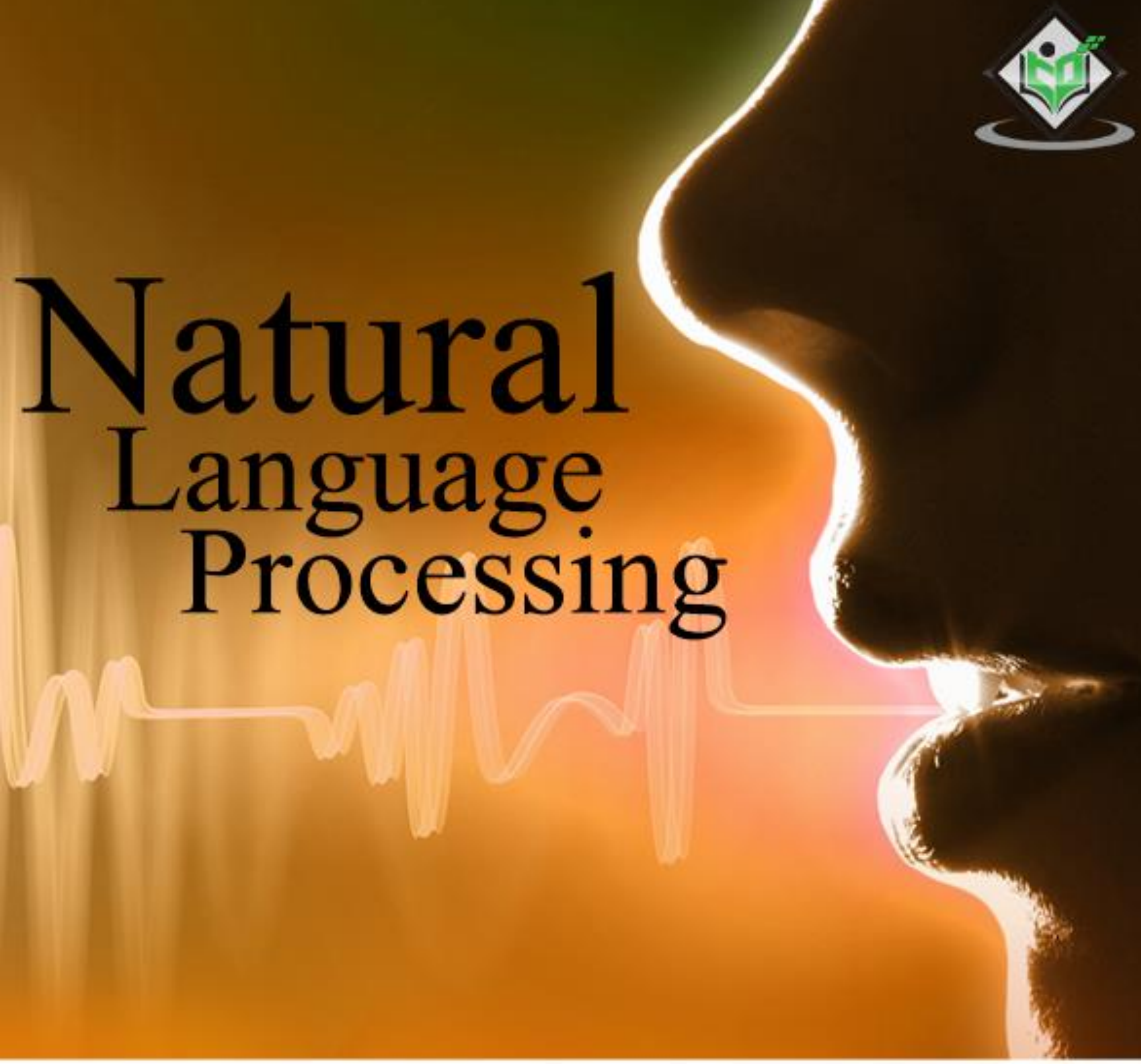
Natural Language Processing



tutorialspoint

SIMPLY EASY LEARNING

www.tutorialspoint.com

https://www.facebook.com/tutorialspointindia

https://twitter.com/tutorialspoint

About the Tutorial

Language is a method of communication with the help of which we can speak, read and write. Natural Language Processing (NLP) is a subfield of Computer Science that deals with Artificial Intelligence (AI), which enables computers to understand and process human language.

Audience

This tutorial is designed to benefit graduates, postgraduates, and research students who either have an interest in this subject or have this subject as a part of their curriculum. The reader can be a beginner or an advanced learner.

Prerequisites

The reader must have basic knowledge about Artificial Intelligence. He/she should also be aware about basic terminologies used in English grammar and Python programming concepts.

Copyright & Disclaimer

Table of Contents

1. Natural Language Processing — Introduction

Language is a method of communication with the help of which we can speak, read and write. For example, we think, we make decisions, plans and more in natural language; precisely, in words. However, the big question that confronts us in this AI era is that can we communicate in a similar manner with computers. In other words, can human beings communicate with computers in their natural language? It is a challenge for us to develop NLP applications because computers need structured data, but human speech is unstructured and often ambiguous in nature.

In this sense, we can say that Natural Language Processing (NLP) is the sub-field of Computer Science especially Artificial Intelligence (AI) that is concerned about enabling computers to understand and process human language. Technically, the main task of NLP would be to program computers for analyzing and processing huge amount of natural language data.

History of NLP

We have divided the history of NLP into four phases. The phases have distinctive concerns and styles.

First Phase (Machine Translation Phase) – Late 1940s to late 1960s

The work done in this phase focused mainly on machine translation (MT). This phase was a period of enthusiasm and optimism.

Let us now see all that the first phase had in it:

- The research on NLP started in early 1950s after Booth & Richens' investigation and Weaver's memorandum on machine translation in 1949.
- 1954 was the year when a limited experiment on automatic translation from Russian to English demonstrated in the Georgetown-IBM experiment.
- In the same year, the publication of the journal MT (Machine Translation) started.
- The first international conference on Machine Translation (MT) was held in 1952 and second was held in 1956.
- In 1961, the work presented in Teddington International Conference on Machine Translation of Languages and Applied Language analysis was the high point of this phase.

Second Phase (AI Influenced Phase) – Late 1960s to late 1970s

In this phase, the work done was majorly related to world knowledge and on its role in the construction and manipulation of meaning representations. That is why, this phase is also called AI-flavored phase.

The phase had in it, the following:

- In early 1961, the work began on the problems of addressing and constructing data or knowledge base. This work was influenced by AI.

- In the same year, a BASEBALL question-answering system was also developed. The input to this system was restricted and the language processing involved was a simple one.

- A much advanced system was described in Minsky (1968). This system, when compared to the BASEBALL question-answering system, was recognized and provided for the need of inference on the knowledge base in interpreting and responding to language input.

Third Phase (Grammatico-logical Phase) – Late 1970s to late 1980s

This phase can be described as the grammatico-logical phase. Due to the failure of practical system building in last phase, the researchers moved towards the use of logic for knowledge representation and reasoning in AI.

The third phase had the following in it:

- The grammatico-logical approach, towards the end of decade, helped us with powerful general-purpose sentence processors like SRI's Core Language Engine and Discourse Representation Theory, which offered a means of tackling more extended discourse.

- In this phase we got some practical resources & tools like parsers, e.g. Alvey Natural Language Tools along with more operational and commercial systems, e.g. for database query.

- The work on lexicon in 1980s also pointed in the direction of grammatico-logical approach.

Fourth Phase (Lexical & Corpus Phase) – The 1990s

We can describe this as a lexical & corpus phase. The phase had a lexicalized approach to grammar that appeared in late 1980s and became an increasing influence. There was a revolution in natural language processing in this decade with the introduction of machine learning algorithms for language processing.

Study of Human Languages

Language is a crucial component for human lives and also the most fundamental aspect of our behavior. We can experience it in mainly two forms – written and spoken. In the written form, it is a way to pass our knowledge from one generation to the next. In the spoken form, it is the primary medium for human beings to coordinate with each other in their day-to-day behavior. Language is studied in various academic disciplines. Each discipline comes with its own set of problems and a set of solution to address those.

Consider the following table to understand this:

Discipline	Problems	Tools
Linguists	How phrases and sentences can be formed with words? What curbs the possible meaning for a sentence?	Intuitions about well-formedness and meaning. Mathematical model of structure. For example, model theoretic semantics, formal language theory.
Psycholinguists	How human beings can identify the structure of sentences? How the meaning of words can be identified? When does understanding take place?	Experimental techniques mainly for measuring the performance of human beings. Statistical analysis of observations.
Philosophers	How do words and sentences acquire the meaning? How the objects are identified by the words? What is meaning?	Natural language argumentation by using intuition. Mathematical models like logic and model theory.
Computational Linguists	How can we identify the structure of a sentence How knowledge and reasoning can be modeled? How we can use language to accomplish specific tasks?	Algorithms Data structures Formal models of representation and reasoning. AI techniques like search & representation methods.

Ambiguity and Uncertainty in Language

Ambiguity, generally used in natural language processing, can be referred as the ability of being understood in more than one way. In simple terms, we can say that ambiguity is the capability of being understood in more than one way. Natural language is very ambiguous. NLP has the following types of ambiguities:

Lexical Ambiguity

The ambiguity of a single word is called lexical ambiguity. For example, treating the word **silver** as a noun, an adjective, or a verb.

