International Journal of Advanced Computer Research
ISSN (Print): 2249-7277 ISSN (Online): 2277-7970
Volume-5 Issue-19 June-2015

Tree Formation Using Coordinate Method

Monika Choudhary[1*], Nirja Shukla[1*], Shashi Pal Singh[2], Ajai Kumar[2] and Hemant Darbari[2]
Banasthali Vidyapith, Banasthali, India[1]
AAIG, Center for development of Advanced Computing, Pune, India[2]

Received: 06-April-2015; Revised: 03-June-2015; Accepted: 06-June-2015

Abstract

In this paper we are introducing a new method of tree formation, we propose a coordinate based method by which we can store and access tree structures. As we know in NLP, parsing is the most important module. The output of this module is generally parsed trees. Currently, TAG (Tree Adjoining Grammar) is widely used grammar due to its linguistic and formal nature. It is simply tree generating system. The unit structure used in TAG is structured trees. So we used our new method to store trees where we worked on English to Hindi language. We worked on different sentences from English to Hindi, our method is the easiest way to manipulate tree. We have implemented within small corpus and for finite number of structures and further can be extended in future.

Keywords

TAG (Tree Adjoining Grammar), NLP (Natural Language Processing), MCSLs (Mildly context-sensitive Languages), LISP (List Processing), CFLs (Context free languages), Initial Trees, Auxiliary trees, Substitution, Adjoining, LTAG (Lexicalized Tree-Adjoining Grammar), RDBMS, java 2D API, POS tag.

1. Introduction

Natural languages are not context free grammar. Context-free grammars, are simple and very efficient formalism, which are used for restricted languages (such as programming languages), but are insufficient for human languages. Also the other formalisms are enough cantered. They cannot be restricted to some target linguistic phenomena and generate languages which are much more complex than human languages.

*Authors for correspondence

Mildly context-sensitive Languages (MCSLs) is introduced by Joshi 1985, is the smallest possible language class which include all linguistics phenomena. Mildly context sensitive grammars are derived to model the natural language structures restricted in their formal nature. The properties of MCSLs are1) CFLs are strictly included in MSCLs .2) Languages in MSCL are analysed in polynomial time.3) MCSGs capture only certain kind of dependencies.4) Languages in MCSLs have constant growth property [1][2][3].

TAG comes under mildly context sensitive grammar. It is a tree re-writing system rather than string re-writing system. We will be discussing TAG in section 2 There are various methods by which tree can be formed like using LISP notations etc. but our method is simplest way to implement the Tree Adjoining Grammar which we call as coordinate based method for the formation of tree by using Mysql as database and high level language i.e. java using net beans platform, we will discuss it in section 3 with example. In section 4 we will show our experimental result. Our conclusion is discussed in section 5. Future enhancements are described in section 6.

2. Tree Adjoining Grammar

TAGs were introduced by Joshi, Levy and Takahashi (1975) and Joshi (1985).It is motivated due to its formal and linguistic relevance. It has two most important properties 1) Extended domain of locality and [11] 2) factoring recursion due to its domain of locality [7].

It consist of quintuple (**T, N, I, A, S**) where. **T** is a finite set of terminal symbols, **N** is a finite set of non-terminal symbols, **I** is a finite set of Initial tress, **A** is a finite set of Auxiliary trees. There are mainly two types of operation is performed.

It contains finite set of elementary trees, i.e. , **I U A. I** are described as in this interior nodes contain non terminal symbols and frontier nodes contain terminal or non- terminal symbols. Substitution operation is performed in Initial trees and (↓) down arrow notation is used for substitution.

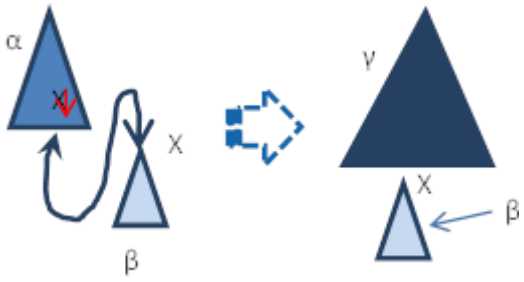

Figure 1: Substitution

A is described as in this interior nodes contain non-terminal symbols and frontier nodes contain terminal or terminal symbols. Adjoining operation is performed in Auxiliary trees. Non terminal symbol on frontier node must have at least one node and the label of which is same as foot node. We use asterisk (*) for Adjoining operation [4][5][8][9].

