TREE ADJOINING GRAMMAR BASED "LANGUAGE INDEPENDENT GENERATOR"

Pavan Kurariya, Prashant Chaudhary, Jahnavi Bodhankar, Lenali Singh, Ajai Kumar and Hemant Darbari
Centre for Development of Advanced Computing, Pune, India
pavank@cdac.in, cprashant@cdac.in, jahnavib@cdac.in, lenali@cdac.in, ajai@cdac.in and darbari@cdac.in

Abstract

This paper proposes language independent natural language generator for Tree Adjoining Grammar (TAG) based Machine Translation System. In this model, the TAG based parsing and generation approach considered for the syntactic and semantic analysis of a source language. This model provides an efficient and a systematic way of encapsulating language resources with engineering solution to develop the machine translation System. A TAG based Generator is developed with existing resources using TAG formalism to generate the target language from TAG based parser derivation. The process allows syntactic feature-marking, the Subject-Predicate Agreement marking and multiple synthesized generated outputs in complex and morphological rich language. The challenge in applying such approach is to handle the linguistically diversified features. It is achieved using rule-based translation grammar model to align the source language to corresponding target languages. Nevertheless, this paper also describes the process of lexicalization and explain the state charts, TAG based adjunction and substitution function and the complexity and challenges beneath parsing-generation process.

1 Introduction

Machine Translation is a sub-field of (computational linguistics) under natural language processing (NLP) where computer is act as a human translator. It processes natural language constructs to automate the process of language translation. Every machine translation system requires programs for translation and automated dictionaries and grammars to support translation. Among the various statistical and rule based methodologies, we have researched on Grammar based model for English to Indian language Translation. Lexicalized Tree Adjoining Grammar (LTAG) is a non Chomsky formalism initially proposed in (Joshi et al., 1975) considered to be mildly context grammar that is ideal for any natural language.

The proposed Translation scheme is based on compatible Tags of the source and target languages. A TAG Parser is act as compliers used to analyze the source sentence based on the Tree Adjoining Grammar and construct the source derivation which is the summarization of the state chart processing. TAG Generator is like an interpreter that interprets the source derivation to a target derivation and lexicalizes the target Derivation into Derived Tree Generator which gives us the target Sentence accordingly.

The basic motivation to use this TAG Grammar model is that it is a rule based language independent feature oriented approach. Any complex agglutinative language can be represented using Tags and their unification features. The translation accuracy of this approach depends on the Tree Grammar rather than "bag of corpus". While any statistical approach translation accuracy depends on the corpus, more the corpus size better will be the output. We can still build a robust model for Indian languages with complex verb and noun morphology. Generation of the tree grammar for the source and Target language requires good linguistic knowledge and expertise for providing feature with the grammar.

Related research work of TAG Generator in NLP is presented in Section-2 as Literature survey. Section-3 elaborates more about basic flow of TAG Generator considering the TAG based MTS. Detailed architecture along with internal modules description have also been explained in this section. While Section-4 talks about results and analysis. Conclusion of the paper is done in Section-5.

2 Literature Survey

There has been a fair amount of research in the field of Tree Adjoining Grammar based generation of Machine Translation (MT). Some closely related research work (Joshi et al., 1997) is reported to address the similar work. (Joshi et al., 1997) discussed that TAG may be an appropriate formalism for generation because of their syntactic

Proceedings of the 17th International Conference on Natural Language Processing, pages 138–143
Patna, India, December 18 - 21, 2020.

attributes. The same observation was found in the work of (J. firgen Wedekind et al. 1988), who applies TAGs to the task of generation. Several researchers describe the properties of TAGs for extracting the syntactic processing of a natural language essential for natural-language generation. Although, generation is not a problem in Machine Translation Application as any system that is based on the TAG formalism has to build a generation component by which a TAG can articulate appropriately with semantic information. In this paper, we discuss one such mechanism where source and target grammar are aligned by defining a relation between the rule sets. The recent research in this field can be viewed as an effort to utilize Syntactic and semantic feature during the generation process. As discussed by (J. firgen Wedekind et al. 1988) requires a property of a grammar which specifies that complements be semantically connected to their head while (Stuart M. Shieber et al. 1988) defines a notion of semantic information, a compositional property which guarantees that it can be locally determined whether phrases can contribute to forming an expression with a given meaning. Generation approach that reorder top-down generation (Marc Dymetman et al. 1988) so as to make available information that utilize the top- down recursion also fall into the localizing information. Semantic-head-driven generation (Yves Schabes et al. 1989) uses semantic heads and their complements as a locus of semantic locality.

