A rule based approach for automatic clause boundary detection and classification 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 identification and classification of clauses 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 rule based system using linguistic cues such as coordinating conjunct, subordinating conjunct etc. Our system gives 91.53% and 80.63% F1-scores for identification and classification for finite clauses respectively, and 60.57% accuracy for non-finite clauses.

1 Introduction

A 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 a 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). 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 rule based method to automatically determine *'clause'* boundaries (beginnings and ends) in complex or compound sentences, and further categorize the identified clauses according to their types. Thus our system is made up of two parts, the first determines the boundaries of the clauses (clause identification) and the second part determines the type of the clause (Clause Classification). Rules for the system were framed by thoroughly analyzing the Hindi-Urdu treebank (Palmer et al., 2009). This provides significant insights for the task as clause boundaries can be inferred from the dependency relations marked in dependency trees. The rules devised for our system have 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 talk about clause and its types. In Section 3, we discuss the related works that has been done earlier on clause identification and classification. Section 4 describes the data flow of our system and rules for identifying and classification of a clause. 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.

2 Clause and its Types

As defined in introduction, a clause is a group of words consisting of a verb (or a verb group) and its arguments(explicit and implicit), and forms part of a sentence. Depending on the type of the verb, a clause is classified as a finite clause that contains a finite verb and non-finite clause that contains a non-finite verb. For example:

(1)

raam	khaanaa	khaakar	soyaa
Ram	food	having+eaten	sleep+past

'Having eaten food, Ram slept.'

In example (1), 'raam soyaa' is a finite clause; contains 'soyaa' finite verb, and 'khaanaa khaakar' is a non-finite clause; contains 'khaakar' non-finite verb.
We come across two types of clauses in a complex sentence:

1. Main clause, which is an independent clause, is also called Superordinate clause. It is always a finite clause in a sentence.

2. Subordinate clause, which is dependent on the main clause. It can be both finite and non-finite in a sentence.

For our task we have divided subordinate clause into five different types, which are complement clause, adverbial clause, relative clause, coordinate clause and non-finite clause (discussed shortly). Subordinate clauses can be nested or non-nested depending on a sentence. Nested here means one clause is embedded in another clause. 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 2) jo khela. The second clause is embedded in 'raam ghar gayaa'.
Various kinds of subordinate clauses are discussed below:

(a) **Complement Clause**: These clauses are introduced by the subordinator 'ki' (that) and generally follow the verb of main clause (Koul, 2009) and occur as a finite clause in a sentence. For example:

(3)

yaha	sach	hai	ki	mohan	bimaara	hai
It	true	is	that	Mohan	sick	is

'It is true that Mohan is sick'

'ki mohan bimaar hai' is a Complement Clause and 'ki' is a complementizer in example (3). It must be noted that 'complement clause' is also 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 that verb clause. We have handled this type of construction separately (discussed in section 4).

(b) **Relative Clause**: Relative clauses which are also finite in nature occur as a modifier of verb's argument and contain a relative pronoun (Koul, 2009). Such clause can be either nested or non-nested.

(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 'vo', the argument of the verb 'gayaa'.
Another example of this, is:

(5)

raam	ghar	gayaa	jo	khel rahaa thaa
Ram	home	go+past	who	play+past+conti

'Raam who was playing went home'

In example (5) relative clause ‘jo khel rahaa thaa’ is a non-nested one.

(c) **Adverbial Clause**: These clauses are determined based on their adverbial markers/function in a sentence (Koul, 2009). Manner, purpose, cause, condition etc. form the types of adverbial clauses. We take this type of clauses as the modifier of the verb’s modifier. These clauses are present as a finite clause in sentence. For example:

(6)	jaise	vaha	jaaegaa	waise	main	jaaungaa
	the way	he	go+fut.	that way	I	go+fut

‘I will go the way he will go’

In example (6) ‘jaise vaha jaaegaa’ is an Adverbial Clause with ’jaise’ as the (manner) Adverbial Marker. Here ‘waise’ is the modifier of the verb ‘jaaungaa’ and ‘jaise vaha jaaegaa‘ modifies it.
It may be noted that we consider clauses that are modifiers of verb’s modifiers as adverbial clauses and clauses that are modify arguments of verbs as relative clauses.