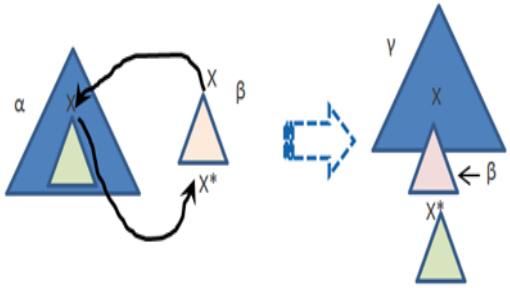

Figure 2:Adjoining

LTAG (schabes 1988) is extended form of TAG. Lexicalized Tree-Adjoining Grammar (LTAG) is also tree-generating system. LTAG is also Mildly Context Sensitive Grammar (MCSG). The elementary units of rewriting in this Grammar formalism are basic syntactic structures, as elementary trees as in TAG. These elementary trees are combined by substitution and adjunction operations to form derived/derivation trees; only difference is that each elementary tree is associated with a lexical item, called the anchor of the tree. So in this we get rich information about structures due to anchor also it improves the performance of parsing [6] [7].

As discussed above both the grammar formalism needs to store elementary trees as its basic structure. There is no properly standard documented method is available. Some of the commonly used methods are like quasi tree where it is used to represent partial description trees [10].

There are some notations to store these trees but all this is not easy to understand. In this paper, simple coordinate method is used in which coordinates can set and can draw lines and strings by using graphics it form trees.

3. Our Methodology

3.1 Technology Used

Here, in this paper new methodology is used to store trees structure by setting coordinate, for this we will be using MySQL. Mysql is a fastest and easy to use RDBMS for many applications. Different structures are used to store English and its corresponding Hindi in a tree form by declaring its coordinate in a static time. Tree formation is implemented by using java 2D API. It uses *java*.awt.*graphics* class. It provides the capabilities of drawing on a screen along with various inbuilt methods for various functions. By graphics we implemented the tree formation, to form a tree we used various method to draw lines and strings and further set coordinates to draw a tree in the java frame.

3.2 Explanation with an example

We have our hand-made bilingual corpus. In this corpus we have English to Hindi translated lexicons and POS tagging of English lexical items are done and stored in a DB table. In this paper displaying of simple English tree structure for input sentence and its corresponding Hindi tree structure.

Now when we input a sentence to the system it fetches the corresponding POS tags from corpus and if any word is not available in corpus than nothing will be fetched but we can manually add the word into the corpus.

Example: This is a simple sentence
POS match found for word: THIS=DET
POS match found for word: IS=VERB
POS match found for word: A=DET
POS match found for word: SIMPLE=NOUN
POS match found for word: SENTENCE=NOUN

As in above example we can see that the POS Tag has fetched for each word. As verb is the heart of the sentence so, we break down the sentence from first VERB in the sentence into two parts as to separate the noun phrase and verb phrase from the sentence. Now we have created rules for noun phrase and verb phrase. Each of different structure will have a unique rule and is identified by its id.

Rule 1

THIS
DET

Rule 2

IS	A	SIMPLE	SENTENCE
VERB	DET	NOUN	NOUN

Rule table

ID	CHILD1	CHILD2	CHILD3	CHILD4	CHILD4
1	NP	DET			
2	VP	VERB	DET	NOUN	NOUN

Here, we get the corresponding rules for NP and VP of input sentence form a rule table. In rule table we can see that the noun phrase and verb phrase have different id for different-different phrase structure. We store these ID's in the linked list for further fetching coordinate table. As we have made individual coordinate tables for each unique structure alike it's ID.

Here, we have used small letter alphabet for sequencing of words.

THIS
DET
a

a is temporary string variable and also defined in coordinate table of the above unique rule and replaces during tree formation.

IS	A	SIMPLE	SENTENCE
VERB	DET	NOUN	NOUN
a	b	c	d

a, b, c, d are temporary variables used which are defined in coordinate table of the above unique rule, further these a, b, c, d are replaced by its corresponding words.