3 Workflow of TAG Generator

As earlier said, TAG Generation act like interpreters, just like Java virtual machine (JVM) in the Java Programming language which interpret the byte code to machine code, like wise TAG Generator interpret the source derivation to target Language. TAG Generation compresses into three parts: Transfer Model, Derivation generation, Derived tree generator.

As shown in figure 1, Transfer model is a linguistic based model in which source tree is mapped with the target tree i.e. every node of the source tree is mapped with the node of target tree which is called as Link Information. Link Information is an agreement between a source tree and the target tree, illustrate the node mapping between the language pair. Derivation Generation process the Derivation Parser (byte code in java) convert it into intermediate Derivation understand by the Generator intermediate process called the Derived Tree Generator. Derived Tree Generator is the Lexicalization process of the Derivation Generator which gives us the target output and language based feature required for synthesis.

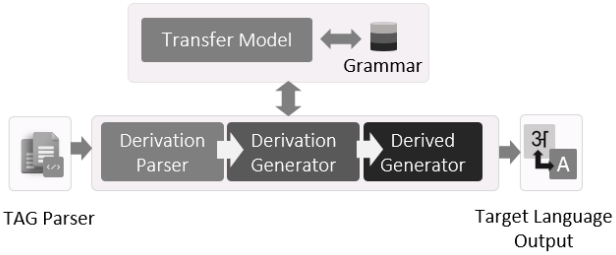

Figure 1: Basic architecture of TAG Generator

3.1 Tree Vector

Tree vector is a very important structure for parser and generator. The tree vector is like a pool for TAG trees, from which the lexicalized trees are spooled up for parsing and generation. The tree vector is a conventional structure implicitly defined as an input to the parser. The tree vector holds maps between trees, tree names and lexicons. There is a string array which holds the segmented sentence with all the words in it. Each word is a key to map, holding the set of tree lexicalized by that word. It also contains a reverse map where a tree is a key to a set of lexicon that uses this particular tree. The most interesting concept of the tree vector is the idea of non repeating trees. The tree vector stores exactly one copy of every tree even if it is lexicalized by many words in the sentence.

3.2 Multithreaded TAG Parser & Derivation

Figure 2, depicts monolithic hybrid parser for Tree Adjunction Grammar (constraint). It is modified multithreaded implementation of 'Early-Type Parsing Algorithm' by Arvind Joshi (Joshi et al. 1997). In this multithreaded parser, every parse requires multithreading and the parser clones a new thread but with a different current state. Multi threaded parser implements the higher RECOGNIZER algorithm. It is an offline recognizer it is designed to identify the first successful parse of an input string. The termination of the thread of the successfully parse is meant to be the end of that iteration. The 'RECOGNIZER' is a non-backtracking algorithm, which instead multi-threads all possibilities at a decision point. The main decisions involves the operation of adjunction and substitution as which tree should be adjuncted or substituted at a given node. The initiation of the parser itself is parallel. Externally the parser starts with 'n' sentence initial

trees, each of which initiates a parse in a different thread. This means that multi-threading occurs at 2 levels, one at the start and then at the parsing level.