(d) **Coordinate Clause**: It is one of the independent finite clauses in a sentence that has the same status as the other clauses, and is introduced by a coordinating conjunction (Koul, 2009). For example:

(7)	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 (7). And for 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. When there are more than one coordinating conjunct in a sentence, clause boundary identification becomes more complex because of nesting of the coordinate clause. This is illustrated using example (8).

(8)	raam ne	kaam	kiyaa	aur	khaanaa	khaayaa	lekin	siitaa	khelii
	Ram+erg	work	do+past	and	food	eat+past	but	Sita	play+past

‘Ram did the work and ate food but Sita played’

In such examples there is more than one way to mark the coordinate clauses:

- ((raam ne kaam kiyaa) aur (khaanaa khaayaa)) lekin (siitaa khelii)
- (raam ne kaam kiyaa) aur ((khaanaa khaayaa) lekin (siitaa khelii))
- ((raam ne kaam kiyaa) aur (khaanaa khaayaa) lekin (siitaa khelii))

‘(’ and ‘)’ are symbols to denote the start and end of the clause. As we can see there is more than one output possible for the given example. Our system only marks the linear boundary of the clause in a sentence. Nesting in more than two coordinate clauses is not handled by it. So for the example (8), our output is: (raam ne kaam kiyaa) aur (khaanaa khaayaa) lekin (siitaa khelii)
--It must be noted that we do not take coordinating conjuncts as part of any of the clauses, it is conjoining. However subordinate marker are taken to be part of clause.

(e) **Non-finite Clause**: These clauses are dependent clause in a sentence which contain non-finite verb.

(9)	raam	khaanaa	khaakar	aur	paani	peekar	ghar	gayaaa
	Ram	food	eat	and	water	drink	home	go+past

’Raam after eating food and drinking water, went home’

In above example (9), two clauses, ‘khaanaa khaakar’ and ‘paani peekar’ are non-finite as they contain non-finite verbs.

--In Hindi, We come across some complex cases where one type of clause is embedded in another type clause. For example:

(10)	raam	jisne	khaanaa	khaayaa	aur	paani	piyaa,	ghar	gayaaa
	Ram	who	food	eat+past	and	water	drink+past	home	go+past

’Raam who ate food and drank water, went home’

In example (10) relative clause and coordinate clause overlap with each other. The coordinate clauses are: (jisne khaanaa khaayaa) and (paani piyaa), and relative clause is : (jisne khaanaa khaayaa aur paani piyaa). So our system will mark the clause boundaries as: (raam *((jisne khaanaa khaayaa) aur (paani piyaa))* ghar gayaaa).

3 Related works

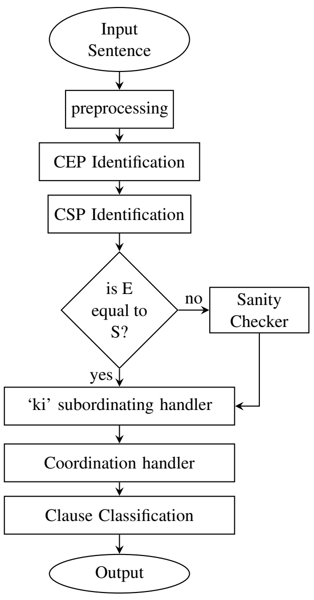
Studies in identifying clauses date back to (Ejerhed, 1988) work, where they showed how automatic clause boundary identification in discourse can benefit a parser's performance. However her experiments could detect only basic clauses. Later (Abney, 1990) used clause filter as part of his CASS parser. 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.
Our work is similar to that of (Leffa, 1998) in that both first mark clause boundaries and then classify the clauses into various types. Both use linguistic cues such as coordinating conjuncts, subordinating conjunction, surrounding context, however , while (Leffa, 1998) use POS information and valency of the verbs , we use POS tags and chunks as the only linguistic information.

4 Methodology

We propose a rule based system which first identifies the clause(s) in the input sentence and marks the *'clause start position'* (**CSP**) and *'clause end position'* (**CEP**) with brackets and then it classifies the identified clauses into one of the proposed types mentioned in section 2. 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 and the type of the clauses. Thus our system uses lists of coordinate conjunctions, relative markers and adverbial clause markers (see Appendix A and Appendix B for the lists). These lists were created using (Kachru, 2006). Further, the rules for our system have been framed based on our in depth analysis of a section of the Hindi treebank (Palmer et al., 2009). Apart from the lexical cues we have also used POS tag and chunk information to frame these rules.