Table 1: Coordinate table for rule_1

string	X1	Y1	X2	Y2
NP	455	175	650	70
DET	455	280	455	175
a	455	350	455	280

Table 2: Coordinate table for rule_2

string	X1	Y1	X2	Y2
S	650	70		
NP	455	175	650	70
VP	845	175	650	70
VERB	715	280	845	175
NP	845	280	845	175
NP	975	280	845	175
DET	780	350	845	280
NOUN	865	350	845	280
NOUN	960	350	975	280
a	715	350	715	280
b	780	420	780	350
c	865	420	865	350
d	960	420	960	350

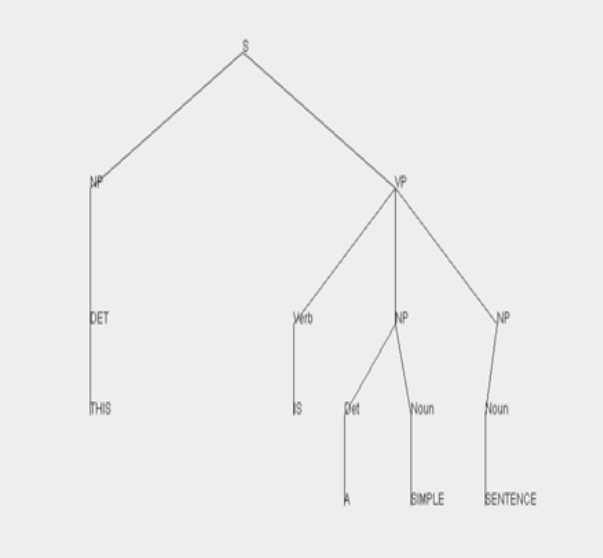

Figure 3: English parsed Tree

This is the output of our English input sentence. Likewise we can form parsed tree for different sentences. We can see here the NP and VP are joined by the S and form a tree.

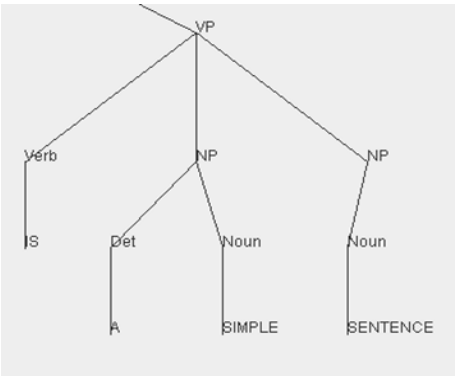

Figure 4: Rule 1

Figure 5: Rule 2

Figure 4, verb phrase and Figure 5 is noun phrase and combined to form tree in Figure 3.

Similarly, Hindi tree can be formed by rearranging the rule. For Hindi we have rearranged coordinates of the structure and temporary string variables in the Hindi tree which are replaced by corresponding Hindi lexicons from corpus.

ENGLISH	THIS	IS	A	SIMPLE	SENTENCE
HINDI	यह	है	एक	सामान्य	बाक्य

This is simple one to one mapping in English to Hindi now we have to rearrange the lexical items to form a correct Hindi tree.

यह	है	एक	सामान्य
VERB	DET	NOUN	NOUN

एक	सामान्य	बाक्य	है
DET	NOUN	NOUN	VERB

As we can see that by rearranging verb comes to end so instead of rearrange the rule and making separate table we will rearrange the coordinates of lexical items in a tree. Similarly many rules can be manipulated.

Temporary string variables are used for sequencing of words in a sentence and these are also store in a coordinate table for each unique structure or rule so now for Hindi tree structure we will also have unique tree structure for every corresponding English tree structure. So we will rearrange these temporary string variables in every coordinate table of Hindi Trees. Similarly we will get the correct Hindi translated tree for input sentence.

Table 3: Coordinate table of rule_hin_1

String	X1	Y1	X2	Y2
NP	455	175	650	70
DET	455	280	455	175
A	455	350	455	280

Table 3 contain the coordinates for NP of Hindi tree and similarly, Table 4 contains coordinates for VP of Hindi tree. So combining we will form corresponding translated Hindi tree for the input sentence.