Syntactic Ambiguity

This kind of ambiguity occurs when a sentence is parsed in different ways. For example, the sentence "The man saw the girl with the telescope". It is ambiguous whether the man saw the girl carrying a telescope or he saw her through his telescope.

Semantic Ambiguity

This kind of ambiguity occurs when the meaning of the words themselves can be misinterpreted. In other words, semantic ambiguity happens when a sentence contains an ambiguous word or phrase. For example, the sentence "The car hit the pole while it was moving" is having semantic ambiguity because the interpretations can be "The car, while moving, hit the pole" and "The car hit the pole while the pole was moving".

Anaphoric Ambiguity

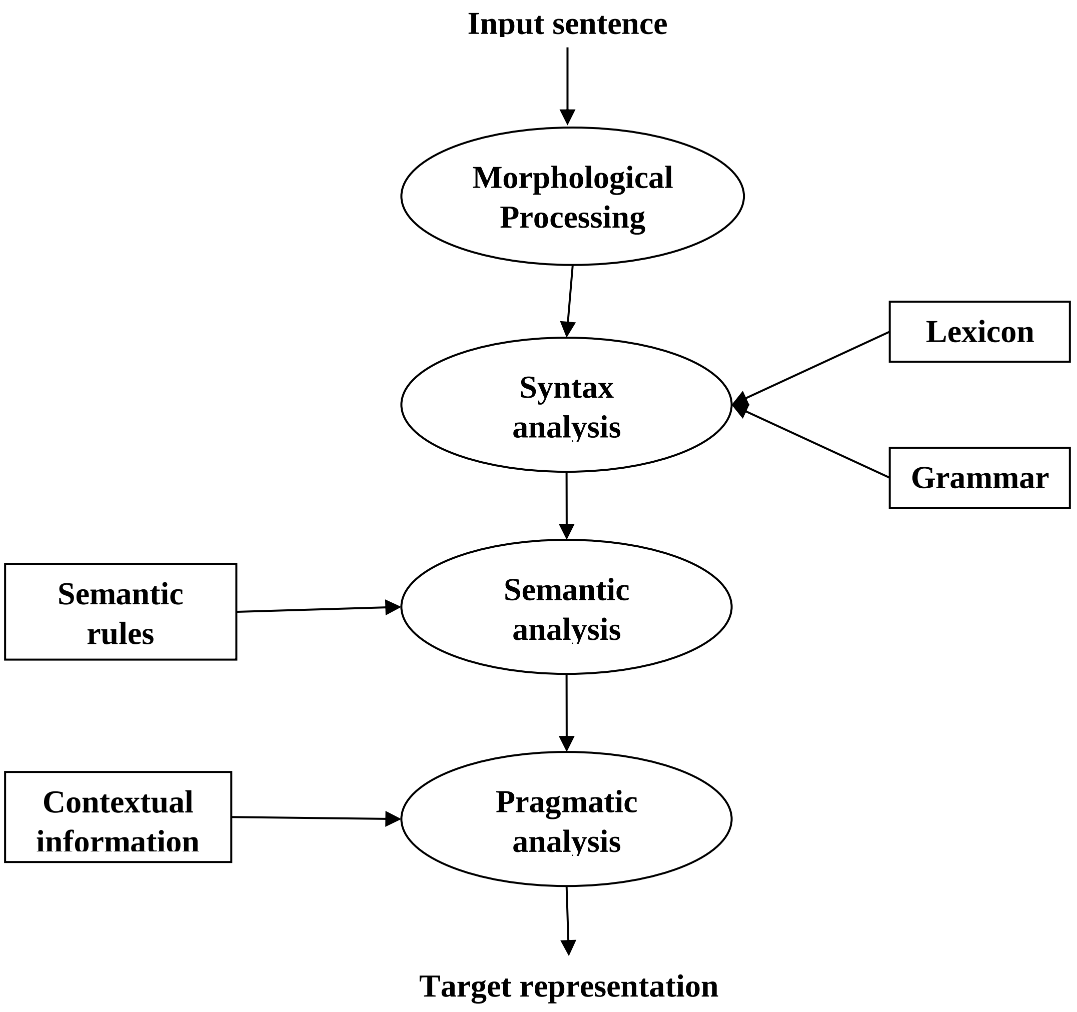
This kind of ambiguity arises due to the use of anaphora entities in discourse. For example, the horse ran up the hill. It was very steep. It soon got tired. Here, the anaphoric reference of "it" in two situations cause ambiguity.

Pragmatic ambiguity

Such kind of ambiguity refers to the situation where the context of a phrase gives it multiple interpretations. In simple words, we can say that pragmatic ambiguity arises when the statement is not specific. For example, the sentence "I like you too" can have multiple interpretations like I like you (just like you like me), I like you (just like someone else dose).

NLP Phases

Following diagram shows the phases or logical steps in natural language processing:



Morphological Processing

It is the first phase of NLP. The purpose of this phase is to break chunks of language input into sets of tokens corresponding to paragraphs, sentences and words. For example, a word like **"uneasy"** can be broken into two sub-word tokens as **"un-easy"**.

Syntax Analysis

It is the second phase of NLP. The purpose of this phase is two folds: to check that a sentence is well formed or not and to break it up into a structure that shows the syntactic relationships between the different words. For example, the sentence like **"The school goes to the boy"** would be rejected by syntax analyzer or parser.

Semantic Analysis

It is the third phase of NLP. The purpose of this phase is to draw exact meaning, or you can say dictionary meaning from the text. The text is checked for meaningfulness. For example, semantic analyzer would reject a sentence like "Hot ice-cream".

Pragmatic Analysis

It is the fourth phase of NLP. Pragmatic analysis simply fits the actual objects/events, which exist in a given context with object references obtained during the last phase (semantic analysis). For example, the sentence "Put the banana in the basket on the shelf" can have two semantic interpretations and pragmatic analyzer will choose between these two possibilities.

2. Natural Language Processing — Linguistic Resources

In this chapter, we will learn about the linguistic resources in Natural Language Processing.

Corpus

A corpus is a large and structured set of machine-readable texts that have been produced in a natural communicative setting. Its plural is *corpora*. They can be derived in different ways like text that was originally electronic, transcripts of spoken language and optical character recognition, etc.

Elements of Corpus Design

Language is infinite but a corpus has to be finite in size. For the corpus to be finite in size, we need to sample and proportionally include a wide range of text types to ensure a good corpus design.

Let us now learn about some important elements for corpus design:

Corpus Representativeness

Representativeness is a defining feature of corpus design. The following definitions from two great researchers – Leech and Biber, will help us understand corpus representativeness:

- **According to Leech (1991),** "A corpus is thought to be representative of the language variety it is supposed to represent if the findings based on its contents can be generalized to the said language variety".

- **According to Biber (1993),** "Representativeness refers to the extent to which a sample includes the full range of variability in a population".

In this way, we can conclude that representativeness of a corpus are determined by the following two factors:

- **Balance –** The range of genre include in a corpus.
- **Sampling –** How the chunks for each genre are selected.

Corpus Balance

Another very important element of corpus design is corpus balance – the range of genre included in a corpus. We have already studied that representativeness of a general corpus depends upon how balanced the corpus is. A balanced corpus covers a wide range of text categories, which are supposed to be representatives of the language. We do not have any reliable scientific measure for balance but the best estimation and intuition works in this concern. In other words, we can say that the accepted balance is determined by its intended uses only.

Sampling

Another important element of corpus design is sampling. Corpus representativeness and balance is very closely associated with sampling. That is why we can say that sampling is inescapable in corpus building.

- According to **Biber(1993),** "Some of the first considerations in constructing a corpus concern the overall design: for example, the kinds of texts included, the number of texts, the selection of particular texts, the selection of text samples from within texts, and the length of text samples. Each of these involves a sampling decision, either conscious or not."

While obtaining a representative sample, we need to consider the following:

- **Sampling unit:** It refers to the unit which requires a sample. For example, for written text, a sampling unit may be a newspaper, journal or a book.

- **Sampling frame:** The list of al sampling units is called a sampling frame.

- **Population:** It may be referred as the assembly of all sampling units. It is defined in terms of language production, language reception or language as a product.

Corpus Size

Another important element of corpus design is its size. How large the corpus should be? There is no specific answer to this question. The size of the corpus depends upon the purpose for which it is intended as well as on some practical considerations as follows:

- Kind of query anticipated from the user.
- The methodology used by the users to study the data.
- Availability of the source of data.

With the advancement in technology, the corpus size also increases. The following table of comparison will help you understand how the corpus size works:

Year	**Name of the Corpus**	**Size (in words)**
1960s-70s	Brown and LOB	1 Million words
1980s	The Birmingham corpora	20 Million words
1990s	The British National corpus	100 Million words
Early 21st century	The Bank of English corpus	650 Million words

In our subsequent sections, we will look at a few examples of corpus.

TreeBank Corpus

It may be defined as linguistically parsed text corpus that annotates syntactic or semantic sentence structure. Geoffrey Leech coined the term 'treebank', which represents that the most common way of representing the grammatical analysis is by means of a tree structure. Generally, Treebanks are created on the top of a corpus, which has already been annotated with part-of-speech tags.

