**National University of Computer & Emerging Sciences**

**Karachi Campus**

A picture containing text, metalware, gear, wheel

Description automatically generated

**THEORY OF AUTOMATA**

**PROJECT REPORT**

**Topic: NATURAL LANGUAGE PROCESSING**

**SUBMITTED TO:**

SIR Musawar

**GROUP MEMBERS:**

Insha Samnani (20K-0247)

Ismail Ahmed Ansari (20K-0228)

**INTRODUCTION:**

**NATURAL LANGUAGE PROCESSING** refers to a branch of **COMPUTER SCIENCE** that deals with the ability of the **COMPUTERS** to understand the text and words same as **HUMAN** **BEINGS**. It applies the computational techniques to analyze or synthesize the speech or natural language. But why should we talk about **NLP**? Because it have many **APPLICATIONS**:

* Smart Assistants
* Email Filters
* Language Translation
* Data and Text Analysis

And much more. It is to be noted that there are five phases of **NLP**.

* Lexical or structure analysis
* Parsing
* Semantic Analysis
* Discourse Integration
* Pragmatic Analysis

Although our report will not cover all the phases as we are bounded to specific topics that have limited applications. Which **TOPICS** and **APPLICATIONS**? Here you go...

It basically involves the idea of **ARTIFICIAL INTELLIGENCE**. **AI**? Sounds very difficult and boring. However, the concept behind **NLP** can be well understood by various concepts of **THEORY OF AUTOMATA** manually. You don’t need to go through the hard and fast **ALGORITHMS** to understand its basic idea. We have summarized the working of **NLP** via **CONTEXT FREE GRAMMAR** and **FINITE STATE AUTOMATA**. How **CFG** and **FA**? You don’t need to get confused right now. We will take you to the deep analysis of these concepts in **NLP**. By the time you can have the citations of **FOUR** research papers that we have studied to prepare this **REPORT**:

1. *Chirag Goyal (June 24, 2021)*, “Step by Step Guide to Master NLP – Grammar in NLP”
2. *Andrew McCallum (2007)*, “Context Free Grammars and Introduction to Natural Language Processing”
3. *Jimmy Ma (December 2008)*, "Automata in Natural Language Processing"
4. *Ronald M. Kaplan* and *Martin Kay* *(Unknown),* "Finite State Methods in Natural-Language Processing"

**NATURAL LANGUAGE PROCESSING AND CONTEXT FREE GRAMMAR:**

**CITATION 1:**

**WHAT IS GRAMMAR?**

Grammar is defined as syntactical rules for forming well-structured sentences in natural languages. Mathematically, it can be written as:

N -> Set of **NON-TERMINALS**.

T -> Set of **TERMINALS**.

S -> **START SYMBOL**.

P -> **PRODUCTION RULES** for **N** and **T.**

**WHAT IS CONTEXT FREE GRAMMAR?**

**CFGs** are the superset of **REGULAR** **LANGUAGES** that are used to describe the languages using some grammar rules. It consists of following components:

1. Set of **NON-TERMINALS**: They helps in defining the strings that will be generated through grammar.
2. Set of **TERMINALS** (Tokens): They helps to form strings.
3. **START SYMBOL** (S): It helps to initiate the productions.
4. Set of **PRODUCTIONS**: They helps to give an idea regarding the combinations of terminals and non-terminals.

**WHAT IS CONSTITUENCY GRAMMAR?**

**CGs** are based on the constituency relations. It helps to derive the constituency relation to organize any sentence into its constituents and view the clause structure in terms of **NP** (Noun Phrase) and **VP** (Verb Phrase). We can consider three constituents that are: **SUBJECT**, **CONTEXT**, **OBJECT**. Now it can take different words of these categories to generate different sentences. **FOR** **EXAMPLE**:

**SUBJECT**: Insha | Ismail

**CONTEXT**: is studying | is playing

**OBJECT**: in park. | in room.

Following sentences can be generated using the above-mentioned constituencies.

* Insha is studying in room.
* Ismail is playing in park.

This is the view of **CG** in terms of **NOUN** and **VERB** identification. Another view of **CGs** is in terms of **SPEECH** **TAGS**. We can consider its structure as:

[**DETERMINER**, **NOUN**] [**ADJECTIVE**, **VERB**] [**PREPOSITION** **DETERMINER** **NOUN**]

Same Example can be constituted as:

<**NN**> <**JJ** **VB**> <**PRP** **NN**> ------------------- > Insha is studying in room.

<**NN**> <**JJ** **VB**> <**PRP** NN**>** ------------------- > Ismail is playing in park.

Note that it is not necessary that all tags are fulfilled. Here Determiners are missing.

