Major Project 2022

NLP - Hindi Text Analysis
&
Identifying Clause Boundary

Project Report

*s*ubmitted in partial fulfillment of the

*r*equirements for the award of

BACHELOR OF TECHNOLOGY

Submitted by

Aksh Sharma - 195019
Khushwant Kaswan - 195030
Khem Singh - 195038
Priyanshu Dhiman - 195049
Anupam Kumar - 195050
Astha Dad - 195076
Ravi Kant - 195115

Under the guidance of

Dr. Jyoti Srivastava

DEPARTMENT OF COMPUTER SCIENCE
NATIONAL INSTITUTE OF TECHNOLOGY
HAMIRPUR,HIMACHAL PRADESH

Declaration

We hereby declare that the research presented in this dissertation titled **"Hindi Text Analysis using NLP and Determining Clause Boundary"** in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology** and submitted in the Department of Computer Science and Engineering of the National Institute of Technology Hamirpur, is an record of our work carried out during the **VII Semester** under the guidance of **Jyoti Srivastava**, Assistant Professor
Department of Computer Science and Engineering, National Institute of Technology Hamirpur.

The matter presented in this report has not been submitted by us for the award of any other degree of this or any other Institute/University.

Aksh Sharma - 195019
Khushwant Kaswan - 195030
Khem Singh - 195038
Priyanshu Dhiman - 195049
Anupam Kumar - 195050
Astha Dad - 195076
Ravi Kant - 195115

This is to certify that the above statement made by the candidates is true to the best of my knowledge and belief.

Dr. Jyoti Srivastava

Date: 12 December 2022

Acknowledgement

We would like to start by expressing our sincerest gratitude to our major project supervisor **Dr. Jyoti Srivastava**, Assistant Professor, Department of Computer Science and Engineering at the National Institute of Technology Hamirpur, for her expertise, guidance, and enthusiastic involvement during our coursework.

We are highly obliged to faculty members of the Computer Science and Engineering Department because without their valuable insights and constructive opinions during evaluations, our project would not have yielded the significant results and led us to explore a myriad of use cases that we have put forward in this report.

We express our special thanks to our parents for their encouragement, constant moral Support, and to our friends and colleagues for being inquisitive and supportive during the course of this project.

Aksh Sharma - 195019
Khushwant Kaswan - 195030
Khem Singh - 195038
Priyanshu Dhiman - 195049
Anupam Kumar - 195050
Astha Dad - 195076
Ravi Kant - 195115

Abstract

In this project, we will demonstrate some of Natural Language Processing (NLP) techniques for Hindi Language and explored techniques to identify the clause boundaries present in compound sentence.

Clause boundary identification for compound sentences in any language is one of the basic necessity for processing of compound sentences. For grammar checking of compound sentences, it is necessary to identify the structure of various independent clauses present in compound sentence. Once the sentence is identified as compound sentence, the next step is to identify its pattern. After identification of patterns, various clauses present in the sentence are extracted as it is the basic step for performing grammar checking.

In this report, we has explored two techniques to identify the clause boundaries present in compound sentence.This may be be helpful in developing other Natural Language Processing (NLP) applications like simplification compound sentence in simple sentences, Improving Machine translation system and grammar checking of compound sentences. The problem of computing complex sentences in natural language processing is to make sentences simple to understand, by identifying clause boundaries as complex sentences are divided into clauses. Clause identification is very important task as many applications need simple sentences for computation.

This report is about the identification of clauses to some extent. The output will be a list of the clauses present in the sentence.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Clause

A clause is a largest unit of the sentence that contains a predicate and an explicit or implied subject. A sentence may have any number of clauses. Clause boundary identification means to split the sentence into clauses by identifying the starting and the ending position of the clause. The task of clause boundary identification is not only detecting a non-recursive phrase of the sentence, rather it is a three step process: identifying start of clause, identifying end of clause and finding complete clause

1.1.1 Types of Clause

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: "राम खाना खाकर सोया"

In above example,"राम सोया" is a finite clause; contains "सोया"' finite verb, and '"खाना खाकर"' is a non-finite clause; contains '"खाकर"' 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.

1.1.2 Types of Subordinate Clause

Subordinate clauses can be nested or non-nested depending on a sentence. Nested here means one clause is embedded in another clause.

Complement Clause

These clauses are introduced by the subordinator "कि" (that) and generally follow the verb of main clause and occur as a finite clause in a sentence.

For example: "यहाँ सच है कि मोहन बीमार है"

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

In above example, "कि मोहन बीमार है" is a Complement Clause and "कि" is a complementizer

Relative Clause

Relative clauses which are also finite in nature occur as a modifier of verb' s argument and contain a relative pronoun. Such clause can be either nested or non-nested.

For example: "वह लड़का जो खेल रहा था घर गया "

```
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 above example, relative clause is "जो खेल रहा था" with "जो" as a relative marker. "जो " modifies " वो ", the argument of the verb " गया ".

Adverbial Clause

These clauses are determined based on their adverbial markers/function in a sentence. 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: "जैसे वह जाएगा वैसे मैं जाउंगा"

```
 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 above example, "जैसे वह जाएगा" is an Adverbial Clause with "जैसे " as the (manner) Adverbial Marker. And "वैसे" is the modifier of the verb "जाउंगा" and "जैसे वह जाएगा" modifies it.