Types of TreeBank Corpus

Semantic and Syntactic Treebanks are the two most common types of Treebanks in linguistics. Let us now learn more about these types -

Semantic Treebanks

These Treebanks use a formal representation of sentence's semantic structure. They vary in the depth of their semantic representation. Robot Commands Treebank, Geoquery, Groningen Meaning Bank, RoboCup Corpus are some of the examples of Semantic Treebanks.

Syntactic Treebanks

Opposite to the semantic Treebanks, inputs to the Syntactic Treebank systems are expressions of the formal language obtained from the conversion of parsed Treebank data. The outputs of such systems are predicate logic based meaning representation. Various syntactic Treebanks in different languages have been created so far. For example, **Penn Arabic Treebank**, **Columbia Arabic Treebank** are syntactic Treebanks created in Arabia language. **Sininca** syntactic Treebank created in Chinese language. **Lucy**, **Susane** and **BLLIP WSJ** syntactic corpus created in English language.

Applications of TreeBank Corpus

Followings are some of the applications of TreeBanks:

In Computational Linguistics

If we talk about Computational Linguistic then the best use of TreeBanks is to engineer state-of-the-art natural language processing systems such as part-of-speech taggers, parsers, semantic analyzers and machine translation systems.

In Corpus Linguistics

In case of Corpus linguistics, the best use of Treebanks is to study syntactic phenomena.

In Theoretical Linguistics and Psycholinguistics

The best use of Treebanks in theoretical and psycholinguistics is interaction evidence.

PropBank Corpus

PropBank more specifically called "Proposition Bank" is a corpus, which is annotated with verbal propositions and their arguments. The corpus is a verb-oriented resource; the annotations here are more closely related to the syntactic level. Martha Palmer et al., Department of Linguistic, University of Colorado Boulder developed it. We can use the term PropBank as a common noun referring to any corpus that has been annotated with propositions and their arguments.

In Natural Language Processing (NLP), the PropBank project has played a very significant role. It helps in semantic role labeling.

VerbNet(VN)

VerbNet(VN) is the hierarchical domain-independent and largest lexical resource present in English that incorporates both semantic as well as syntactic information about its contents. VN is a broad-coverage verb lexicon having mappings to other lexical resources such as WordNet, Xtag and FrameNet. It is organized into verb classes extending Levin classes by refinement and addition of subclasses for achieving syntactic and semantic coherence among class members.

Each VerbNet (VN) class contains:

A set of syntactic descriptions or syntactic frames

For depicting the possible surface realizations of the argument structure for constructions such as transitive, intransitive, prepositional phrases, resultatives, and a large set of diathesis alternations.

A set of semantic descriptions such as animate, human, organization

For constraining, the types of thematic roles allowed by the arguments, and further restrictions may be imposed. This will help in indicating the syntactic nature of the constituent likely to be associated with the thematic role.

WordNet

WordNet, created by Princeton is a lexical database for English language. It is the part of the NLTK corpus. In WordNet, nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms called **Synsets**. All the synsets are linked with the help of conceptual-semantic and lexical relations. Its structure makes it very useful for natural language processing (NLP).

In information systems, WordNet is used for various purposes like word-sense disambiguation, information retrieval, automatic text classification and machine translation. One of the most important uses of WordNet is to find out the similarity among words. For this task, various algorithms have been implemented in various packages like Similarity in Perl, NLTK in Python and ADW in Java.

3. Natural Language Processing — Word Level Analysis

In this chapter, we will understand world level analysis in Natural Language Processing.

Regular Expressions

A regular expression (RE) is a language for specifying text search strings. RE helps us to match or find other strings or sets of strings, using a specialized syntax held in a pattern. Regular expressions are used to search texts in UNIX as well as in MS WORD in identical way. We have various search engines using a number of RE features.

Properties of Regular Expressions

Followings are some of the important properties of RE:

- American Mathematician Stephen Cole Kleene formalized the Regular Expression language.

- RE is a formula in a special language, which can be used for specifying simple classes of strings, a sequence of symbols. In other words, we can say that RE is an algebraic notation for characterizing a set of strings.

- Regular expression requires two things, one is the pattern that we wish to search and other is a corpus of text from which we need to search.

Mathematically, A Regular Expression can be defined as follows –

- **ε** is a Regular Expression, which indicates that the language is having an empty string.
- **φ** is a Regular Expression which denotes that it is an empty language.
- If **X** and **Y** are Regular Expressions, then
    - **X, Y**
    - **X.Y(Concatenation of XY)**
    - **X+Y (Union of X and Y)**
    - **X*, Y* (Kleen Closure of X and Y)**

are also regular expressions.

- If a string is derived from above rules then that would also be a regular expression.

Examples of Regular Expressions

The following table shows a few examples of Regular Expressions:

Regular Expressions	**Regular Set**
(0 + 10*)	{0, 1, 10, 100, 1000, 10000, … }
(0*10*)	{1, 01, 10, 010, 0010, …}
(0 + ε)(1 + ε)	{ε, 0, 1, 01}
(a+b)*	It would be set of strings of a's and b's of any length which also includes the null string i.e. {ε, a, b, aa , ab , bb , ba, aaa…….}
(a+b)*abb	It would be set of strings of a's and b's ending with the string abb i.e. {abb, aabb, babb, aaabb, ababb, …………..}
(11)*	It would be set consisting of even number of 1's which also includes an empty string i.e. {ε, 11, 1111, 111111, ……….}
(aa)*(bb)*b	It would be set of strings consisting of even number of a's followed by odd number of b's i.e. {b, aab, aabbb, aabbbbb, aaaab, aaaabbb, …………..}
(aa + ab + ba + bb)*	It would be string of a's and b's of even length that can be obtained by concatenating any combination of the strings aa, ab, ba and bb including null i.e. {aa, ab, ba, bb, aaab, aaba, …………..}

Regular Sets & Their Properties

It may be defined as the set that represents the value of the regular expression and consists specific properties.

Properties of regular sets

- If we do the union of two regular sets then the resulting set would also be regular.
- If we do the intersection of two regular sets then the resulting set would also be regular.

- If we do the complement of regular sets, then the resulting set would also be regular.
- If we do the difference of two regular sets, then the resulting set would also be regular.
- If we do the reversal of regular sets, then the resulting set would also be regular.
- If we take the closure of regular sets, then the resulting set would also be regular.
- If we do the concatenation of two regular sets, then the resulting set would also be regular.

Finite State Automata

The term automata, derived from the Greek word "αὐτόματα" meaning "self-acting", is the plural of automaton which may be defined as an abstract self-propelled computing device that follows a predetermined sequence of operations automatically.

An automaton having a finite number of states is called a Finite Automaton (FA) or Finite State automata (FSA).

Mathematically, an automaton can be represented by a 5-tuple (Q, ∑, δ, q0, F), where −

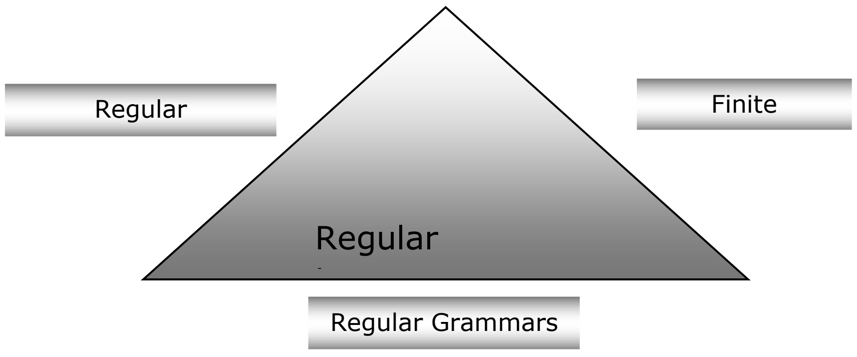
- Q is a finite set of states.
- ∑ is a finite set of symbols, called the alphabet of the automaton.
- δ is the transition function.
- q0 is the initial state from where any input is processed (q0 ∈ Q).
- F is a set of final state/states of Q (F ⊆ Q).

Relation between Finite Automata, Regular Grammars and Regular Expressions

Following points will give us a clear view about the relationship between finite automata, regular grammars and regular expressions:

- As we know that finite state automata are the theoretical foundation of computational work and regular expressions is one way of describing them.
- We can say that any regular expression can be implemented as FSA and any FSA can be described with a regular expression.
- On the other hand, regular expression is a way to characterize a kind of language called regular language. Hence, we can say that regular language can be described with the help of both FSA and regular expression.
- Regular grammar, a formal grammar that can be right-regular or left-regular, is another way to characterize regular language.

Following diagram shows that finite automata, regular expressions and regular grammars are the equivalent ways of describing regular languages.



Types of Finite State Automation (FSA)

Finite state automation is of two types. Let us see what the types are.

Deterministic Finite automation (DFA)

It may be defined as the type of finite automation wherein, for every input symbol we can determine the state to which the machine will move. It has a finite number of states that is why the machine is called Deterministic Finite Automaton (DFA).