**WHAT IS DEPENDENCY GRAMMAR?**

DGs are having the words connected to each other via directed links. They are opposite to CGs and are based on dependency relations. One of the word acts as a root and all other words of the same sentence are linked to that word. DGs are used to figure out the structure and dependencies between the words. We can consider its structure as:

(**GOVERNOR**, **RELATION**, **DEPENDENT**)

This means that Dependent will be linked to governor via some relation. Consider the same examples as given in **CGs**:

<Insha> <is> <studying in room.>

Here **INSHA** is the **SUBJECT** and **GOVERNOR**. IS is **VERB**. **STUDYING IN ROOM** is **DEPENDED** on **GOVERNOR**. Now it’s your time to explain this example:

<Ismail> <is> <playing in park.>

**WHAT IS THE USE OF DGs?**

* **COREFERENCE RESOLUTION**: To map pronouns to respective noun phrases.
* **QUESTION ANSWERING SYSTEM**: To understand its relational and structural aspects.
* **TEXT SUMMARIZATION AND TEXT CLASSIFICATION**.

**GAPS IN CITATION:**

* They didn’t introduce **REGULAR EXPRESSIONS** to compare the benefits of **CFGs**.
* They didn’t provided examples of **CFGs** to well understand the concept of CGs and DGs. They also not compared CFGs while explaining **CGs** and **DGs** to analyze the application of **CFGs** in **NATURAL LANGUAGE PROCESSING**.
* They didn’t provide real life applications of **CGs** as provided for **DGs**.
* They Didn’t conclude the topic clearly to maintain a relationship between **CFGs**, **CGs**, and **DGs**.

**CITATION 2:**

**WHAT IS LANGUAGE STRUCTURE AND MEANING?**

Basically, here the idea is given that what language structure is used to map the meaning of the sentence? Some common ways are:

* [State [Thing Insha] was sitting [Place in the garden] [Time yesterday]]
* [Action [Thing Ismail] shouted [Property/Manner loudly]]
* [Thing Ismail] is [Place in the garden]

**WHAT IS CONSTITUENCY?**

Group of words can be constituted to indicate a specific meaning. **FOR EXAMPLE**, “*Third March*” is a **DATE**. Sentences can have subparts and the words that are grouped together are called as constituents. **FOR** **EXAMPLE**, following sentence can be grouped in different ways: “Insha hit Urwa with a hockey.”

* **POSSIBILITY** **1**: [Insha hit Urwa] with a hockey
* **POSSIBILITY** **2**: Insha hit [Urwa with a hockey]

**WHAT ARE CONSTITUENT PHRASES?**

Phrases are named based on the word that heads the constituent. **FOR EXAMPLE:**

* *Insha in garden. ------>* Noun Phrase (**NP**) because head is **INSHA**.
* *So pretty. -------------->* Adjective Phrase (**AP**) because head is **PRETTY**.
* *Ate ice cream. ------->* Verb Phrase (**VP**) because head is **ATE**.

**EVIDENCE CONSTITUENCY EXISTS:**

* They usually exist before a verb. **FOR EXAMPLE:**
  + *Ismail* came to me this Sunday.
  + *The reason they were fighting for the post* is open now
* The constituent can be used at different places in a sentence. **FOR EXAMPLE:**
  + Insha will come to me *on February fourteen*.
  + *On February fourteen*, Insha will come to me.
    - But the constituent cannot be split apart like:
      * *On Februar*y, Insha will come *fourteen* to me.
      * *On*, Insha will come *February fourteen* to me.

**WHAT IS CONTEXT FREE GRAMMAR?**

It is the way of modeling the constituencies. It consists of four components and described in **PREVIOUS** **CITATION**. The report also provided the **APPLICATION** **OF GRAMMAR REWRITE RULES.**

Text

Description automatically generated

Now different sentences can be generated using this **CFG**. **FOR EXAMPLE**:

S -> NP VP

* Det NOM VP
* The Noun VP
* The man VP
* The man Verb NP
* The man book NP
* The man book Det NOM
* The man book the Noun
* The man book the flight

**CFGs** can also look for **RECURSION**. This means, you can continuously use **NON-TERMINALS** and terminate them later. It also look for **GRAMMATICALITY** *(Sentences that can be derived by grammar)*.