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.

For example: "मैं घर जाउंगा और राम दिल्ली जाएगा"

```
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'
```

In above example, "मैं घर जाउंगा" and "राम दिल्ली जाएगा" are two independent and coordinate clauses.

Non-finite Clause

These clauses are dependent clause in a sentence which contain non-finite verb.

For example : "राम खाना खाकर और पानी पीकर घर गया"

```
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, two clauses, "खाना खाकर" and "पानी पीकर" are non-finite as they contain non-finite verbs.

Chapter 2

Hindi Text Analysis

2.1 Introduction to NLP

Natural Language Processing (NLP) is a process of manipulating or understanding the text or speech by any software or machine. An analogy is that humans interact and understand each other ' s views and respond with the appropriate answer. In NLP, this interaction, understanding, and response are made by a computer instead of a human.

Natural Language	Computer Language
They are ambiguous in nature.	They are designed to unambiguous.
Natural languages employ lots of redundancy.	Formal languages are less redundant
Natural languages are made of idiom metaphor	Formal languages mean exactly what they want to say

Table 2.1: Natural Language vs. Computer Language

2.2 Basic NLP Techniques

There are various libraries available to do NLP tasks for English. Such is not the case with many other languages. We will be using **HaNLP**(A multilingual NLP library)

2.2.1 Tokenization

Tokenization is the first step in any NLP pipeline. It has an important effect on the rest of the pipeline. A tokenizer breaks unstructured data or natural language text into chunks of information that can be considered as discrete elements.
This immediately turns an unstructured string (text document) into a numerical data structure suitable for machine learning.

Tokenization can separate sentences, words, characters, or subwords. When we split the text into sentences, we call it **sentence tokenization**. For words, we call it **word tokenization**.

```
Input : "Lorem Ipsum is simply dummy text of the printing and typesetting industry."
Output :  {'tok': [['Lorem','Ipsum','is','simply','dummy','text','of','the',
                    'printing','and','typesetting','industry','.']]}
```

2.2.2 POS Tagging

POS Tagging (Parts of Speech Tagging) is a process to mark up the words in text format for a particular part of a speech based on its definition and context. It is responsible for text reading in a language and assigning some specific token (Parts of Speech) to each word. It is also called grammatical tagging.

```
Input : Even within India itself, there are a multitude of languages that are
        spoken and used in day to day life
Output : [['INTJ','ADV','ADP','PROPN','PRON','PUNCT','PRON','VERB','DET',
            'NOUN','ADP','NOUN','PRON','AUX',
            'VERB','CCONJ','VERB','ADP','NOUN','ADP','NOUN','NOUN']]
```

2.2.3 Chunking

Chunking in NLP is a process to take small pieces of information and group them into large units. The primary use of Chunking is making groups of "noun phrases." It is used to add structure to the sentence by following POS tagging combined with regular expressions. The resulted group of words are called "chunks." It is also called shallow parsing.

Chunking is used for entity detection. An entity is that part of the sentence by which machine get the value for any intention.

2.2.4 Dependency Parsing

Dependency parsing is the process of analyzing the grammatical structure of a sentence based on the dependencies between the words in a sentence.
In Dependency parsing, various tags represent the relationship between two words in a sentence. These tags are the dependency tags.
There exist many dependencies among words in a sentence and a dependency involves only two words in which one acts as the head and other acts as the child.

As of now, there are 37 dependency relations used in Universal Dependency.

If you start tracing the dependencies in a sentence you can reach the root word, no matter from which word you start.

A dependency parse connects words according to their relationships. Each vertex in the tree represents a word, child nodes are words that are dependent on the parent, and edges are labeled by the relationship.

```
Input : "It took me more than two hours to translate a few pages of English."
Output : 'dep': [[[2, 'expl'],[0, 'root'],[2, 'iobj'],[7, 'advmod'],[4, 'fixed'],
[7, 'nummod'],[2, 'obj'],[9, 'mark'],[2, 'csubj'],[12, 'det'],[12, 'amod'],
[9, 'obj'],[14, 'case'],[12, 'nmod'],[2, 'punct']]]}
```

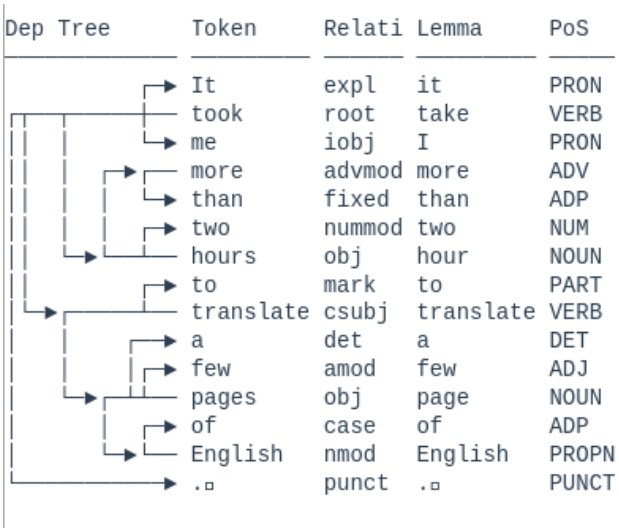


Figure 2.1: Dependency Parsing Example

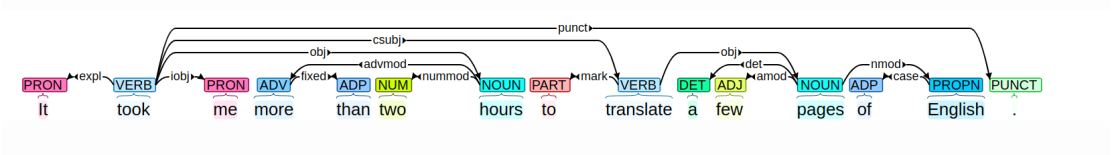


Figure 2.2: Dependency Parsing Representation

2.2.5 Constituency Parsing

A constituency parse tree breaks a text into sub-phrases. Non-terminals in the tree are types of phrases, the terminals are the words in the sentence, and the edges are unlabeled. A constituency parse tree always contains the words of the sentence as its terminal nodes. All the other non-terminal nodes represent the constituents of the sentence and are usually one of verb phrase, noun phrase, or prepositional phrase (PP).