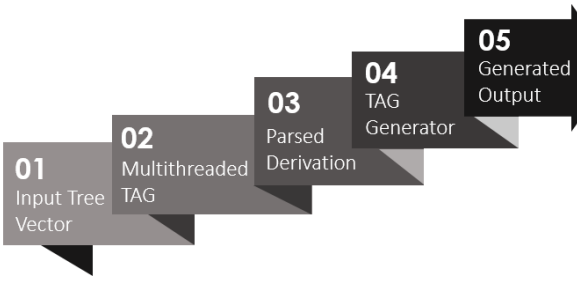

Figure 2: Workflow of TAG Generator

3.3 Transfer Model

It is a linguistic based model in which grammar is written in the tree format for the source and target language. In this model, the source and target grammar are aligned by defining a relation between the rule sets. Similarly of two generative grammar can be mathematically proved to be the similarity in their rule sets and not the language generated. The translation model that we are defining reflects from the proposal in [Abeille, Schabes and Joshi, 1990] is in fact manipulation of multiple similar Grammars. Consider the following illustration where two TAG trees are drawn, similar in nature. The mapping between the nodes is evident here. But there is one problem as the computability of the representation. Through it is complete with respect to the information conveyed; it is computationally redundant. That means the generator will have to compute the actual links from the lexical link given.

Link-info:

1.S_r=2.S_r.~1.NPadjn=2.NPadjn.~1.NP_0=2.NP.~1.VPadjn=2.VPadjn.~1.VP=2.VP.~1.PP=2.PP_1.~1.V=2.ADJ.~

3.4 Target derivation generation

There was an observation during the research on the TAG Generator that how TAG Parser communicates with the TAG Generator. TAG Parser keeps all the parsing information in the states, are stacked in the state chart. TAG Generator doesn't know about the parsing algorithm, state and state chart. To process state chart, you have to know the flow of the parsing so we need one structure which keep the summarization of the state chat and it could be understandable by Generator that interpret in target language.

To summarize the state chart information, we adopted the derivation structure(see figure 3) which is a record of how the elementary trees of a TAG are put together by the operations of substitution and adjoining in order to obtain the derived tree whose yield is the string being parsed. The nodes of the derivation tree are labeled by the names of the elementary trees and the edges are labeled by the addresses of the tree labeling the parent node in which the trees labeling the child nodes are either substituted or adjoined.

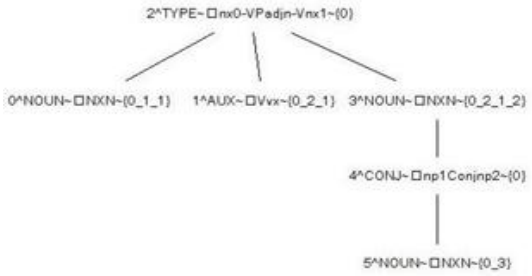

Figure 3: Parser derivation

Generator derivation is the kernel, building the target derivation and tree vector (see figure 4). It also dynamically builds the target tree vector, which is used to clone tree and stitching the clone together to get the final derivation tree. We build the target derivation from the parser derivation. Every node of the parser derivation is mapped with the Target derivation node. During mapping, name of the source elementary tree will be mapped with the target elementary tree, source lexicon will be replace with the target lexicon and Translation model identify in which node of the parent tree, child tree will be adjuncted , it is indicated by gorn number.

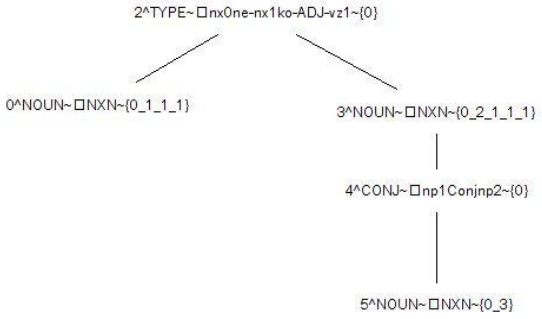

Figure 4: Generated derivation

Major advantages to make derivation is to save memory, don't need to know how the parser

analyze the sentence and easily understandable by generation process. Lexicalization of the generation process could be done by only derivation information, not need of extra information. But there is one drawback, it is very sensitive about grammar, mapping between the nodes in the trees (source and target) should be proper mapped otherwise operation will give the wrong output.