Mathematically, a DFA can be represented by a 5-tuple (Q, Σ, δ, q0, F), where −

- Q is a finite set of states.
- Σ is a finite set of symbols, called the alphabet of the automaton.
- δ is the transition function where $\delta: Q \times \Sigma \rightarrow Q$.
- q0 is the initial state from where any input is processed ($q0 \in Q$).
- F is a set of final state/states of Q ($F \subseteq Q$).

Whereas graphically, a DFA can be represented by diagraphs called state diagrams where −

- The states are represented by **vertices**.
- The transitions are shown by labeled **arcs**.
- The initial state is represented by an **empty incoming arc**.
- The final state is represented by **double circle**.

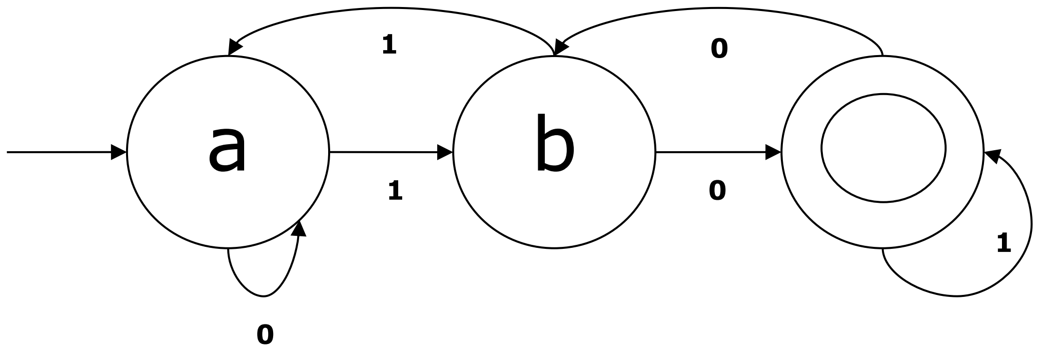
Example of DFA

Suppose a DFA be

- Q = {a, b, c},
- Σ = {0, 1},
- q_0 = {a},
- F = {c},
- Transition function δ is shown in the table as follows:-

Current State	Next State for Input 0	Next State for Input 1
A	a	B
B	c	A
C	b	C

The graphical representation of this DFA would be as follows:



Non-deterministic Finite Automation (NDFA)

It may be defined as the type of finite automation where for every input symbol we cannot determine the state to which the machine will move i.e. the machine can move to any combination of the states. It has a finite number of states that is why the machine is called Non-deterministic Finite Automation (NDFA).

Mathematically, NDFA can be represented by a 5-tuple (Q, ∑, δ, q0, F), where –

- Q is a finite set of states.
- ∑ is a finite set of symbols, called the alphabet of the automaton.
- δ :-is the transition function where δ: $Q \times \Sigma \rightarrow 2^Q$.
- q0 :-is the initial state from where any input is processed ($q0 \in Q$).
- F :-is a set of final state/states of Q ($F \subseteq Q$).

Whereas graphically (same as DFA), a NDFA can be represented by diagraphs called state diagrams where –

- The states are represented by **vertices**.
- The transitions are shown by labeled **arcs**.
- The initial state is represented by an **empty incoming arc**.
- The final state is represented by **double circle**.

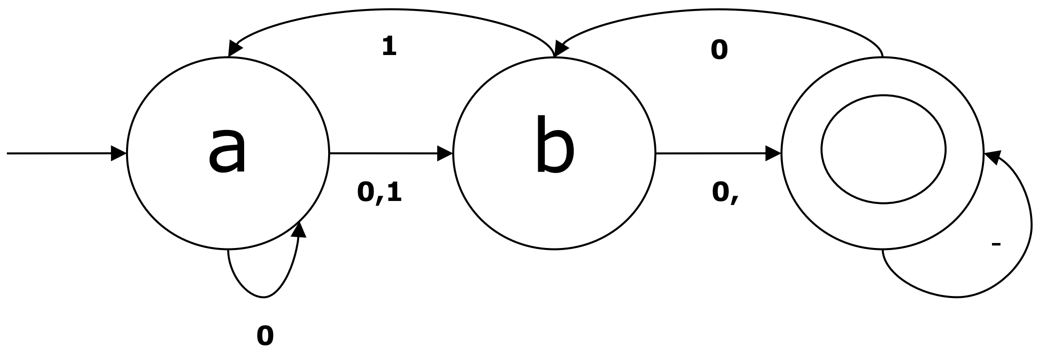
Example of NDFA

Suppose a NDFA be

- Q = {a, b, c},
- ∑ = {0, 1},
- q_0 = {a},
- F = {c},
- Transition function δ is shown in the table as follows: -

Current State	**Next State for Input 0**	**Next State for Input 1**
A	a, b	B
B	C	a, c
C	b, c	C

The graphical representation of this NDFA would be as follows:



Morphological Parsing

The term morphological parsing is related to the parsing of morphemes. We can define morphological parsing as the problem of recognizing that a word breaks down into smaller meaningful units called morphemes producing some sort of linguistic structure for it. For example, we can break the word *foxes* into two, *fox* and *-es*. We can see that the word *foxes,* is made up of two morphemes, one is *fox* and other is *-es.*

In other sense, we can say that morphology is the study of:

- The formation of words.
- The origin of the words.

- Grammatical forms of the words.
- Use of prefixes and suffixes in the formation of words.
- How parts-of-speech (PoS) of a language are formed.

Types of Morphemes

Morphemes, the smallest meaning-bearing units, can be divided into two types:

- Stems
- Word Order

Stems

It is the core meaningful unit of a word. We can also say that it is the root of the word. For example, in the word foxes, the stem is fox.

- **Affixes:** As the name suggests, they add some additional meaning and grammatical functions to the words. For example, in the word foxes, the affix is -es.

Further, affixes can also be divided into following four types:

- **Prefixes:** As the name suggests, prefixes precede the stem. For example, in the word unbuckle, un is the prefix.

- **Suffixes:** As the name suggests, suffixes follow the stem. For example, in the word cats, -s is the suffix.

- **Infixes:** As the name suggests, infixes are inserted inside the stem. For example, the word cupful, can be pluralized as cupsful by using -s as the infix.

- **Circumfixes:** They precede and follow the stem. There are very less examples of circumfixes in English language. A very common example is 'A-ing' where we can use -A precede and -ing follows the stem.

Word Order

The order of the words would be decided by morphological parsing. Let us now see the requirements for building a morphological parser:

Lexicon

The very first requirement for building a morphological parser is lexicon, which includes the list of stems and affixes along with the basic information about them. For example, the information like whether the stem is Noun stem or Verb stem, etc.

Morphotactics

It is basically the model of morpheme ordering. In other sense, the model explaining which classes of morphemes can follow other classes of morphemes inside a word. For example, the morphotactic fact is that the English plural morpheme always follows the noun rather than preceding it.

Orthographic rules

These spelling rules are used to model the changes occurring in a word. For example, the rule of converting y to ie in word like city+s = cities not citys.

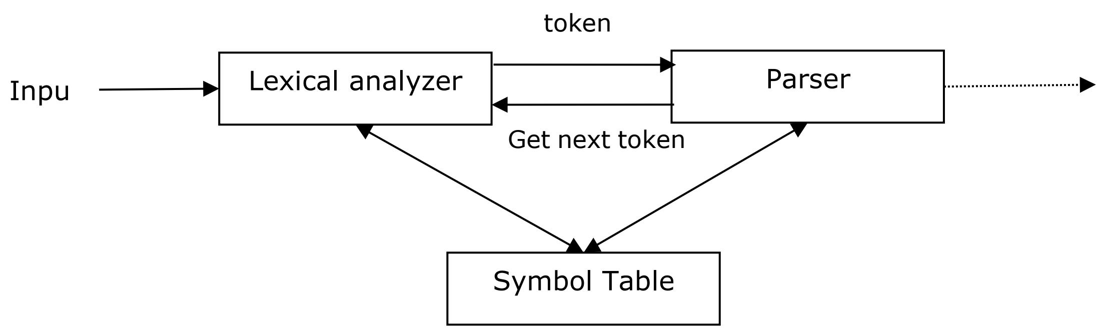
4. Natural Language Processing — Syntactic Analysis

Syntactic analysis or parsing or syntax analysis is the third phase of NLP. The purpose of this phase is to draw exact meaning, or you can say dictionary meaning from the text. Syntax analysis checks the text for meaningfulness comparing to the rules of formal grammar. For example, the sentence like "hot ice-cream" would be rejected by semantic analyzer.

In this sense, syntactic analysis or parsing may be defined as the process of analyzing the strings of symbols in natural language conforming to the rules of formal grammar. The origin of the word ***'parsing'*** is from Latin word ***'pars'*** which means ***'part'***.

Concept of Parser

It is used to implement the task of parsing. It may be defined as the software component designed for taking input data (text) and giving structural representation of the input after checking for correct syntax as per formal grammar. It also builds a data structure generally in the form of parse tree or abstract syntax tree or other hierarchical structure.



The main roles of the parse include:

- To report any syntax error.
- To recover from commonly occurring error so that the processing of the remainder of program can be continued.
- To create parse tree.
- To create symbol table.
- To produce intermediate representations (IR).