```
Input : "It took me more than two hours to translate a few pages of English."
Output : (TOP(S(S(NP (PRON It))(VP(VERB took)(NP (PRON me))(NP (QP (ADV more)
        (ADP than) (NUM two)) (NOUN hours))(S(VP(PART to)(VP(VERB translate)(NP(NP (DET a)
        (ADJ few) (NOUN pages))(PP (ADP of) (NP (PROPN English)))))))))(PUNCT .)))
```

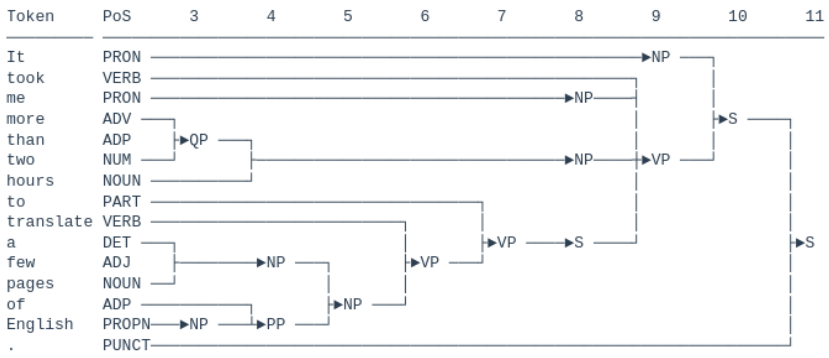


Figure 2.3: Constituency Parsing Example

2.2.6 Usecase of Dependency and Constituency Parsing

- Dependency parsing can be more useful for several downstream tasks like Information Extraction or Question Answering
- Another situation where using a dependency parser might prove advantageous is when working with free word order languages.
- As when we want to extract sub-phrases from the sentence, a constituency parser might be better.
- We can use both types of parse trees to extract features for a supervised machine learning model.

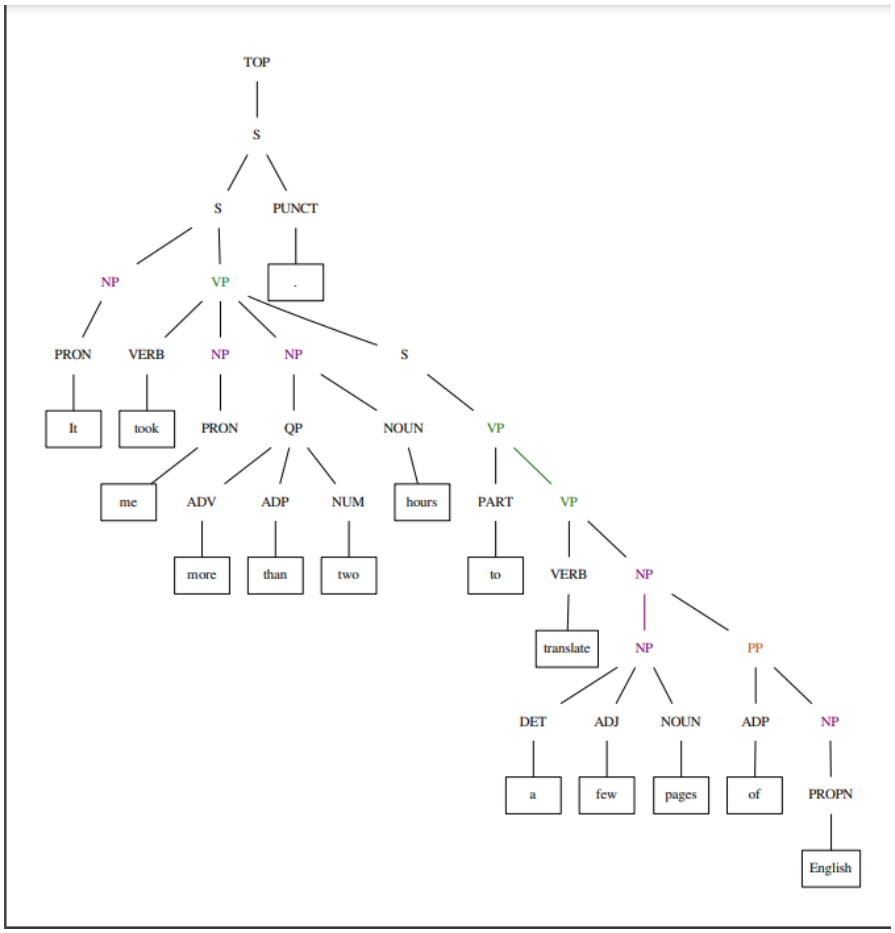


Figure 2.4: Constituency Parse Tree Representation

2.3 Practical Implementation on Hindi Sentences

Sentence : राम जिसने खाना खाया और खेल खेला घर गया

2.3.1 NLP Operations

Tokenization

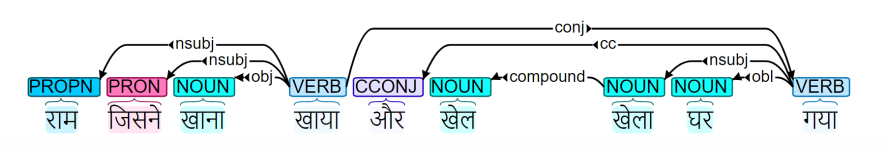
[['राम','जिसने','खाना','खाया','और','खेल','खेला','घर','गया']]

POS Tagging