4.1 Algorithm

Our system consists of two parts, the first part determines the boundaries of the clauses (clause identification) and the second part determines the type of the clause (clause classification). Identification of clause boundaries is further divided into two tasks, i.e. to find the beginnings and the ends of clauses. Then, the sentences with the clause boundaries marked are processed by the clause classification component, and are assigned to one of the clause types--main clause, complement clause, adverbial clause, relative clause, coordinate clause and non-finite clause. Figures 1 shows the data flow of our system, components of which have been discussed in detail, further in this section.



In this Data flow of our system, E represents number of *'clause end position'* and S represents number of *'clause start position'* marked by our system.

Figure 1: Data Flow

4.1.1 Preprocessing

In this module, input sentences are processed and each lexical item is assigned a POS tag, and chunk information . For example:
Input sentence:

(11)	raam	soyaa.
	Ram	sleep+past
	'Ram slept.'	

Output:

```
1    ((   NP
1.1  raam NNP
     ))
2    ((   VGF
2.1  soyaa VM
2.2  .     SYM
     ))
```

--Here 'NP' and 'VGF' are the chunk tags, and POS tags 'NNP' and 'VM' stand for Noun and Verb respectively (Bharati et al., 2007; Bharati et al., 2009) .

4.1.2 CEP Identification

The unmarked word order of Hindi mainly being SOV, the verb is taken to be the end of the clause. In cases where a sentence does not end with a verb , the end of sentence is taken as end of the clause. This helps to handle instances of scrambling and ellipses. For example:

(12)	siitaa	ghar	jaa rahii hai	aur	giitaa	bhii.
	Sita	home	go+present+cont	and	Gita	also
	'Sita is going home and so does Gita.'					

In example (12), there is an ellipses of the verb 'jaa rahii hai' in the second clause 'giitaa bhii'. In cases like this, our system marks the verb as end of the first clause and sentence end as end of the second clause. The marked boundaries in the sentence after this module will be: ‘siitaa ghar jaa rahii hai **)** aur gitaa bhi **)**’.

4.1.3 CSP Identification

We have made two modules to find the start of the clauses; one identifies the start of finite clauses and the other identifies the start of non-finite clauses. As we have mentioned a clause is a finite or non-finite, depending on the verb in that clause. So we have used chunk information which gives the verb type (finite or non-finite). Both these modules are independent of each other, so running them parallel will not affect the system, and this helps to speed up the system processing.

4.1.3.1 CSP for finite clause

This module uses linguistic cues such as relative markers (*jo* ‘that/who’, *jisane* ‘who’), coordinating conjuncts (*aur* ‘and’, *lekin* ‘but’) and so on, to identify the start of clauses. It may be noted that the immediate context of cues is also taken into account at times. For instance, a coordinating conjunct ‘aur’ (and) in a sentence marks the start of the clause only if it is preceded by a verb, whereas the subordinating conjunct ‘ki’ (that) always marks the start of a clause. After the start/s of clause/s in a sentence are identified, the module checks whether the beginning of the sentence is marked as a clause start, and marks it as clause beginning if it is not already marked. For example:

(13)	raam	jo	khel rahaa tha	nahii	aayaa.
	Ram	who	play+past+conti.	not	come+present

‘Ram who was playing did not come.’

In example (13), first our module identifies ‘jo’ relative marker and marks it as a start of the clause ‘jo khel rahaa tha’, and then, marks the beginning of the sentence as the start of the other clause ‘raam nahii aayaa’. After this, the boundaries marked in example (12) will be : **(** raam **(jo** khel rahaa tha) nahii aayaa.)
It needs a mention here that the boundaries marked in the previous module are also included in the current module’s output.

4.1.3.2 CSP for non-finite clause

Non-finite verbs do not have Tense-Aspect-Mood(TAM) information, they take optional arguments which are not specific in number. In Hindi, we don’t find any cues to detect where a non-finite clause starts. So to identify the start of a non-finite clause, we have built templates/regular expressions on chunks in a sentence, and whenever a pattern in a sentence matches the template, we mark that as a start of the clause. Following example shows the working of this module:

(14)	raam	ghar	para	jaakar	khaanaa	khaayega.
	Ram	home	to	after going	food	eat+future

‘After going to home, Ram will eat food.’