Types of Parsing

Derivation divides parsing into the followings two types:

- Top-down Parsing
- Bottom-up parsing

Top-down Parsing

In this kind of parsing, the parser starts constructing the parse tree from the start symbol and then tries to transform the start symbol to the input. The most common form of top-down parsing uses recursive procedure to process the input. The main disadvantage of recursive descent parsing is backtracking.

Bottom-up Parsing

In this kind of parsing, the parser starts with the input symbol and tries to construct the parser tree up to the start symbol.

Concept of Derivation

In order to get the input string, we need a sequence of production rules. Derivation is a set of production rules. During parsing, we need to decide the non-terminal, which is to be replaced along with deciding the production rule with the help of which the non-terminal will be replaced.

Types of Derivation

In this section, we will learn about the two types of derivations, which can be used to decide which non-terminal to be replaced with production rule:

Left-most Derivation

In the left-most derivation, the sentential form of an input is scanned and replaced from the left to the right. The sentential form in this case is called the left-sentential form.

Right-most Derivation

In the left-most derivation, the sentential form of an input is scanned and replaced from right to left. The sentential form in this case is called the right-sentential form.

Concept of Parse Tree

It may be defined as the graphical depiction of a derivation. The start symbol of derivation serves as the root of the parse tree. In every parse tree, the leaf nodes are terminals and interior nodes are non-terminals. A property of parse tree is that in-order traversal will produce the original input string.

Concept of Grammar

Grammar is very essential and important to describe the syntactic structure of well-formed programs. In the literary sense, they denote syntactical rules for conversation in natural languages. Linguistics have attempted to define grammars since the inception of natural languages like English, Hindi, etc.

The theory of formal languages is also applicable in the fields of Computer Science mainly in programming languages and data structure. For example, in 'C' language, the precise grammar rules state how functions are made from lists and statements.

A mathematical model of grammar was given by **Noam Chomsky** in 1956, which is effective for writing computer languages.

Mathematically, a grammar G can be formally written as a 4-tuple (N, T, S, P) where –

- **N** or **V_N** = set of non-terminal symbols, i.e., variables.
- **T** or **Σ** = set of terminal symbols.
- **S** = Start symbol where $S \in N$
- **P** denotes the Production rules for Terminals as well as Non-terminals. It has the form $\alpha \rightarrow \beta$, where α and β are strings on $V_N \cup \Sigma$ and least one symbol of α belongs to V_N.

Phrase Structure or Constituency Grammar

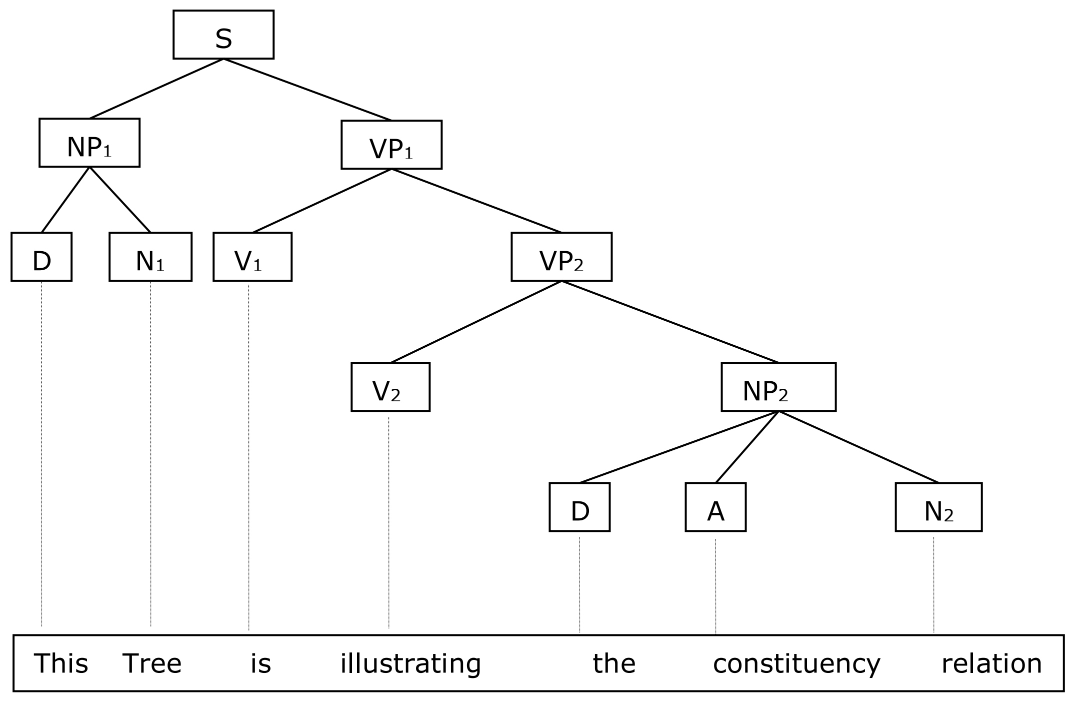
Phrase structure grammar, introduced by Noam Chomsky, is based on the constituency relation. That is why it is also called constituency grammar. It is opposite to dependency grammar.

Example

Before giving an example of constituency grammar, we need to know the fundamental points about constituency grammar and constituency relation.

- All the related frameworks view the sentence structure in terms of constituency relation.
- The constituency relation is derived from the subject-predicate division of Latin as well as Greek grammar.
- The basic clause structure is understood in terms of **noun phrase NP** and **verb phrase VP**.

We can write the sentence **"This tree is illustrating the constituency relation"** as follows:



Dependency Grammar

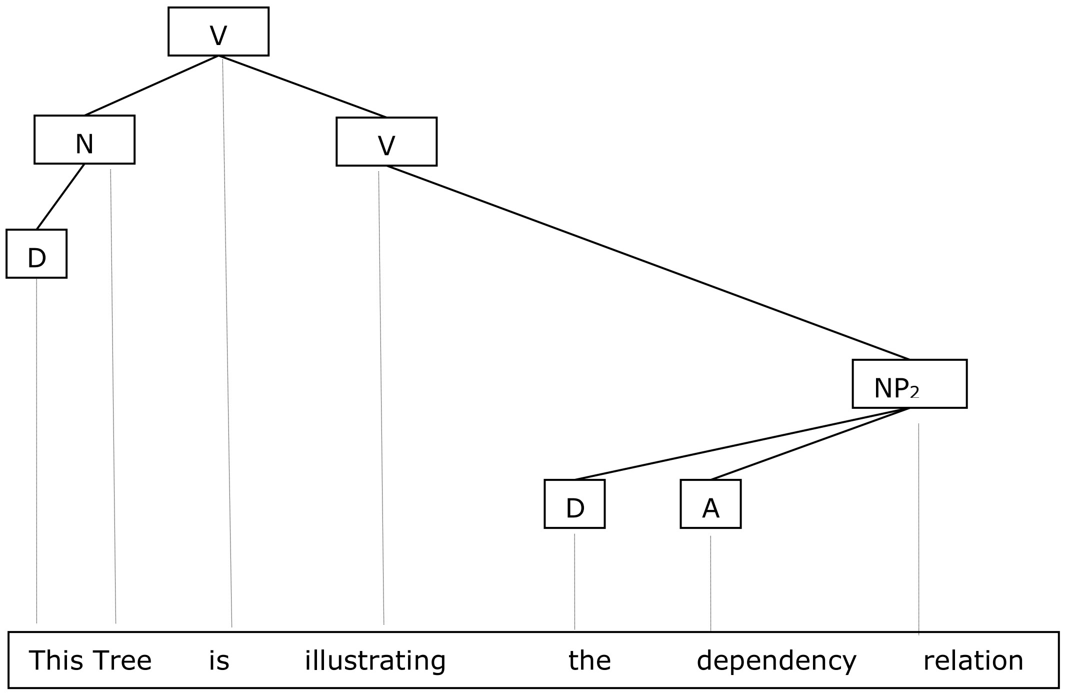
It is opposite to the constituency grammar and based on dependency relation. It was introduced by Lucien Tesniere. Dependency grammar (DG) is opposite to the constituency grammar because it lacks phrasal nodes.

Example

Before giving an example of Dependency grammar, we need to know the fundamental points about Dependency grammar and Dependency relation.

- In DG, the linguistic units, i.e., words are connected to each other by directed links.
- The verb becomes the center of the clause structure.
- Every other syntactic units are connected to the verb in terms of directed link. These syntactic units are called ***dependencies***.

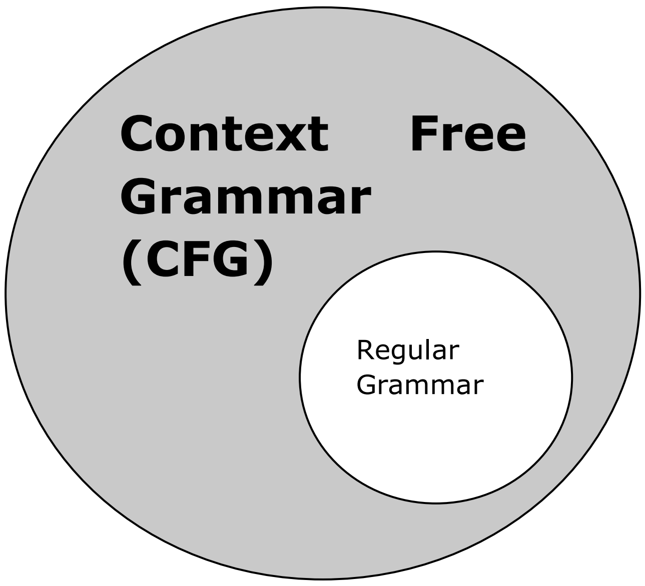
We can write the sentence **"This tree is illustrating the dependency relation"** as follows;



Parse tree that uses Constituency grammar is called constituency-based parse tree; and the parse trees that uses dependency grammar is called dependency-based parse tree.

