ISSN:2229-6093

Hindi Parser-based on CKY algorithm

Nitin Hambir
Assistant Professor, CSE Deptt.
Acropolis Institute of Technology and Research,Indore, India
nitin.hambir@gmail.com

Ambrish Srivastav
Assistant Professor, CSE deptt.
Swami Vivekanand College of Engineering,Indore, India
a.srivastav30@gmail.com

Abstract

Hindi parser is a tool which takes Hindi sentence and verifies whether or not given Hindi sentence is correct according to Hindi language grammar. Parsing is important for Natural Language Processing tools. Hindi parser uses the CKY (Coke- Kasami-Younger) parsing algorithm for Parsing of Hindi language. It parses whole sentence and generate a matrix.

Keywords: *Parser, CKY, Compiler, Context free grammar.*

1. Introduction

Parsing of Hindi in particular is not a very widely explored territory. There have been many attempts at developing a good Parser for Hindi. Parsing is the process of analyzing an input sequence in order to determine its grammatical structure with respect to a given formal grammar.

Hindi parser is Efficient parser that parses Hindi language sentence. It parses Hindi language with respect to a formal grammar which I have been defining. It parses Hindi language at sentence level. It uses Cocke-Kasami-Younger (CKY) Parsing algorithm to parse sentence. CKY Parsing algorithm is a dynamic programming approach to parse context free grammar (CFG).

Its worst case time complexity is- Θ (n^3) where n is the length of the string to be parsed. The standard version of CKY can only recognize languages defined by context-free grammars in Chomsky Normal Form (CNF)[1].

A CFG is a 4-tupple G = (V, Σ, S, P), where V is a finite set of non-terminals and Σ is a finite set of terminal symbols, P is a finite set of production rules of the form A => α with A є V, α є (V Ụ Σ)* and S є V is a starting symbol.

CNF is defined as a form having productions of the type either A => BC or A => a, where A, B and C are non-terminal symbols and a is a terminal symbol.

1.1 The Need

Language is source of communication. Human is an intelligent creature that has come up with so many languages around the globe in order to express his feelings, to communicate his necessities and to offer his services. India is a multilingual country with as many as 22 scheduled languages and computer technology breaks the language barrier and bridges the gap between the various sections of the society through easier access to information using their respective languages and hence language computing becomes central to the exchange of information across speakers of various languages. The need for computers to understand natural language is growing by the day as human-computer interfaces become more intuitive, and machine translation gains prominence due to increasing multilingual content on the net. Some other applications of Hindi Parser are Natural Language Interface to Computer, Database, Question-Answering System and Machine Translation[2].

2. System Description

For parsing the Hindi language there is a requirement for developing a parsing mechanism. The parsing mechanism requires an interface for taking Hindi sentence from user and database from where tags can be searched and core parser which parse sentence. Thus, the important components of the system are:

- Interface
- Database for hindi word.
- Parser

Interface provides a graphical user interface to user for entering hindi sentence easily. Procedure to enable use of Unicode to work in Hindi on computers having Windows 2000 or later version Operating Systems written in appendix. Interface also tokenize input sentence and assign tag to each token[4].

Database stores the tag of hindi language word. Interface Searches tag for generated Unicode string of token.

Parser takes string of tag as input and states whether or not input string is correct. To parse string parser use formal hindi grammar rules that I have written in proposed solution and CKY algorithm. Parser generates a matrix instead of parse tree.

2.1 Grammar

CKY Parser use a handful of grammar rules that define the whole language. We have used some rules to cover the assertive sentences of Hindi languages and based on these rules we apply parsing and build the tree structure of the sentences. These are some rules we have found out for some sentences. There might be a need to add some more rules for interrogative sentences, sentences that have prepositions. Our Hindi formal grammar has 14 non-terminals and 13 terminals. The following table 2.1 shows non terminals.

N	**Noun**
M	**Masculine Noun**
F	**Feminine noun**
X	**Plural Noun**
V	**Verb**
W	**Male Verb Form**
U	**Female Verb Form**
B	**Plural Verb Form**
P	**Pronoun**
A	**Adjective**
D	**Adverb**
K	**Karak**
G	**Intermediate Verb**
E	**Intermediate Verb**

Table 2.1: Abbreviation for non terminals.

2.2 System Architecture

The system architecture, as shown in figure 2.2, has the following stages through which the source text is passed.

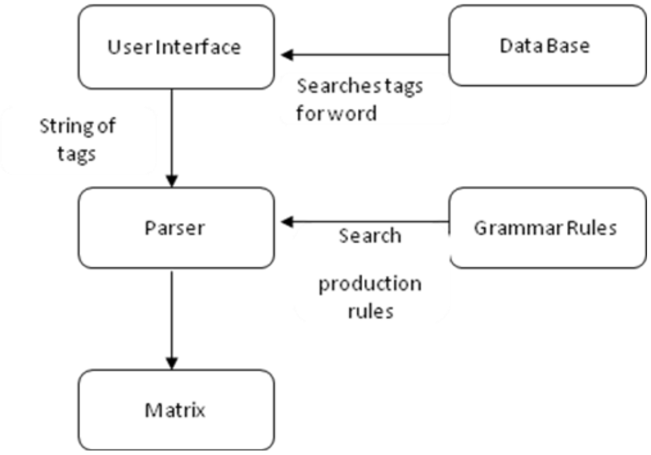

Fig. 2.2: System Architecture of Hindi Parser.

3. Result

We implemented the CKY algorithms, as well as the interface mentioned in section 2[5]. The code is written in Java and executed on the windows machines, which have dual core Intel 3.0 GHz processors and 2 GB of memory.

Our assumption in this experiment was that CKY algorithm would perform significantly better in all experiments. We felt that CKY could perform better in the case of strong database having more words. It could give better result for sentence of 4-5 words[5].

4. Conclusion

We have discussed the CKY algorithm could parse sentence of Hindi language. To cover Hindi language we have to improve size and structure of our database and grammar. In general, an increase in the grammar size allows better syntax analysis and improves the average accuracy of the parse tree. In CKY implementation above, iteration over the entire space of the grammar is nested within the inner n^3 loop and has a very large impact on the run time of the algorithm.

A large database would slow speed of parsing and also introduce word sense ambiguity in assigning tag to words of input sentence.

5. References

[1] Natural Language Processing- A Paninian Perspective" by Akshar Bharati, Vineet Chaitanya, Rajeev Sangal, IIT-K.
[2] CYK Algorithm, Wikipedia.
[3] "Technology Development for Indian Languages, Link- http://ildc.in/
[4] Captions Language Interface Pack (CLIP), Link: http://www.bhashaindia.com/Pages/Home.aspx
[5] Unicode for Hindi Language, Link: http://unicode.org.
[6] Bharati, Akshar,Vineet Chaitanya, and Sangal R., Tree Adjoining Grammar and Paninian Grammar Technical Report TRCS-94-219, Dept. of CSE, IIT Kanpur, March 1994b.
[7] Bharati, Akshar, Vineet Chaitanya, and Rajeev Sangal, A Computational Grammar for Indian Languages Processing, Indian Linguistics journal, 52(1{4):91103, Mar.Dec. 1991a.
[8] Carlos Gomez-Rodriguez, Miguel A. Alonso and Manuel Vilares. On Theoretical and Practical Complexity of TAG Parsers.
[9] Earley, Jay, An Efficient Context-Free Parsing Algorithm, Communications of ACM 6, 8, 1970, pp.451-455.
[10] Jurafsky D. and Martin J.H. , "Speech and Language Processing"; Prentice Hall; 2000.