```
[['PROPN', 'PRON', 'NOUN', 'VERB', 'CCONJ', 'NOUN', 'NOUN', 'NOUN', 'VERB']]
```

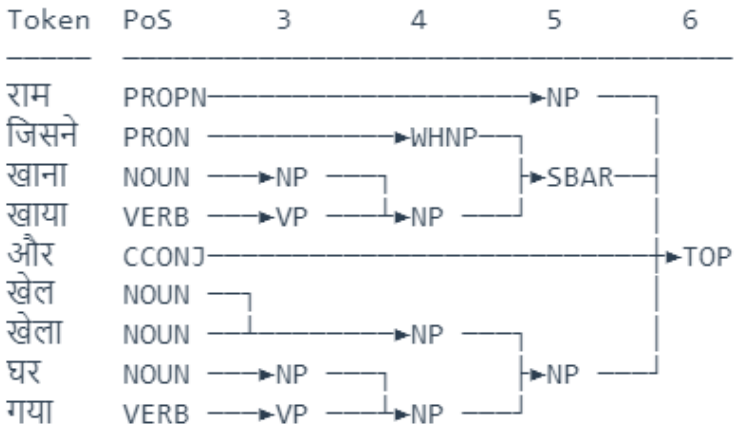
Dependency Parsing

```
'dep': [[[4, 'nsubj'],[4, 'nsubj'],[4, 'obj'],[0, 'root'],[9, 'cc'],
[7, 'compound'],[9, 'nsubj'],[9, 'obl'],[4, 'conj']]]}
```



Constituency Parsing

(TOP (NP (PROPN राम)) (SBAR (WHNP (PRON जिसने)) (NP (NP (NOUN खाना)) (VP (VERB खाया)))) (CCONJ और) (NP (NP (NOUN खेल) (NOUN खेला)) (NP (NP (NOUN घर)) (VP (VERB गया)))))



Chapter 3

Literature Review

Table 3.1: Literarture Review

Chapter 4

Methodology - Clause Boundary

4.1 Constituency Parse Approach

Constituency parsing is based on phrase structure grammar, which is the most relevant if you are seeking to extract clauses from a sentence.

4.1.1 Why Constituency Parse ?

Rule based approach using constituency parse tree is efficient over dependency tree because constituency-parse is based on phrase structure grammar, which is the most relevant if one is seeking to extract clauses from a sentence. It can be done using dependencies as well, but in that case, one will essentially be reconstructing the phrase structure – starting from the root and looking at dependent nodes.

4.1.2 Clause Levels

(This information is from "Bracketing Guidelines for Treebank II Style Penn Treebank")

S - simple declarative clause, i.e. one that is not introduced by a (possible empty) subordinating conjunction or a wh-word and that does not exhibit subject-verb inversion.

SBAR - Clause introduced by a (possibly empty) subordinating conjunction.

SBARQ - Direct question introduced by a wh-word or a wh-phrase. Indirect questions and relative clauses should be bracketed as SBAR, not SBARQ.

SINV - Inverted declarative sentence, i.e. one in which the subject follows the tensed verb or modal.

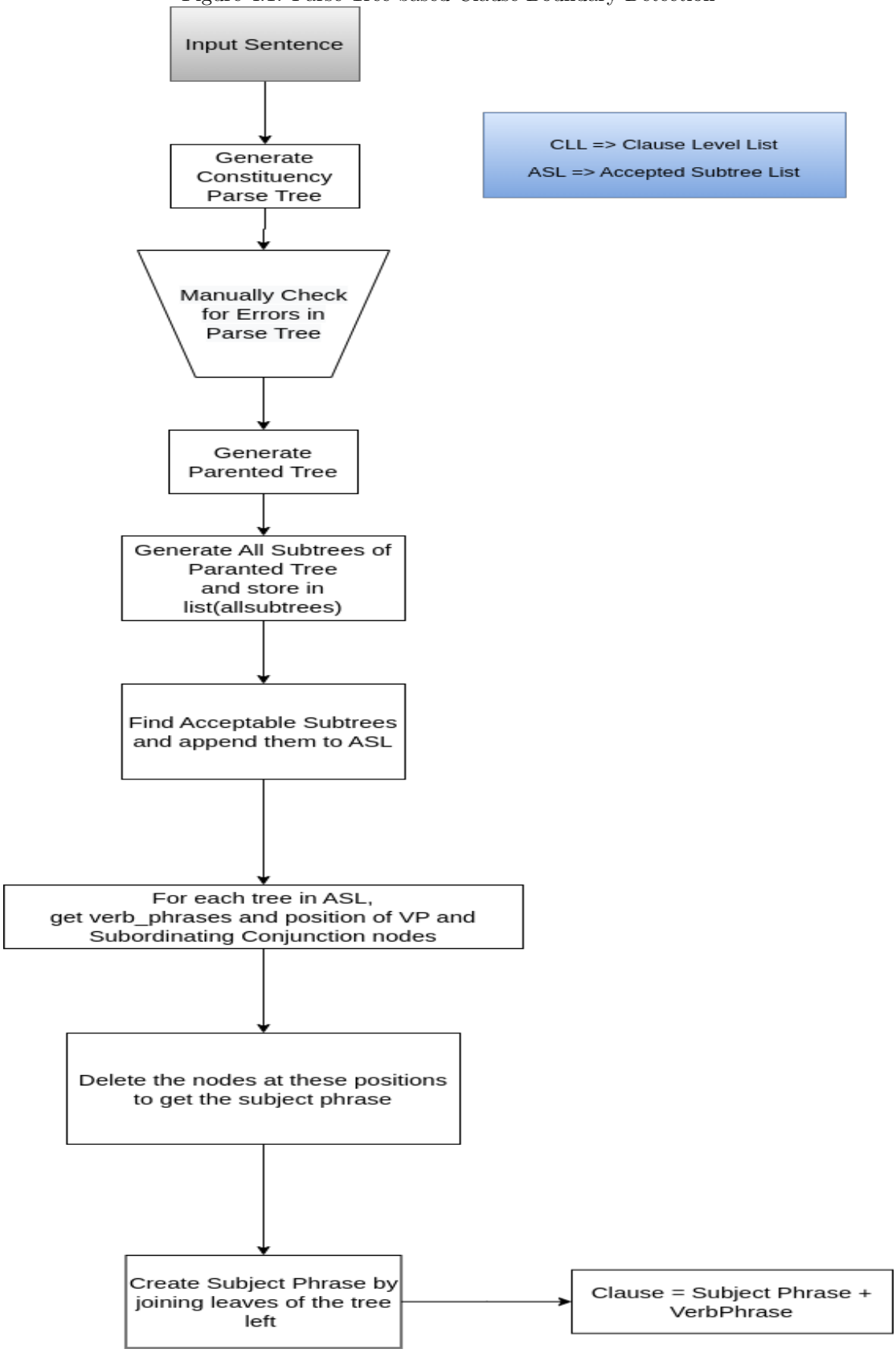
SQ - Inverted yes/no question, or main clause of a wh-question, following the wh-phrase in SBARQ.

4.1.3 Steps and Flowchart

The steps for extracting clauses and the data flow chart of the algorithm :