Context Free Grammar

Context free grammar, also called CFG, is a notation for describing languages and a superset of Regular grammar. It can be seen in the following diagram:



Definition of CFG

CFG consists of finite set of grammar rules with the following four components:

Set of Non-terminals

It is denoted by V. The non-terminals are syntactic variables that denote the sets of strings, which further help defining the language, generated by the grammar.

Set of Terminals

It is also called tokens and defined by Σ. Strings are formed with the basic symbols of terminals.

Set of Productions

It is denoted by P. The set defines how the terminals and non-terminals can be combined. Every production(P) consists of non-terminals, an arrow, and terminals (the sequence of terminals). Non-terminals are called the left side of the production and terminals are called the right side of the production.

Start Symbol

The production begins from the start symbol. It is denoted by symbol S. Non-terminal symbol is always designated as start symbol.

5. Natural Language Processing — Semantic Analysis

The purpose of semantic analysis is to draw exact meaning, or you can say dictionary meaning from the text. The work of semantic analyzer is to check the text for meaningfulness.

We already know that lexical analysis also deals with the meaning of the words, then how is semantic analysis different from lexical analysis? Lexical analysis is based on smaller token but on the other side semantic analysis focuses on larger chunks. That is why semantic analysis can be divided into the following two parts:

Studying meaning of individual word

It is the first part of the semantic analysis in which the study of the meaning of individual words is performed. This part is called lexical semantics.

Studying the combination of individual words

In the second part, the individual words will be combined to provide meaning in sentences.

The most important task of semantic analysis is to get the proper meaning of the sentence. For example, analyze the sentence **"Ram is great."** In this sentence, the speaker is talking either about Lord Ram or about a person whose name is Ram. That is why the job, to get the proper meaning of the sentence, of semantic analyzer is important.

Elements of Semantic Analysis

Followings are some important elements of semantic analysis:

Hyponymy

It may be defined as the relationship between a generic term and instances of that generic term. Here the generic term is called hypernym and its instances are called hyponyms. For example, the word color is hypernym and the color blue, yellow etc. are hyponyms.

Homonymy

It may be defined as the words having same spelling or same form but having different and unrelated meaning. For example, the word "Bat" is a homonymy word because bat can be an implement to hit a ball or bat is a nocturnal flying mammal also.

Polysemy

Polysemy is a Greek word, which means "many signs". It is a word or phrase with different but related sense. In other words, we can say that polysemy has the same spelling but different and related meaning. For example, the word "bank" is a polysemy word having the following meanings:

- a financial institution.
- the building in which such an institution is located.

- a synonym for "to rely on".

Difference between Polysemy and Homonymy

Both polysemy and homonymy words have the same syntax or spelling. The main difference between them is that in polysemy, the meanings of the words are related but in homonymy, the meanings of the words are not related. For example, if we talk about the same word "Bank", we can write the meaning 'a financial institution' or 'a river bank'. In that case it would be the example of homonym because the meanings are unrelated to each other.

Synonymy

It is the relation between two lexical items having different forms but expressing the same or a close meaning. Examples are 'author/writer', 'fate/destiny'.

Antonymy

It is the relation between two lexical items having symmetry between their semantic components relative to an axis. The scope of antonymy is as follows:

- **Application of property or not:** Example is 'life/death', 'certitude/incertitude'.
- **Application of scalable property:** Example is 'rich/poor', 'hot/cold'.
- **Application of a usage:** Example is 'father/son', 'moon/sun'.

Meaning Representation

Semantic analysis creates a representation of the meaning of a sentence. But before getting into the concept and approaches related to meaning representation, we need to understand the building blocks of semantic system.

Building Blocks of Semantic System

In word representation or representation of the meaning of the words, the following building blocks play an important role:

- **Entities:** It represents the individual such as a particular person, location etc. For example, Haryana. India, Ram all are entities.

- **Concepts:** It represents the general category of the individuals such as a person, city, etc.

- **Relations:** It represents the relationship between entities and concept. For example, Ram is a person.

- **Predicates:** It represents the verb structures. For example, semantic roles and case grammar are the examples of predicates.

Now, we can understand that meaning representation shows how to put together the building blocks of semantic systems. In other words, it shows how to put together entities, concepts, relation and predicates to describe a situation. It also enables the reasoning about the semantic world.

Approaches to Meaning Representations

Semantic analysis uses the following approaches for the representation of meaning:

- First order predicate logic (FOPL)
- Semantic Nets
- Frames
- Conceptual dependency (CD)
- Rule-based architecture
- Case Grammar
- Conceptual Graphs

Need of Meaning Representations

A question that arises here is why do we need meaning representation? Followings are the reasons for the same:

Linking of linguistic elements to non-linguistic elements

The very first reason is that with the help of meaning representation the linking of linguistic elements to the non-linguistic elements can be done.

Representing variety at lexical level

With the help of meaning representation, unambiguous, canonical forms can be represented at the lexical level.

Can be used for reasoning

Meaning representation can be used to reason for verifying what is true in the world as well as to infer the knowledge from the semantic representation.

Lexical Semantics

The first part of semantic analysis, studying the meaning of individual words is called lexical semantics. It includes words, sub-words, affixes (sub-units), compound words and phrases also. All the words, sub-words, etc. are collectively called **lexical items**. In other words, we can say that lexical semantics is the relationship between lexical items, meaning of sentences and syntax of sentence.

Following are the steps involved in lexical semantics:

- Classification of lexical items like words, sub-words, affixes, etc. is performed in lexical semantics.

- Decomposition of lexical items like words, sub-words, affixes, etc. is performed in lexical semantics.

- Differences as well as similarities between various lexical semantic structures is also analyzed.

6. Natural Language Processing — Word Sense Disambiguation

We understand that words have different meanings based on the context of its usage in the sentence. If we talk about human languages, then they are ambiguous too because many words can be interpreted in multiple ways depending upon the context of their occurrence.

Word sense disambiguation, in natural language processing (NLP), may be defined as the ability to determine which meaning of word is activated by the use of word in a particular context. Lexical ambiguity, syntactic or semantic, is one of the very first problem that any NLP system faces. Part-of-speech (POS) taggers with high level of accuracy can solve Word's syntactic ambiguity. On the other hand, the problem of resolving semantic ambiguity is called WSD (word sense disambiguation). Resolving semantic ambiguity is harder than resolving syntactic ambiguity.

For example, consider the two examples of the distinct sense that exist for the word ***"bass":***

- I can hear bass sound.
- He likes to eat grilled bass.

The occurrence of the word **bass** clearly denotes the distinct meaning. In first sentence, it means **frequency** and in second, it means **fish**. Hence, if it would be disambiguated by WSD then the correct meaning to the above sentences can be assigned as follows:

- I can hear bass/frequency sound.
- He likes to eat grilled bass/fish.

Evaluation of WSD

The evaluation of WSD requires the following two inputs:

A Dictionary

The very first input for evaluation of WSD is dictionary, which is used to specify the senses to be disambiguated.

Test Corpus

Another input required by WSD is the high-annotated test corpus that has the target or correct-senses. The test corpora can be of two types:

- **Lexical sample:** This kind of corpora is used in the system, where it is required to disambiguate a small sample of words.
- **All-words:** This kind of corpora is used in the system, where it is expected to disambiguate all the words in a piece of running text.

Approaches and Methods to Word Sense Disambiguation (WSD)

Approaches and methods to WSD are classified according to the source of knowledge used in word disambiguation.

Let us now see the four conventional methods to WSD:

Dictionary-based or Knowledge-based Methods

As the name suggests, for disambiguation, these methods primarily rely on dictionaries, treasures and lexical knowledge base. They do not use corpora evidences for disambiguation. The Lesk method is the seminal dictionary-based method introduced by Michael Lesk in 1986. The Lesk definition, on which the Lesk algorithm is based is "**measure overlap between sense definitions for all words in context**". However, in 2000, Kilgarriff and Rosensweig gave the simplified Lesk definition as "**measure overlap between sense definitions of word and current context**", which further means identify the correct sense for one word at a time. Here the current context is the set of words in surrounding sentence or paragraph.

Supervised Methods

For disambiguation, machine learning methods make use of sense-annotated corpora to train. These methods assume that the context can provide enough evidence on its own to disambiguate the sense. In these methods, the words knowledge and reasoning are deemed unnecessary. The context is represented as a set of "features" of the words. It includes the information about the surrounding words also. Support vector machine and memory-based learning are the most successful supervised learning approaches to WSD. These methods rely on substantial amount of manually sense-tagged corpora, which is very expensive to create.