**WHAT IS CHOMSKY HIERARCHY?**

Diagram, venn diagram

Description automatically generated

**PARSING REGULAR GRAMMAR, CONTEXT FREE GRAMMAR and CONTEXT SENSITIVE GRAMMAR:**

**REs** are the languages that can be generated by finite-state automata (**FAs**). They cannot do embedded recursion like . However, the limitation is overcome by **CFGs**. But they cannot do . However, this limitation is overcome by Context Sensitive Grammars. **CSGs** are the languages that can be recognized by **Turing Machines**.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **REs** | **CFGs** | **CSLs** |
| **Space to Parse** | Constant | Stack | - |
| **Time to Parse** | Linear | O () | O () OR O () |

**WHAT IS CHOMSKY NORMAL FORM?**

The **CFG** that follows the following rules are in the normal form. However, any **CFG** can be converted in **CNF**. Rules are:

* S -> AB
* S -> a
* S -> e

**FOR EXAMPLE**: The **CNF** of the **CFG** given in *“WHAT IS CONTEXT FREE GRAMMAR?”* will be:

S -> NP VP Det -> that | this | a | the

S -> A VP Noun -> book | flight | meal | man

S -> book | include | read | VP NP Verb -> book | include | read

NP -> Det NOM Aux -> does

NOM -> Noun | Noun NOM

VP -> book | include | read | VP NP

A -> Aux NP

**WHAT IS PARSING?**

It is a search problem, where we want to derive a string using the production rules of grammars by finding ALL structures matching an input string of words. It can be achieved via **BOTTOM-UP** or **TOP-DOWN** way. If your goal can be achieved in different ways, then you can choose which rule to apply.

**WHAT IS TOP-DOWN PARSING?**

**TDP** is **GOAL-DIRECTED**. It rewrites the goal by matching one against **LEFT** and expanding it to the **RIGHT**. Consider the same **CFG**:

Diagram, engineering drawing

Description automatically generated

But there are several limitations to it: Infinite left recursions, useless, repeated and expanding work, etc.

**WHAT IS BOTTOM-UP PARSING?**

**BUP** is **DATA-DIRECTED**. It replaces the **RHS** of the rule if it matches the goal in the sequence by the **LHS** of the rule. Consider the same **CFG**.

A piece of paper with writing

Description automatically generated with medium confidence

But there are several limitations to it: Unable to deal with termination problem, useless and repeated work, and inefficient when there is ambiguity.

**WHAT IS SHIFT REDUCE PARSING?**

**SRP** performs two operations:

* **SHIFT**: Refers to push next input symbol from the sentence into stack.
* **REDUCE**: Occurs when we have rules of RHS on the top of the stack.

|  |  |  |
| --- | --- | --- |
| **STACK** | **INPUT REMAINING** | **ACTION** |
| () | Include the meal | Shift |
| (Include) | the meal | Reduce, Verb -> Include |
| (Verb) | the meal | Shift |
| (Verb that) | meal | Reduce, Det -> that |
| (Verb Det) | meal | Shift |
| (Verb Det meal) |  | Reduce, Noun -> meal |
| (Verb Det Noun) |  | Reduce, NOM -> Noun |
| (Verb Det NOM) |  | Reduce, NP -> Det NOM |
| (Verb NP) |  | Reduce, VP -> Verb NP |
| (Verb) |  | Reduce, S -> V |
| (S) |  | **DONE!** |

But there is a limitation to it: Backtracking needed when there is an ambiguity.

**GAPS IN CITATION:**

All gaps that were fount in the **PREVIOUS CITATION** were overcome by this **CITATION**, **EXCEPT**: The topics were not organized in **HEIRARCHY**.

**NATURAL LANGUAGE PROCESSING AND FINITE STATE AUTOMATA:**

**BOTH CITATIONS:**

**WHAT ARE FINITE STATE AUTOMATA:**

Finite Automaton/Finite Machine can be modeled as **ON/OFF** switch, A string which is passed to Finite State Automata will form an **INPUT** and the **OUTPUT** will be whether the string passed is **ACCEPTED OR REJECTED**. They have extremely limited memory, but it can still do many things and is very often used in our everyday life.

A picture containing text, sign, clipart

Description automatically generated

Mathematically, it can be written as:

**Q** -> Set of States

**E** -> INPUT ALPHABETS

**T** -> is the transition function

**q0** -> START SYMBOL

**F** -> FINITE STATE OF FINAL/ ACCEPTING STATE

# BASIC IDEA FOR FA

* Finite Number of States means **FINITE NUMBER OF STRINGS** can be read.
* Transitions will represent the **SYMBOLS** read on the tape.
* Transition possible, only when FA contains **CORRESPONDING TRANSITION**
* String Accepted, only when FA is in **FINAL STATE**.

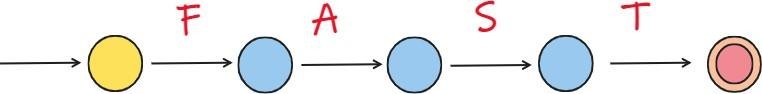
A picture containing text, clipart

Description automatically generated

# WORD RECOGNITION

FA works as a word recognizer, which will take input of string characters as input and returns **“YES”** or **“NO”**.

For word recognition we can use two methods:



A picture containing diagram

Description automatically generated

# APPROXIMATE METHODS:

**RIGHT SET** (alphabets can be in any order) and **RIGHT SOUND**, of the string to the machine, will be further forwarded so we can analyze a string. A bit complex and difficult as it is not possible on a finite automaton state but for a concept that for how many ways a string can be read.