- Obtain the Constituency Parse tree structure of the given sentence and check manually for errors.
- Generate Parented Tree from the parse tree structure.
- Generate all subtrees of Parented tree and Identify the non-root clausal nodes .
- Remove, but retain separately the subtrees rooted at these clausal nodes from the main tree.
- In the main tree (after removal of subtrees), remove any hanging prepositions,subordinating conjunctions and adverb nodes . The portion left will be subject phrase.
- Combining all verb phrases with subject phrase will give us the clauses.

Figure 4.1: Parse Tree based Clause Boundary Detection



4.2 Rule Based Approach

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.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.
Apart from the lexical cues we have also used POS tag and chunk information to frame these rules.

4.2.1 Steps and Flowchart

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.

Preprocessing

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

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.

CSP Identification

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.

Sanity Checker

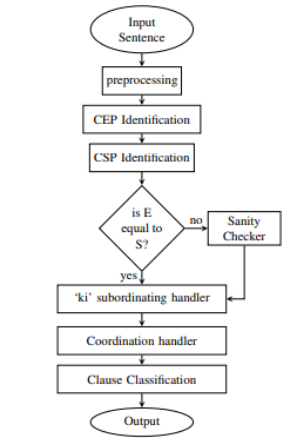
In case the number of CSPs is not equal to the number of CEPs in a sentence, the Sanity Checker module comes into play. It iterates through the CSP identifier' s output for the sentence and marks the omitted CSPs.

'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.

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.



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

Figure 4.2: Rule Based Clause Boundary Detection

Chapter 5

Implementation - Clause Boundary

5.1 Data Set Creation

In order to build a clause boundary identifier, using a 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.

We have created a dataset of 70 Hindi sentences from various resources and have successfully marked them with respective clause boundaries.

5.2 Constituency Parse Approach

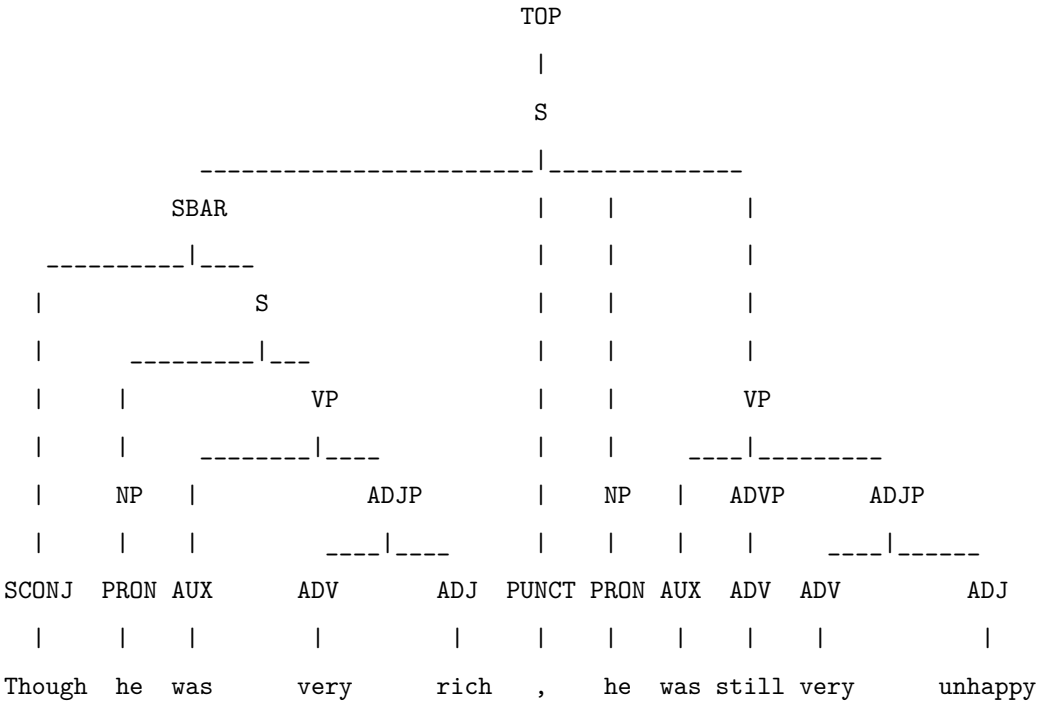
5.2.1 Analysis on Simple English Sentence