3.5 Parse Tree Builder (Derived Tree Generator)

Another concern in our generation process is when and where will the actual lexicalization of the derived tree happen? The lexicon that a tree is to be lexicalized with is present in the derivation node mapped to that tree. The lexicon that a tree is to be lexicalized with is present in the derivation node mapped to that tree. So this property of the derivation map is used to Preserve and still achieve strong lexicalization of the derived tree that is generated. The target derivation node will preserve the additional information required for smoothing and morph synthesis, so no external map or structure are required to carry them.

The derivation tree summarizes all the translate information stores in the state chart and compresses it to minimal size (see figure 5), easier to manipulate. But to see the actual derivation of how the sentence emerged from the grammar trees, we require the translate tree or more commonly, just, the translate. It is stitching of the entire set of tree used in deriving this particular sentence. But although it is a straight forward problem with a linear time algorithm, it has vast temporal space complexities, when we come to implementation. The TAG grammar derives the sentence in a lexical order and other order should not be assumed. To make it manipulatable in post and pre-order space, we

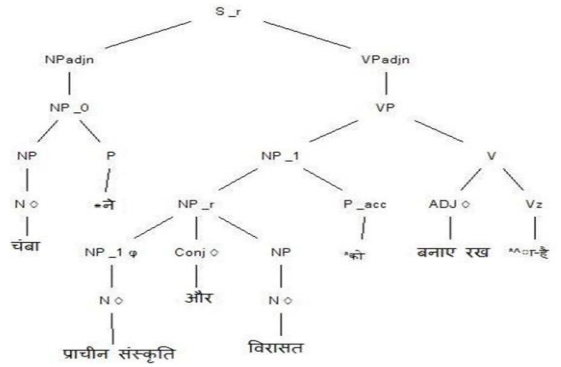

Figure 6: Derived Generation

terms of subject object agreement marker, which gives us the information of the subject, Object and main verb; Possessive case marker in the Indian language like 'Ne', 'Ko', 'Ke-Pass'; multiple synonym generation; multiple output generation based on the context.

TAG Generation approach give us lot of feature in terms of subject object agreement marker, which gives us the information of the subject, Object and main verb; Possessive case marker in the Indian language like 'Ne', 'Ko', 'Ke-Pass'; multiple synonym generation; multiple output generation based on the context.

4 Results and Experiments

Sample sentences covering different type of grammatical structures were generated using TAG Generator in different languages. One of the examples taken from English to Hindi Translation process, here figure 7 and figure8 depict derivation trees created in source and target language. It indicates syntactic relationship between word and how this information has utilized during translation process. Here Tree Vector (see figure 6) is also shown for source sentence.

INPUT TO PP BEAN
The Hawa-Mahal is the most recognizable monument of Jaipur
C-DAC POS OUTPUT
The Hawa-Mahal@@NOUN is @@VERB the most-recognizable-monument @@NOUN of@@PREP Jaipur@@NOUN
APPLIED REDUCTION RULES
@NP+-NP
NP+@VP+NP
@NP+@VP+NP
@NP+-MP
NP+@PP+NP
@NP+-NP
TREE VECTOR
0^ NOUN → [□NXN(Initial)]
1^ TYPE → [□nx0-VPadjn-V(Initial), □nx0-VPadjn-Vnx1-S(Initial), □nx0-VPadjn-Vnx1(Initial), □nx0e-Vpadjn-Vnx1(Initial)]
2^ NOUN → [□NXN(Initia)]
3^ PREP → [□nxPnx(Auxiliary),□vxadjnPnx(Auxiliary)]
4^ NOUN → [□NXN(Initial)]

Figure 5 : Tree Vector

Source Sentence: The Hawa-Mahal is the most recognizable monument of Jaipur.