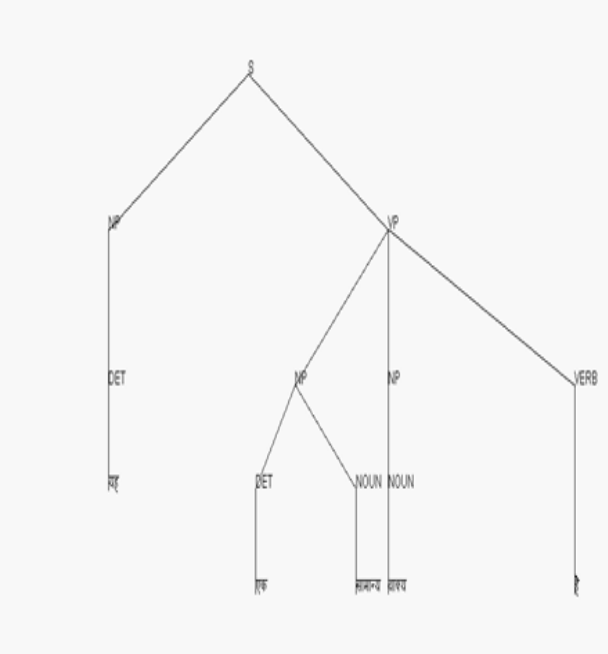

Figure 6: Hindi parsed Tree

Table 4: Coordinate table of rule_hin_2

String	X1	Y1	X2	Y2
S	650	70		
NP	455	175	650	70
VP	845	175	650	70
VERB	1105	280	845	175
NP	715	280	845	175
NP	845	280	845	175
DET	660	350	715	280
NOUN	800	350	715	280
NOUN	845	350	845	280
A	1105	420	1105	280
B	660	420	660	350
C	800	420	800	350
D	845	420	845	350

In the above example when particular rule is matched than its corresponding table is fetched and tree is formed, but if rule is not present in database then we will not get its corresponding tree. So this is the drawback of this system that we have to make rules and their respective coordinate tables for each and every possible structure.

For example 2

EXAMPLE: He speaks English better than I do

HE	SPEAKS	ENGLISH	BETTER	THAN	I	DO
NOUN	VERB	NOUN	ADV	PREP	NOUN	VERB

EXAMPLE: He speaks English better than I do

POS match found for word: HE=NOUN
POS match found for word: SPEAKS=VERB
POS match found for word: ENGLISH=NOUN
POS match found for word: BETTER=ADV
POS match found for word: THAN=PREP
POS match found for word: I=NOUN
POS match found for word: DO=VERB

Rule 3

HE
NOUN

Rule 4

SPEAK	ENGLIS	BETTE	THA	I	DO
S	H	R	N		
VERB	NOUN	ADV	PREP	NOUN	VERB

Rule Table

ID	PARENT	CHILD1	CHILD2	CHILD3	CHILD4	CHILD5
3	NOUN					

As in the above example rule 4 is not available in the rule table also its coordinate table for this structure also not present. So there is no tree formation for this sentence.

4. Experimental Results

We have worked on a small corpus and limited number of unique structures and rules. We worked on total 300 trees including both English and Hindi trees.

We worked on both simple sentences and complex sentences but our system works good for simple sentences. As complexity and length of sentences increase manually workload increases in both setting the coordinates and range of the coordinates are limited. This is only a static method as there is no dynamic tree formation is done.

It can be used for only small number of structures or small grammar. As far as structure increases complexity of managing and creating Database increases. It will degrade the system performance.

Table 5: work done on corpus

Words	13461
RULES	174
English and Hindi Tree	248
Sentences	800

5. Conclusion

In this paper it is shown that coordinate method is easiest method to form tree and store tree. It does not need any notation to form a tree. We have to simply set the coordinates which is easiest way for beginners to form a tree. We worked in English to Hindi language.

Moreover, this is static method so can be used for other languages for future as it only replaces words as per the rule fetched and need different corpus for other languages. So it is not specified to English and Hindi, it can be used by other languages too Since neither of these advantages is bound to this particular model, we hope to continue this work by exploring the usefulness of TAG under this approach by 3D view.

6. Future Enhancement

This can be extended to dynamic coordinate system where we can set coordinate once and further can increment both x-axis and y-axis according to requirement dynamically.