```
Sentence :  Though he was very rich , he was still very unhappy

POS Tags:  [('Though', 'SCONJ'), ('he', 'PRON'), ('was', 'AUX'), ('very', 'ADV'),
            ('rich', 'ADJ'), (',', 'PUNCT'), ('he', 'PRON'), ('was', 'AUX'), ('still', 'ADV'),
('very', 'ADV'), ('unhappy', 'ADJ')]

Parse Tree :   (TOP(S(SBAR(SCONJ Though)(S(NP (PRON he))(VP (AUX was)
               (ADJP (ADV very) (ADJ rich)))))(PUNCT ,)(NP (PRON he))(VP (AUX was)
               (ADVP (ADV still))(ADJP (ADV very) (ADJ unhappy)))))
```

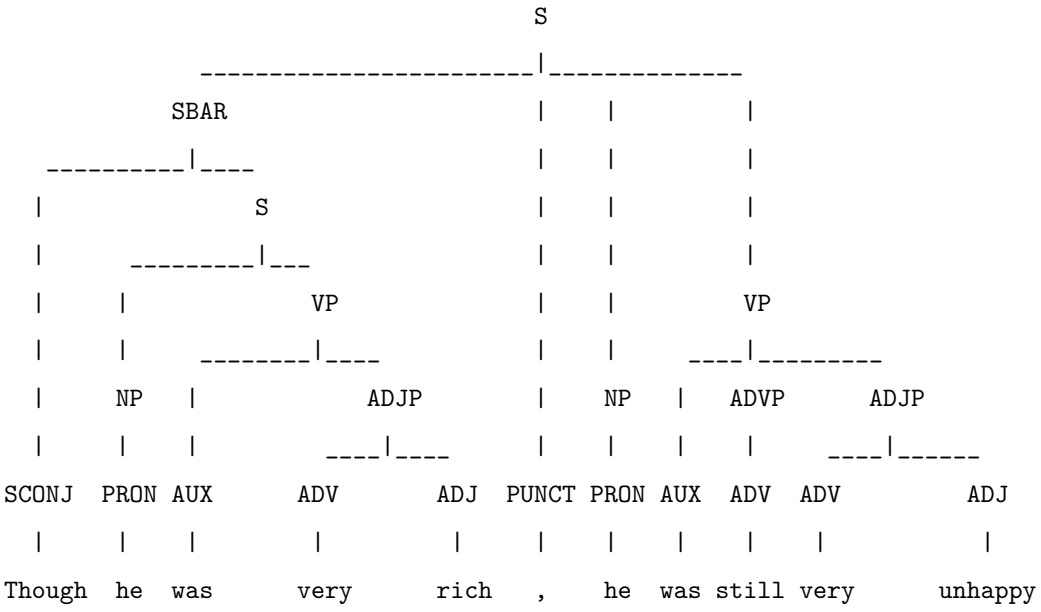
```
-----------------Parented Tree-----------------
(TOP
 (S
   (SBAR
     (SCONJ Though)
     (S (NP (PRON he)) (VP (AUX was) (ADJP (ADV very) (ADJ rich)))))
   (PUNCT ,)
   (NP (PRON he))
   (VP (AUX was) (ADVP (ADV still)) (ADJP (ADV very) (ADJ unhappy)))))
-------------------------------------------------
```