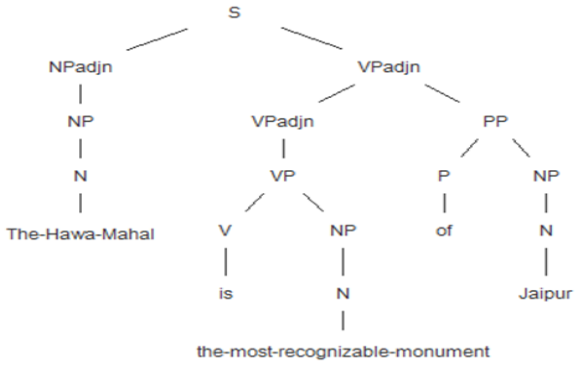

Figure 7: Derived Tress of Parser

Generator Output: हवा महल जयपुर की व्यापक स्तर पर पहचानने योग्य स्मारक है |

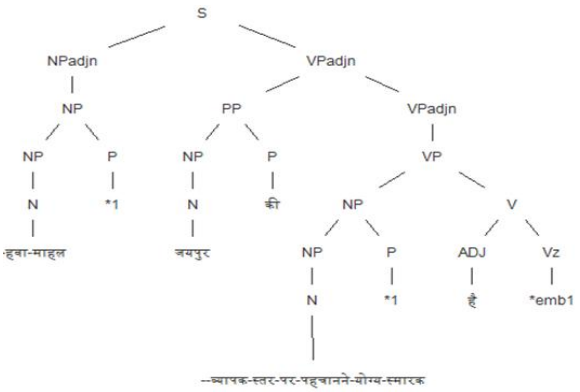

Figure 8: Derived Tress of Generator

4.1 TAG Generator Performance Analysis with PARAM Shavak

Aim of this experiment is to analyze Performance of TAG Generator using multi-core programming on PARAM Shavak*. Virtual Machine has been created by VMware to analyze the performance of Generator. Various test have been carried out to evaluate the Performance of EILMT System with different cores through vnc viewer . During the Test CPU usages and memory Utilizations has been observed.

TAG generator performance experiment on PARAM Shavak with different number of cores.

Data	Number of Cores Time in Second						
	1 core	2 core	4 core	6 core	8 core	12 core	16 core
Sample 1 (217 words)	36	34	32	29	23	20	17
Sample 2 (240 words)	32	21	16	14	12	12	11
Sample 3 (480 words)	58	36	28	26	24	23	21
Sample 4 (960 words)	110	74	52	49	46	42	40

Table 1: Performance experiment table

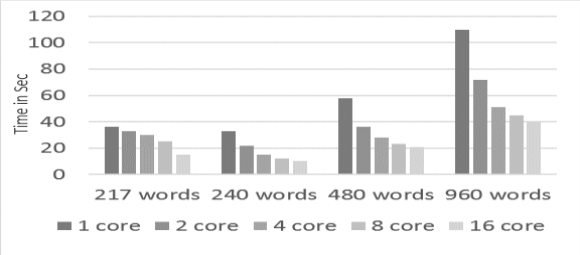

Figure 10: TAG generator performance experiment on PARAM Shavak

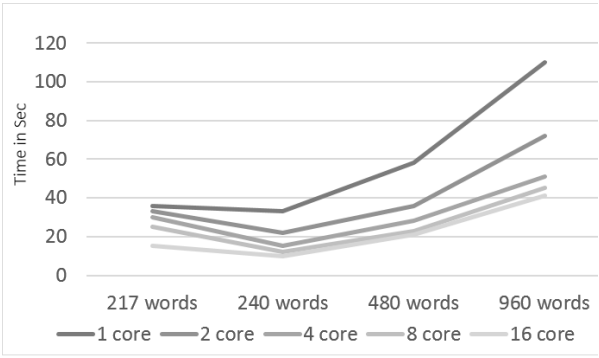

Figure 11: TAG generator performance experiment on PARAM Shavak

Above graph shows as we increase sample size, execution time decreases with each added cores during experiment while CPU usage increased 90-99 % and Memory utilization has been observed (2.1 to 2.8 GB) during test run

*PARAM Shavak: Supercomputer in a Box solution, aims to provide computational resource with advanced technologies to perform the high-end computations on a larger scale for the scientific, engineering and academic programmers. PARAM Shavak is a ready-to-use affordable supercomputer pre-loaded with all the required system software and applications from selected scientific and engineering domains.

