid incremental parsing with repair. pages 1–9.

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

Eva I Ejerhed. 1988. Finding clauses in unrestricted text by finitary and stochastic methods. 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. pages 657–660. Association for Computational Linguistics.

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

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

Vilson J Leffa. 1998. Clause processing in complex sentences. volume 1, pages 937–943.

Andrew McCallum and Wei Li. 2003. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 188–191. Association for Computational Linguistics.

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. pages 14–17.

Harris V Papageorgiou. 1997. Clause recognition in the framework of alignment. pages 417–426.

Georgiana Puscasu. 2004. A multilingual method for clause splitting.

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.

Fei Sha and Fernando Pereira. 2003. Shallow parsing with conditional random fields. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 134–141. Association for Computational Linguistics.

Rahul Sharma, Soma Paul, Riyaz Ahmad Bhat, and Sambhav Jain. 2013. Automatic clause boundary annotation in the hindi treebank.

Vinh Van Nguyen, Minh Le Nguyen, and Akira Shimazu. 2007. Using conditional random fields for clause splitting. *Proceedings of The Pacific Association for Computational Linguistics, University of Melbourne Australia*.