```
Number of Subtrees possible :  22
Number of Accepted Subtrees:  1

--------Accepted Subtree  1 ----------

 Leaf Nodes :  ['Though', 'he', 'was', 'very', 'rich', ',', 'he', 'was',
                'still', 'very', 'unhappy']
```




```
Verb Phrases :  ['was very rich', 'was still very unhappy']
Positions of VerbPhrase and Subordinationg Conjunctions :  ([(0, 1, 1)],
                                                [(0, 0), (1,), (2,), (3,)])
 Deleting Verb Phrase :   ['was', 'very', 'rich']
 Deleting Sub. Ordinating Conj Phrase :  ['was', 'still', 'very', 'unhappy']
 Deleting Sub. Ordinating Conj Phrase :  ['he']
 Deleting Sub. Ordinating Conj Phrase :  [',']
 Deleting Sub. Ordinating Conj Phrase :  ['Though']

 Tree Left After Deletion:
 S
 |
SBAR
 |
 S
 |
 NP
```

```
 |
PRON
 |
 he

 Subject Phrase :  he
--------Accepted Subtree  1  End----------

Clauses :  ['he was very rich', 'he was still very unhappy']
```

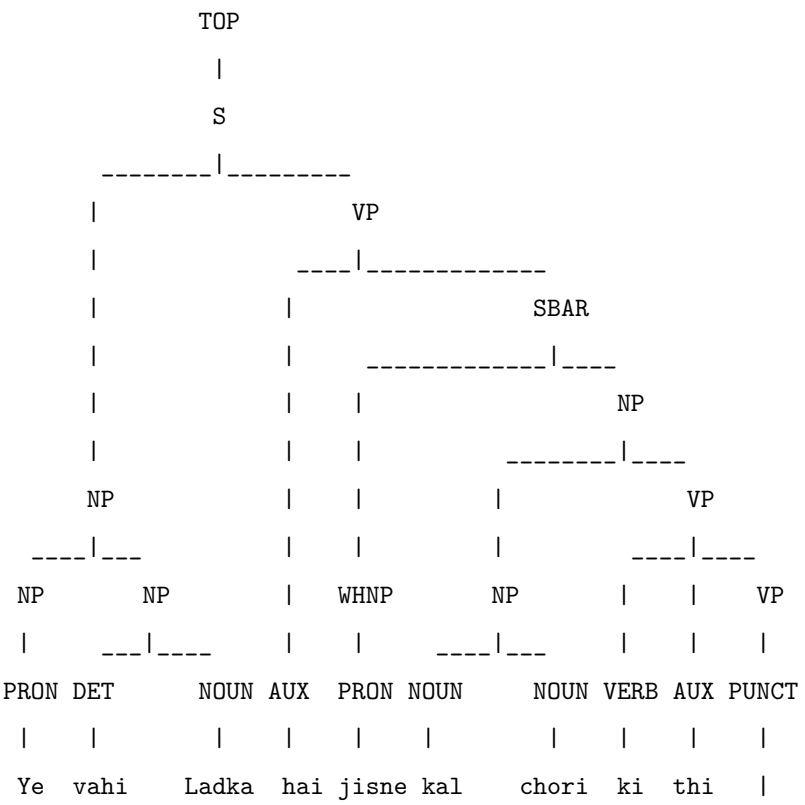
5.2.2 Analysis on Simple Hindi Sentence

Sentence 1

यह वही लड़का है जिसने कल चोरी की थी ।

(TOP(S(NP (NP (PRON यह)) (NP (DET वही) (NOUN लड़का)))(VP(AUX है)(SBAR(WHNP (PRON जिसने))(NP(NP (NOUN कल) (NOUN चोरी))(VP (VERB की) (AUX थी) (VP (PUNCT |))))))))

Parse Tree :



Clauses: 'जिसने कल चोरी की थी ।' , 'यह वही लड़का है'

Sentence 2

राम ने कहा कि तुम घर जाओ

Clauses : 'राम ने कहा', ' तुम घर जाओ'

5.2.3 Analysis on Compound English Sentence

Sentence 1

I looked for Ram and Shyam at the railway station , but they arrived at the station before noon and left on the train before I arrived .

```
Parse Tree :  (TOP(S(S(NP (PRON I))(VP(VERB looked)(PP (ADP for) (NP (PROPN Ram)
(CCONJ and) (PROPN Shyam)))(PP (ADP at) (NP (DET the) (NOUN railway) (NOUN station)))))
(PUNCT ,)(CCONJ but)(S(NP (PRON they))(VP(VP(VERB arrived)(PP (ADP at) (NP (DET the)
(NOUN station)))(PP (ADP before) (NP (NOUN noon))))(CCONJ and)(VP(VERB left)
(PP (ADP on) (NP (DET the) (NOUN train)))(SBAR (ADP before) (S (NP (PRON I))
(VP (VERB arrived)))))))(PUNCT .)))