# FOR EXAMPLE:

A word is tokenized into the defined set of alphabets and the tokens are passed to the hash function and searched in the bit table for acceptance, a typical but YES/NO machine.

A picture containing text, toiletry, businesscard

Description automatically generated

# EXACT METHOD

The finite machine moves to any **COMBINATION OF STATES** and when passed a string it tries to find the possibility of the word, A tree is generated using the Hashing, Searching methods for a complete word recognition in short.

Let us take an **EXAMPLE**:

Diagram

Description automatically generated

# DICTIONARY LOOKUP

FA also works as a **DICTIONARY FINDER**, which will take input of string characters as input and returns “YES” or “NO”, according to the word present and will also return the mapped information about the word.

For Dictionary lookup we can use three techniques:

# WORD IDENTIFIERS

Each word is mapped against a **UNIQUE IDENTIFIER**, it is like indexing the total words. All combinations of strings also referred as **KLEENE STAR/UNARY/FREE MONOID** constructions are listed and then indexes are labelled against eacA picture containing diagram

Description automatically generatedh.

# PREORDER WALK

The idea behind this is to **COUNT ALL POSSIBILITIES OF FINAL STATES** when strings are read, ignoring the case that suffix of each string coincide.

# Diagram, schematic Description automatically generatedFOR EXAMPLE:

1. **SUFFIX COUNTS**

Suffix Automaton **MINIMIZES POSSIBILITIES** of a string, that recognizes same suffixes of s. The concept is to skip the irrelevant transitions and check from starting states from where to add, or to read and then it increments the count.

In this Finite Machine, strings having **SAME SUFFIX WILL FOLLOW UP UNTIL ENCOUNTERS DIFFERENT SUFFIX** they will break into another state.

# FOR EXAMPLE:

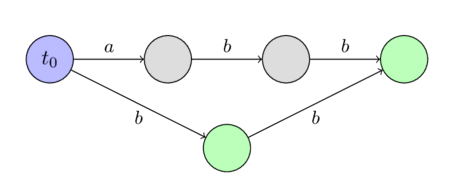
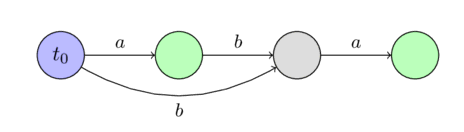
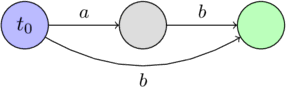
Diagram, venn diagram

Description automatically generatedA picture containing diagram

Description automatically generated

A picture containing clipart

Description automatically generated



**GAPS IN CITATION:**

Not a clearer picture for a person not familiar with automata, complex examples with no explanation, and where explanation is given, they are very straight forward. Broad configuration and more attention is needed to the main topic.

**PROPOSED SOLUTIONS**

So **FINALLY**, we have completed with **PROCESSING** **THE** **NATURAL** **LANGUAGE**. But what if there could be some **BETTERMENTS** in the **THEORY** **OF** **AUTOMATA** for **NATURAL** **PROCESSING** **LANGUAGE**? Let us talk about them:

* A solution to Finite Automata is that to have a higher machine because of the infinite tape and infinite memory, a limited practical usage of natural word processing because linear time insertion in FA if a new word is to be added, finite machine can only count where different states correspond to different values of the counter basically as said above a finite number of input scenario is there, Turing Machines, Push Down Automaton, Linear Bounded Automaton, etc.
* Secondly, it must be noted that whatever the type of machine is, it rejects without specifying the reason for rejection of the string or word of a language. Although, it’s very difficult to introduce such an application which can look after this flaw. But if overcame, could lead to more powerful and useful machines.
* Apart from this, as we know that each machine have their own pros and cons. What if such a machine could be designed to process the natural language that involves the pros of all the existing machines. An **EXAMPLE** could be:
  + **NATURAL LANGUAGE PROCESSOR** that could operate bi-directionally (**TURING MACHINES**), without accepting words other than the words of language (**NFA/DFA**), along with the hierarchical representation of deriving the words (**DERIVATION TREES**), and arrow directing clarifications to show the involvements of alphabets of that language step by step (**CFG**).

**REFERENCES:**

<https://www.analyticsvidhya.com/blog/2021/06/part-12-step-by-step-guide-to-master-nlp-grammar-in-nlp/>

<https://people.cs.umass.edu/~mccallum/courses/inlp2007/lect5-cfg.pdf>

https://[www.lrde.epita.fr/dload/20090107-Seminar/ma-automata\_in\_nlp.pdf](http://www.lrde.epita.fr/dload/20090107-Seminar/ma-automata_in_nlp.pdf)

<https://web.stanford.edu/class/linguist139p/Slides.pdf>