Semi-supervised Methods

Due to the lack of training corpus, most of the word sense disambiguation algorithms use semi-supervised learning methods. It is because semi-supervised methods use both labelled as well as unlabeled data. These methods require very small amount of annotated text and large amount of plain unannotated text. The technique that is used by semi-supervised methods is bootstrapping from seed data.

Unsupervised Methods

These methods assume that similar senses occur in similar context. That is why the senses can be induced from text by clustering word occurrences by using some measure of similarity of the context. This task is called word sense induction or discrimination. Unsupervised methods have great potential to overcome the knowledge acquisition bottleneck due to non-dependency on manual efforts.

Applications of Word Sense Disambiguation (WSD)

Word sense disambiguation (WSD) is applied in almost every application of language technology.

Let us now see the scope of WSD:

Machine Translation

Machine translation or MT is the most obvious application of WSD. In MT, Lexical choice for the words that have distinct translations for different senses, is done by WSD. The senses in MT are represented as words in the target language. Most of the machine translation systems do not use explicit WSD module.

Information Retrieval (IR)

Information retrieval (IR) may be defined as a software program that deals with the organization, storage, retrieval and evaluation of information from document repositories particularly textual information. The system basically assists users in finding the information they required but it does not explicitly return the answers of the questions. WSD is used to resolve the ambiguities of the queries provided to IR system. As like MT, current IR systems do not explicitly use WSD module and they rely on the concept that user would type enough context in the query to only retrieve relevant documents.

Text Mining and Information Extraction (IE)

In most of the applications, WSD is necessary to do accurate analysis of text. For example, WSD helps intelligent gathering system to do flagging of the correct words. For example, medical intelligent system might need flagging of "illegal drugs" rather than "medical drugs".

Lexicography

WSD and lexicography can work together in loop because modern lexicography is corpus-based. With lexicography, WSD provides rough empirical sense groupings as well as statistically significant contextual indicators of sense.

Difficulties in Word Sense Disambiguation (WSD)

Followings are some difficulties faced by word sense disambiguation (WSD):

Differences between dictionaries

The major problem of WSD is to decide the sense of the word because different senses can be very closely related. Even different dictionaries and thesauruses can provide different divisions of words into senses.

Different algorithms for different applications

Another problem of WSD is that completely different algorithm might be needed for different applications. For example, in machine translation, it takes the form of target word selection; and in information retrieval, a sense inventory is not required.

Inter-judge variance

Another problem of WSD is that WSD systems are generally tested by having their results on a task compared against the task of human beings. This is called the problem of inter-judge variance.

Word-sense discreteness

Another difficulty in WSD is that words cannot be easily divided into discrete sub-meanings.

7. Natural Language Processing — Discourse Processing

The most difficult problem of AI is to process the natural language by computers or in other words *natural language processing* is the most difficult problem of artificial intelligence. If we talk about the major problems in NLP, then one of the major problems in NLP is discourse processing – building theories and models of how utterances stick together to form **coherent discourse**. Actually, the language always consists of collocated, structured and coherent groups of sentences rather than isolated and unrelated sentences like movies. These coherent groups of sentences are referred to as discourse.

Concept of Coherence

Coherence and discourse structure are interconnected in many ways. Coherence, along with property of good text, is used to evaluate the output quality of natural language generation system. The question that arises here is what does it mean for a text to be coherent? Suppose we collected one sentence from every page of the newspaper, then will it be a discourse? Of-course, not. It is because these sentences do not exhibit coherence. The coherent discourse must possess the following properties:

Coherence relation between utterances

The discourse would be coherent if it has meaningful connections between its utterances. This property is called coherence relation. For example, some sort of explanation must be there to justify the connection between utterances.

Relationship between entities

Another property that makes a discourse coherent is that there must be a certain kind of relationship with the entities. Such kind of coherence is called entity-based coherence.

Discourse structure

An important question regarding discourse is what kind of structure the discourse must have. The answer to this question depends upon the segmentation we applied on discourse. Discourse segmentations may be defined as determining the types of structures for large discourse. It is quite difficult to implement discourse segmentation, but it is very important for **information retrieval**, **text summarization** and **information extraction** kind of applications.

Algorithms for Discourse Segmentation

In this section, we will learn about the algorithms for discourse segmentation. The algorithms are described below:

Unsupervised Discourse Segmentation

The class of unsupervised discourse segmentation is often represented as linear segmentation. We can understand the task of linear segmentation with the help of an example. In the example, there is a task of segmenting the text into multi-paragraph units; the units represent the passage of the original text. These algorithms are dependent on cohesion that may be defined as the use of certain linguistic devices to tie the textual units together. On the other hand, lexicon cohesion is the cohesion that is indicated by the relationship between two or more words in two units like the use of synonyms.

Supervised Discourse Segmentation

The earlier method does not have any hand-labeled segment boundaries. On the other hand, supervised discourse segmentation needs to have boundary-labeled training data. It is very easy to acquire the same. In supervised discourse segmentation, discourse marker or cue words play an important role. Discourse marker or cue word is a word or phrase that functions to signal discourse structure. These discourse markers are domain-specific.

Text Coherence

Lexical repetition is a way to find the structure in a discourse, but it does not satisfy the requirement of being coherent discourse. To achieve the coherent discourse, we must focus on coherence relations in specific. As we know that coherence relation defines the possible connection between utterances in a discourse. Hebb has proposed such kind of relations as follows:

We are taking two terms $\mathbf{S_0}$ and $\mathbf{S_1}$ to represent the meaning of the two related sentences:

Result

It infers that the state asserted by term $\mathbf{S_0}$ could cause the state asserted by $\mathbf{S_1}$. For example, two statements show the relationship result: Ram was caught in the fire. His skin burned.

Explanation

It infers that the state asserted by $\mathbf{S_1}$ could cause the state asserted by $\mathbf{S_0}$. For example, two statements show the relationship – Ram fought with Shyam's friend. He was drunk.

Parallel

It infers p(a1,a2,…) from assertion of $\mathbf{S_0}$ and p(b1,b2,…) from assertion $\mathbf{S_1}$. Here ai and bi are similar for all i. For example, two statements are parallel – Ram wanted car. Shyam wanted money.

Elaboration

It infers the same proposition P from both the assertions – $\mathbf{S_0}$ and $\mathbf{S_1}$. For example, two statements show the relation elaboration: Ram was from Chandigarh. Shyam was from Kerala.

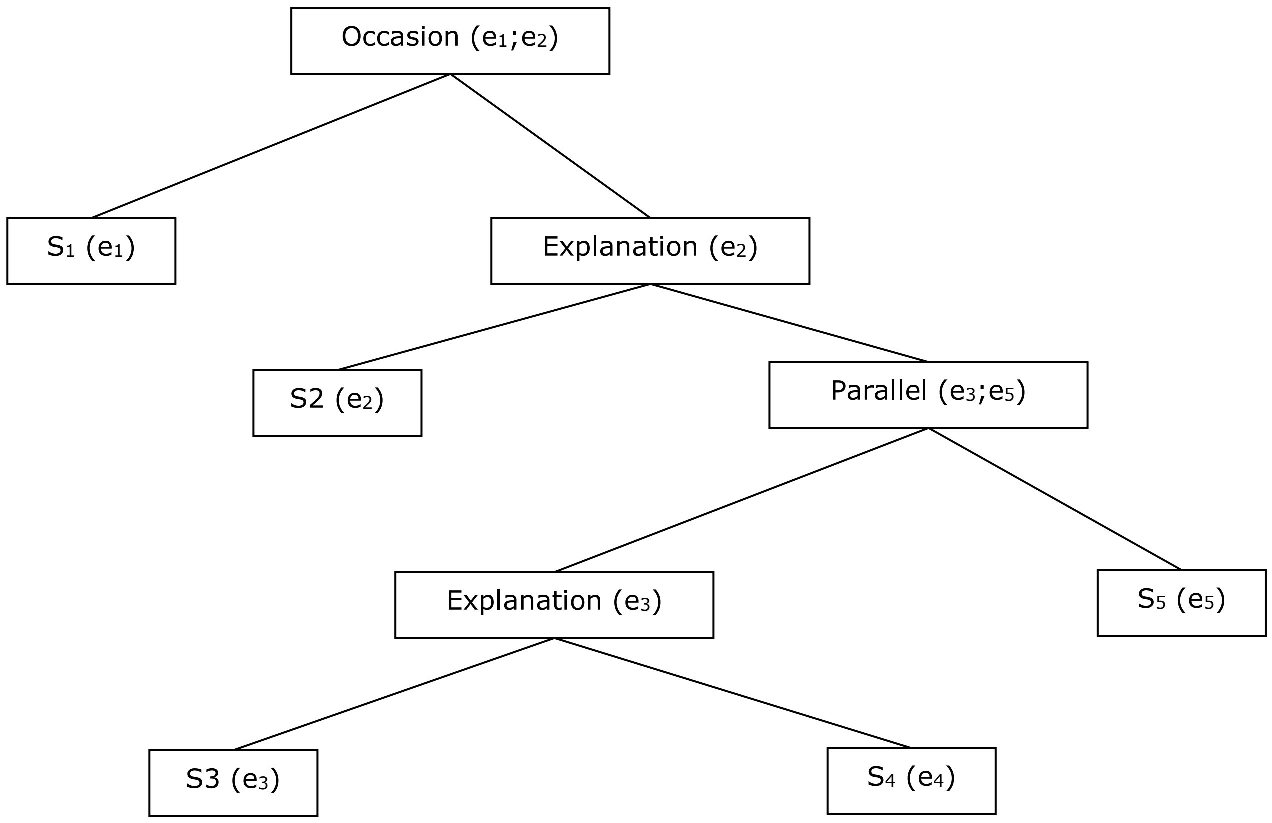
Occasion

It happens when a change of state can be inferred from the assertion of $\mathbf{S_0}$, final state of which can be inferred from $\mathbf{S_1}$ and vice-versa. For example, the two statements show the relation occasion: Ram picked up the book. He gave it to Shyam.