Clauses :
'I arrived'
'I looked for Ram and Shyam at the railway station'
'they arrived at the station before noon'
'they left on the train'
```

Sentence 2

The police officer of Hamirpur District announced that alchol had declined 80 percent in market , whereas there had been a big jump in the number of cigarette

```
Parse Tree :  (TOP(S(NP(NP (DET The) (NOUN police) (NOUN officer))(PP (ADP of)
(NP (PROPN Hamirpur) (PROPN District))))(VP(VERB announced)(SBAR(SCONJ that)
(S(NP (PROPN alchol))(VP(AUX had)(VP(VERB declined)(NP (NUM 80) (NOUN percent))
(PP(ADP in)(NP(NP (PROPN market))(PUNCT ,)(SBAR(WHADVP (SCONJ whereas))
(S(NP (PRON there))(VP(AUX had)(VP(VERB been)(NP(NP (DET a) (ADJ big) (NOUN jump))
(PP(ADP in)(NP(NP (DET the) (NOUN number))(PP (ADP of)
(NP (NOUN cigarette)))))))))))))))))

Clauses :
'there had been a big jump in the number of cigarette'
'alchol had declined 80 percent in market ,'
'The police officer of Hamirpur District announced'
```

5.3 Rule Based Approach

5.3.1 Sentence 1:

राम ने कहा कि तुम घर जाओ.

Preprocessing

Word : [राम ,ने,कहा,कि,तुम,घर,जाओ]

POS : [NNP,PSP,VM,CC,PRP,NN,VM]

exp : [PROPN,ADP,VERB,SCONJ,PRON,NOUN,VERB]

CEP Identification

राम ने कहा) कि तुम घर जाओ).

CSP Identification

(राम ने कहा) (कि तुम घर जाओ).

'ki' complementizer handler

(राम ने कहा (कि तुम घर जाओ))

Clauses - (कि तुम घर जाओ), (राम ने कहा)

Chapter 6

Errors and Future Work

Errors and Caveats

- Even the best parsers will not always parse sentences correctly, so keep that in mind.
- Additionally, many complex sentences involve right node raising, which is almost never parsed correctly by most parsers.
- You may need to modify the algorithm a little if a clause is in passive voice.
- A detailed error analysis is the major sources of error include inaccurate POS tagging, inaacurate Constituency Parse Tree generation and the relatively limited coverage of the rules used to identify clauses, and an inability to discriminate between various subtypes of clause coordination.

Future Work

- Modifying the Parse Tree algorithm as sometimes it is not breaking the sentence at CCONJ.
- Modifying the algorithm for Passive Voice sentences.
- Identifying the type of clause extracted using cue phrases.
- Developing a Web app for Basic Hindi NLP Tasks and Clause Boundary using streamlit coordination.

References

[1] Sanjeev Kumar Sharma , "Clause Boundary Identification for Different Languages: A Survey" International Journal of Computer Applications Information TechnologyVol. 8, Issue II2016(ISSN: 2278-7720)

[2] Bogdan Sacaleanu, Dublin (IE); Alice Marascu, Dublin (IE); Charles Jochim, Dublin (IE)RULE-BASED SYNTACTIC APPROACH TO CLAIM BOUNDARY DETECTION IN COMPLEX SENTENCES

[3] C Poornima, V Dhanalakshmi, M Kumar Anand, P K Sonam, "Rule based Sentence Simplification for English to Tamil Machine Translation System." International Journal of Computer Applications (0975 –8887), Volume 25, No.8, July 2011

[4] KaalepHeiki-Jaan, MuischnekKadri, "Robust clause boundary identification for corpus annotation." In: Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC 2012), Istanbul, Turkey (2012).

[5] ParveenDaraksha, SanyalRatna, Ansari Afreen, "Clause Boundary Identification using Classifier and Clause Markers in Urdu Language." Polibits Research Journal on Computer Science, 43, pp. 61-65, 2011

[6] S Lakshmi, Ram Sundar Vijay, R and Sobha, Devi Lalitha, "Clause Boundary Identification for Malayalam Using CRF." Proceedings of the Workshop on Machine Translation and Parsing in Indian Languages (MTPIL-2012), pages 83–92, COLING 2012, Mumbai, December 2012.

[7] Sharma Rahul, Paul Soma, "A hybrid approach for automatic clause boundary identification in Hindi." 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.

[8] http://en.wikipedia.org/wiki/Clause

[9] Narula, R., Sharma, S. K. 2014. Identification and Separation of Simple, Compound and Complex Sentences in Punjabi Language. International Journal of Computer Applications Information Technology. Vol. 6, Issue II Aug- September 2014.

[10] Sobha, L. D., Lakshmi, S. Malayalam. 2013. Clause Boundary Identifier: Annotation and Evaluation. WSSANLP2013, p. 83

[11] S. K. Sharma , 2017 .Marking Clause Boundary in Compound Sentences of Punjabi Language ,International Journal of Computer Science and Engineering.Volume-5, Issue-9

[12] Aniruddha Ghosh, Amitava Das and Sivaji Bandyopadhyay. 2010. Clause Identification and Classification in Bengali. In Proceedings of the 1st Workshop on South and Southeast Asian Natural Language Processing (WSSANLP, 23rd International Conference on Computational Linguistics (COLING)

[13] Vijay Sundar Ram,R T. Bakiyavathi and Sobha. L.2009. Tamil Clause Identifier. PIMT Journal of Research. Patiala. Vol.2. No.1, pp. 42-46.