5 Conclusion

In this paper, we have discussed some of the major tasks involved in the development of TAG generator using TAG formalism for translation from English to Hindi Language. All the examples illustrated above are taken from the output generated by the machine translation system developed by us. The effectiveness of this approach is tested by experiments on sample corpus abstract taken from existing resources. We have implemented TAG generator as a part of research in Machine Translation system based on Tree Adjoining Grammar. Firstly, we carried out experiments on web application (integrated MT System using TAG based Parser and Generator) running on Windows and Linux platform to provide a baseline. We have demonstrated that Parser and Generator are two core components in Machine Translation System. The system can handle simple and complex sentences with considerably good accuracy rate.

References

Joshi, A. K., Levy, L., and Takahashi, M. (1975). *Tree Adjunct Grammars*. Journal of Computer and System Sciences.

Joshi, A. K and Yves Schabes. 1997. *Tree Adjoining Grammars*. In Handbook of Formal Languages, volume 3, pages 69–123. Springer-Verlag, Berlin.

Nederhof, M.-J. (1998). *an alternative LR algorithm for TAGs*. In Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 16th International Conference on Computational Linguistics, Montreal, Canada.

Khalil Sima'an. 2000. *Tree-gram parsing: Lexical dependencies and structural relations*. In Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics, Hong Kong, China

Prolo, C. A. (Feb., 2002b). *LR parsing for Tree Adjoining Grammars and its application to corpus-based natural language parsing*. Ph.D. Dissertation Proposal, Department of Computer and Information Science, University of Pennsylvania.

Karin Harbusch and Jens Woch. 2000d. *Reuse of plan–based knowledge sources in a uniform tag–based generation system*. Under submission, see: http://www.uni-koblenz.de/~harbusch/plantotag.ps.

Schabes, Y. and Vijay-Shanker, K. (1990). *Deterministic left to right parsing of tree adjoining languages*. In Proceedings of 28th Annual Meeting of the Association for Computational Linguistics, pages 276–283, Pittsburgh, Pennsylvania, USA.

Tomita, M. (1985). *Efficient Parsing for Natural Language*. Kluwer Academic Publishers, Boston, MA, USA.

Nicolas Nicolov. 1998. *Memoisation in sentence generation with lexicalized grammars*. In Abeill´e et al.(Abeill´e et al., 1998), pages 124–127.

XTAG Research Group, T. (1998). *A Lexicalized Tree Adjoining Grammar for English.* Technical Report IRCS 98-18, University of Pennsylvania.

David D. McDonald and James D. Pustejovsky. *TAGs as a grammatical formalism for generation.* In Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics, pages 94-103, University

J. firgen Wedekind. *Generation as structure driven derivation*. In Proceedings of the 12th International Conference on Computational Linguistics, pages 732- 737, Budapest, Hungary, 1988.

Stuart M. Shieber. *A uniform architecture for parsing and generation*. In Proceedings of the 12th International Conference on Computational Linguistics, pages 614-619, Karl Marx University of Economics, Budapest, Hungary, 22-27 August 1988.

David D. McDonald and James D. Pustejovsky. *TAGs as a grammatical formalism for generation*. In Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics, pages 94-103, University

J. firgen Wedekind. *Generation as structure driven derivation*. In Proceedings of the 12th International Conference on Computational Linguistics, pages 732-737, Budapest, Hungary, 1988.

Stuart M. Shieber. *A uniform architecture for parsing and generation*. In Proceedings of the 12th International Conference on Computational Linguistics, pages 614-619, Karl Marx University of Economics, Budapest, Hungary, 22-27 August 1988.

Marc Dymetman and Pierre Isabelle. *Reversible logic grammars for machine translation*. In Proceedings of the Second International Conference on Theoretical and Methodological Issues in Machine Translation of Natural Languages, Pittsburgh, Pennsylvania, 1988. Carnegie-Mellon University.

Yves Schabes and Aravind K. Joshi. *The relevance of lexicalization to parsing*. In Proceedings of the International Workshop on Parsing Technologies, pages 339-349, Pittsburgh, Pennsylvania, 28-31 August 1989. Carnegie-Mellon University.