We can also improve our work by using 3D graphics where we can display trees in three-Dimensional. As we know in TAG or LTAG when it comes to non-projective and cross dependencies for a long and complex sentences tree view must be easily recognizable. 3D representation will provide more clear view of tree structures and also show rotational view of the tree which helps us to study deeper relations of the trees.

References

[1] Joshi, Aravind K. "Mildly Context-Sensitive Grammars." Oxford International Encyclopedia of Linguistics, 2nd Edition. Oxford University Press, Oxford, UK (2003).
[2] David J Weir " Characterizing Mildly Context Sensitive Grammar Formalisms" Department of Computer and Information Science, School of Engineering and Applied science, University of Pennsylvania, Philadelphia, issue : September (1988).
[3] Mery, Bruno, et al. "A case study of the convergence of mildly context-sensitive formalisms for natural language syntax: from minimalist grammars to multiple context-free grammars." (2006): 67.
[4] Aravind K. Joshi and Yves Schabes "Tree-Adjoining Grammars and Lexicalized Grammars" Department of Computer & Information Science, University of Pennsylvania Scholarly Commons, issue: March (1991).
[5] Joshi, Aravind K., and Yves Schabes. "Tree-adjoining grammars." Handbook of formal languages. Springer Berlin Heidelberg, 1997. 69-123.
[6] Ali Basirat,"Linking LTAG with LTIG" Department of Linguistics and Philology, Uppsala University, issue: 20 October (2014).
[7] Joshi, Aravind K. "Factoring recursion and dependencies: an aspect of tree adjoining grammars (TAG) and a comparison of some formal properties of TAGs, GPSGs, PLGs, and LPGs." Proceedings of the 21st annual meeting on Association for Computational Linguistics. Association for Computational Linguistics, 1983.
[8] Joshi, Aravind K., and Yves Schabes. "Tree-adjoining grammars." Handbook of formal languages. Springer Berlin Heidelberg, 1997. 69-123.
[9] Anthony S. Kroch and Aravind K. Joshi "The Linguistic Relevance of Tree Adjoining Grammar" University of Pennsylvania,issue: June(1985).
[10] Vijay-Shanker, K. "Using descriptions of trees in a tree adjoining grammar." Computational Linguistics 18.4 (1992): 481-517.
[11] Joshi, Aravind K. "Explorations of a Domain of Locality: Lexicalized Tree-Adjoining Grammar (LTAG)." University of Utrecht (CLIN meeting). 1999.

Monika Choudhary, born on 25th April 1990 at ITARSI (M.P), received the B.E degree in Computer Science from RGTU, Bhopal and pursuing M.Tech from Banasthali University, currently doing internship from CDAC Pune. Her area of interest is Information Retrieval, Artificial Intelligence, NLP and Machine translation.
Email: monikachoudhary2504@gmail.com

Nirja Shukla born of 12th Nov 1987 at New Delhi completed her B. Tech in Information Technology from Alwar Institute of Engineering and Technology (Alwar), also pursuing M.Tech from Banasthali University, currently doing internship from CDAC Pune. Her area of interest is Information Retrieval, Artificial Intelligence, NLP and Machine translation.

Mr. Shashi Pal Singh is working as STO, AAI Group, C-DAC, Pune. He has completed his B.Tech and M.Tech in Computer Science & Engineering and has published various national & international papers. He is specialised in Natural Language Processing (NLP), Machine assisted Translation (MT), Cloud Computing and Mobile Computing.

Mr. Ajai Kumar is working as Associate Director and Head, AAI Group, C-DAC, Pune. He is handling various projects in the area of Natural Language Processing, Information Extraction and Retrieval, Intelligent Language Teaching/Tutoring, Speech Technology [Synthesis & Recognition ASR], Mobile Computing, Decision Support Systems & Simulations and has published various national & international papers.

Dr. Hemant Darbari is working as Executive Director in C-DAC, Pune. He is one of the founding members of C-DAC, an R&D institute set up by the Department of Electronics and Information Technology; Govt. of India for carrying out advanced research in new and emerging technological domains. He has to his credit, 85 Technical Papers that have been published in national & international Journals & Conference Proceedings.