In the example (14), ‘ghar para’ and ‘raam’ are two separate chunks that precede the non-finite verb ‘jaakar’. As per the template, if a ‘para’ marked NP chunk follows the nominative NP and immediately precedes the ‘jaakar’ type non-finite verb, the NP chunk marks the start of the ‘jaakar’ non-finite clause.

4.1.4 Sanity Checker

In case the number of CSPs is not equal to the number of CEPs in a sentence, the Sanity Checker module comes into play. It iterates through the CSP identifier’s output for the sentence and marks the omitted CSPs. For example:

(15)	raam	ghar	gayaa,	shyaam	nahii	gayaa.
	Ram	home	go+past,	Shyam	not	go+past.

‘Ram went home, Shyam did not go.’

The absence of a coordinator between the two clauses ‘raam ghar gayaa’ and ‘shyaam nahii gayaa’, in Example (15) can lead to potential error of ommision of the CSP for the second clause ‘shyaam nahii gayaa’. The output of such a sentence would be:
‘(raam ghar gayaa) shyaam nahii gayaa.)’
As we can see here, the CSP for the clause ‘shyaam nahii gayaa’ is omitted. On detecting such an error, the sanity checker would iterate the sentence and mark the omitted CSP, and the output would then be:
‘(raam ghar gayaa) **(**shyaam nahii gayaa.)’

4.1.5 ‘ki’ complementizer handler

As mentioned earlier, ‘ki’ complement clause is an argument of the main verb and part of its main verb clause. Thus this modules executes, and identifies ‘ki’ complementizer and its clause in the sentence, and modifies the CEP of its parent clause. Example (16) explains this further.

(16)

raam ne	kahaa	ki	tum	ghar	jaao
ram+erg	say+past	that	you	home	go

‘Ram said that you go home.’

The input for the sentence ‘raam ne kahaa ki tum ghar jaao’ that this module receives would be:
‘(raam ne kahaa) (ki tum ghar jaao)’
The ‘ki’ complementizer module iterates this input and identifies the ‘ki’ complement clause and its CEP. It then modifies this input by moving the CEP, immediate before ‘ki’ complementizer to the position immediate after the CEP of ‘ki’ complement clause. The modified sentence will be:
‘(raam ne kahaa (ki tum ghar jaao))’

4.1.6 Coordination handler

This module handles embedded coordinated clauses in complex sentence where they fall within the scope of a complementizer, a relative marker or an adverbial marker. It makes a new CSP for these clauses immediately before the complementizer, relative marker or adverbial marker and a new CEP after the CEP of the last embedded coordinate clause. For example:

(17)

raam	jisne	khaanaa	khaayaa	aur	khel	khelaa	ghar	gayaa
Ram	who+rel.	food	eat+past	and	game	play+past	home	go+past

‘Ram who ate food and played a game, went home.’

Given the output for the example (17), this module identifies the ‘jisne’ the relative marker and inserts a new CSP immediately before it. It also inserts the CEP for their coordinate clauses after the CEP of the last embedded coordinate clause ‘khel khelaa’. The output would be:
(raam **(** (jisne khaanaa khaayaa) aur (khel khelaa) **)** ghar gayaa.)

4.1.7 Clause Classification

Once the clause boundaries are identified, the output is passed on to the clause classifier where it assign them to one of the clause classes--main clause, complement clause, adverbial clause, relative clause, coordinate clause and non-finite clause. If a sentence has only one clause, it is classified as the main clause. However given more than one clause in a sentence, it iterates the sentence and assign classes to the clauses based on cues such as relative markers, coordinating conjuncts etc. Verb type also helps to deduce whether a clause is non-finite or not. It then checks for potential omission and marks the omitted clauses as main clause, since they fail to fall under any of the other five classes.
In example (17) ,conjunction ‘aur’ helps to mark the two adjacent clauses--‘jisne khaanaa khaayaa’ and ‘khel khelaa’ as coordinate clauses. Relative marker ‘jisne’ helps to identify ‘jisne khaanaa khaayaa aur khel khelaa’ as a relative clause and the clause that remained ‘raam ghar gayaa’ is taken as main clause.

5 Evaluation and Results

As mentioned earlier identification of clause boundary for finite and non-finite clauses are independent, we have evaluated them separately. Finite clause mainly have 5 types; Main clause, Complement Clause, Adverbial Clause, Relative Clause and Coordinate clause and evaluation has been done on them.

5.1 Results for Finite Clause

A fresh set of 100 sentences average length of 16 words is randomly selected from a section of the Hindi treebank. This section is different from the section from which the sentences were chosen for analysis. The selected sentences have 217 clauses. An analysis of the category of these clauses is presented in Table 1. This evaluation set was annotated manually at the level of clause boundary and their type, to evaluate performance of the system. As mentioned earlier, five types of tags ; Main clause, Complement Clause, Adverbial Clause, Relative Clause and Coordinate clause, were used to annotate them.

Clause Type	%
Main Clause	33.79
Coordinate Clause	31.48
Complement Cl ause	24.07
Relative Clause	9.72
Adverbial Clause	0.9

Table 1: Clause distribution table.

5.1.1 Results of Clause Boundary Identification

For the evaluation of Clause Boundary identification, a clause is taken to be marked correctly iff its CSP and CEP are marked correctly. A sentence with more than one clause may have correctly marked clauses as well as incorrectly marked clauses. We evaluate the task at clause level, not at sentence level. The precision and Recall for clause boundary identification are **91.30%** and **91.78%** respectively.

5.1.2 Results of Clause Classification

For the evaluation of Clause Classification, we take a clause to be correctly classified if its boundaries as well as type is marked correctly. So, clauses with incorrectly marked boundaries are considered wrongly classified. The precision and Recall for clause classification are **80.28%** and **81.04%** respectively. Table (2) shows the results for different clause categories.

Clause Type	Precision%	Recall%	F1 score%
Main Clause	77.90	91.78	84.27
Coordiante Clasue	80.00	70.58	74.99
Complement Clause	92.30	92.30	92.30
Relative Clause	93.33	66.66	77.77
Adverbial Clause	100	50	66.66

Table 2: Results of Clause Classification

5.2 Results for Non-finite Clause

A set of 96 sentences containing 104 non-finite clauses was taken for the evaluation. It was found that end of all non-finite clause were identified but there were 63 clauses whose start boundary were identified. The accuracy of the system in identifying non-finite clauses is 60.57%.

6 Error Analysis and Discussion

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

   (18)

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 (18), 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:

   (19)

ghar	gayaa	raam,	vaha	bolaa.
home	go+past	Ram,	he	say+past

   'He said Ram went home'

In example (19), 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:

   (20) 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 (20) 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. Templates used for identification of non-finite clauses are not much efficient. They are more specific and need to be more general.

7 Conclusion and Future Work

We have discussed our work on clause boundary identification and classification 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. While a larger section of the Hindi dependency treebank from the HUTB project was analyzed to formulate the rules for the task. The system, showing a satisfactory performance for finite clauses in terms of F1 scores of 91.53% for clause boundary identification and 80.63% for clause Classification, while giving inadequate results for non-finite clauses with 60.57% accuracy. We would like to mention that at present our system doesn't handle classification of different instances of 'to' (else, then, or etc.) and of coordination where a punctuation serves as a coordinator. In the future we intend to incorporate this in our system. Further, 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, adopting an ML approach for this task 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.

Akshar Bharati, Rajeev Sangal, and Dipti M Sharma. 2007. Ssf: Shakti standard format guide. pages 1–25.

Akshara Bharati, Dipti Misra Sharma, Samar Husain, Lakshmi Bai, Rafiya Begam, and Rajeev Sangal. 2009. Anncorra: Treebanks for indian languages, guidelines for annotating hindi treebank.

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.

Yamuna Kachru. 2006. *Hindi*, volume 12. John Benjamins Publishing Company.

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

Vilson J Leffa. 1998. Clause processing in complex sentences. 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. 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.

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

Appendix A : Conjuction List

aur 'and'	athwaa 'or'	yaa 'or'	evam 'and'	para 'but'	magar 'but'
lekin 'but'	kintu 'but'	parantu 'but'	tathaa 'and'	jabki 'eventhough'	va 'and'
isalie 'therfore'	kyunki 'because'				

Appendix B : List of Relative (and Coorelative) Markers

jo 'who'	jiskaa 'whose'	jiske 'whose'	jiski 'whose'	jisko 'whose'
jisse 'from which'	jise 'who'	jinse 'from whom'	jinhen 'to whom'	jinhone 'who'
jinmen 'where'	jaba 'when'	jisse 'from which'	jise 'who'	