Building Hierarchical Discourse Structure

The coherence of entire discourse can also be considered by hierarchical structure between coherence relations. For example, the following passage can be represented as hierarchical structure:

- **S_1:** Ram went to the bank to deposit money.
- **S_2:** He then took a train to Shyam's cloth shop.
- **S_3:** He wanted to buy some clothes.
- **S_4:** He do not have new clothes for party.
- **S_5:** He also wanted to talk to Shyam regarding his health.



Reference Resolution

Interpretation of the sentences from any discourse is another important task and to achieve this we need to know who or what entity is being talked about. Here, interpretation reference is the key element. **Reference** may be defined as the linguistic expression to denote an entity or individual. For example, in the passage, Ram, the manager of ABC bank, saw his friend Shyam at a shop. He went to meet him, the linguistic expressions like Ram, His, He are reference.

On the same note, **reference resolution** may be defined as the task of determining what entities are referred to by which linguistic expression.

Terminology Used in Reference Resolution

We use the following terminologies in reference resolution:

- **Referring expression:** The natural language expression that is used to perform reference is called a referring expression. For example, the passage used above is a referring expression.

- **Referent:** It is the entity that is referred. For example, in the last given example Ram is a referent.

- **Corefer:** When two expressions are used to refer to the same entity, they are called corefers. For example, ***Ram*** and ***he*** are corefers.

- **Antecedent:** The term has the license to use another term. For example, ***Ram*** is the antecedent of the reference ***he***.

- **Anaphora & Anaphoric:** It may be defined as the reference to an entity that has been previously introduced into the sentence. And, the referring expression is called anaphoric.

- **Discourse model:** The model that contains the representations of the entities that have been referred to in the discourse and the relationship they are engaged in.

Types of Referring Expressions

Let us now see the different types of referring expressions. The five types of referring expressions are described below:

Indefinite Noun Phrases

Such kind of reference represents the entities that are new to the hearer into the discourse context. For example – in the sentence Ram had gone around one day to bring him some food – some is an indefinite reference.

Definite Noun Phrases

Opposite to above, such kind of reference represents the entities that are not new or identifiable to the hearer into the discourse context. For example, in the sentence – I used to read The Times of India – The Times of India is a definite reference.

Pronouns

It is a form of definite reference. For example, Ram laughed as loud as he could. The word **he** represents pronoun referring expression.

Demonstratives

These demonstrate and behave differently than simple definite pronouns. For example, this and that are demonstrative pronouns.

Names

It is the simplest type of referring expression. It can be the name of a person, organization and location also. For example, in the above examples, Ram is the name-refereeing expression.

Reference Resolution Tasks

The two reference resolution tasks are described below.

Coreference Resolution

It is the task of finding referring expressions in a text that refer to the same entity. In simple words, it is the task of finding corefer expressions. A set of coreferring expressions are called coreference chain. For example – He, Chief Manager and His – these are referring expressions in the first passage given as example.

Constraint on Coreference Resolution

In English, the main problem for coreference resolution is the pronoun **it**. The reason behind this is that the pronoun **it** has many uses. For example, **it** can refer much like he and she. The pronoun **it** also refers to the things that do not refer to specific things. For example, It's raining. It is really good.

Pronominal Anaphora Resolution

Unlike the coreference resolution, pronominal anaphora resolution may be defined as the task of finding the antecedent for a single pronoun. For example, the pronoun is **his** and the task of pronominal anaphora resolution is to find the word Ram because Ram is the antecedent.

8. Natural Language Processing — Part of Speech (PoS) Tagging

Tagging is a kind of classification that may be defined as the automatic assignment of description to the tokens. Here the descriptor is called tag, which may represent one of the part-of-speech, semantic information and so on.

Now, if we talk about Part-of-Speech (PoS) tagging, then it may be defined as the process of assigning one of the parts of speech to the given word. It is generally called POS tagging. In simple words, we can say that POS tagging is a task of labelling each word in a sentence with its appropriate part of speech. We already know that parts of speech include nouns, verb, adverbs, adjectives, pronouns, conjunction and their sub-categories.

Most of the POS tagging falls under Rule Base POS tagging, Stochastic POS tagging and Transformation based tagging.

Rule-based POS Tagging

One of the oldest techniques of tagging is rule-based POS tagging. Rule-based taggers use dictionary or lexicon for getting possible tags for tagging each word. If the word has more than one possible tag, then rule-based taggers use hand-written rules to identify the correct tag. Disambiguation can also be performed in rule-based tagging by analyzing the linguistic features of a word along with its preceding as well as following words. For example, suppose if the preceding word of a word is article then word must be a noun.

As the name suggests, all such kind of information in rule-based POS tagging is coded in the form of rules. These rules may be either:

- Context-pattern rules

- Or, as Regular expression compiled into finite-state automata, intersected with lexically ambiguous sentence representation.

We can also understand Rule-based POS tagging by its two-stage architecture:

- **First stage:** In the first stage, it uses a dictionary to assign each word a list of potential parts-of-speech.

- **Second stage:** In the second stage, it uses large lists of hand-written disambiguation rules to sort down the list to a single part-of-speech for each word.

Properties of Rule-Based POS Tagging

Rule-based POS taggers possess the following properties:

- These taggers are knowledge-driven taggers.
- The rules in Rule-based POS tagging are built manually.
- The information is coded in the form of rules.
- We have some limited number of rules approximately around 1000.
- Smoothing and language modeling is defined explicitly in rule-based taggers.

Stochastic POS Tagging

Another technique of tagging is Stochastic POS Tagging. Now, the question that arises here is which model can be stochastic. The model that includes frequency or probability (statistics) can be called stochastic. Any number of different approaches to the problem of part-of-speech tagging can be referred to as stochastic tagger.

The simplest stochastic tagger applies the following approaches for POS tagging:

Word Frequency Approach

In this approach, the stochastic taggers disambiguate the words based on the probability that a word occurs with a particular tag. We can also say that the tag encountered most frequently with the word in the training set is the one assigned to an ambiguous instance of that word. The main issue with this approach is that it may yield inadmissible sequence of tags.

Tag Sequence Probabilities

It is another approach of stochastic tagging, where the tagger calculates the probability of a given sequence of tags occurring. It is also called n-gram approach. It is called so because the best tag for a given word is determined by the probability at which it occurs with the n previous tags.

Properties of Stochastic POS Tagging

Stochastic POS taggers possess the following properties:

- This POS tagging is based on the probability of tag occurring.
- It requires training corpus.
- There would be no probability for the words that do not exist in the corpus.
- It uses different testing corpus (other than training corpus).
- It is the simplest POS tagging because it chooses most frequent tags associated with a word in training corpus.

Transformation-based Tagging

Transformation based tagging is also called Brill tagging. It is an instance of the transformation-based learning (TBL), which is a rule-based algorithm for automatic tagging of POS to the given text. TBL, allows us to have linguistic knowledge in a readable form, transforms one state to another state by using transformation rules.

It draws the inspiration from both the previous explained taggers – rule-based and stochastic. If we see similarity between rule-based and transformation tagger, then like rule-based, it is also based on the rules that specify what tags need to be assigned to what words. On the other hand, if we see similarity between stochastic and transformation tagger then like stochastic, it is machine learning technique in which rules are automatically induced from data.

Working of Transformation Based Learning (TBL)

In order to understand the working and concept of transformation-based taggers, we need to understand the working of transformation-based learning. Consider the following steps to understand the working of TBL:

- **Start with the solution:** The TBL usually starts with some solution to the problem and works in cycles.

- **Most beneficial transformation chosen:** In each cycle, TBL will choose the most beneficial transformation.

- **Apply to the problem:** The transformation chosen in the last step will be applied to the problem.

The algorithm will stop when the selected transformation in step 2 will not add either more value or there are no more transformations to be selected. Such kind of learning is best suited in classification tasks.

Advantages of Transformation-based Learning (TBL)

The advantages of TBL are as follows:

- We learn small set of simple rules and these rules are enough for tagging.

- Development as well as debugging is very easy in TBL because the learned rules are easy to understand.

- Complexity in tagging is reduced because in TBL there is interlacing of machine-learned and human-generated rules.

- Transformation-based tagger is much faster than Markov-model tagger.

Disadvantages of Transformation-based Learning (TBL)

The disadvantages of TBL are as follows:

- Transformation-based learning (TBL) does not provide tag probabilities.
- In TBL, the training time is very long especially on large corpora.

Hidden Markov Model (HMM) POS Tagging

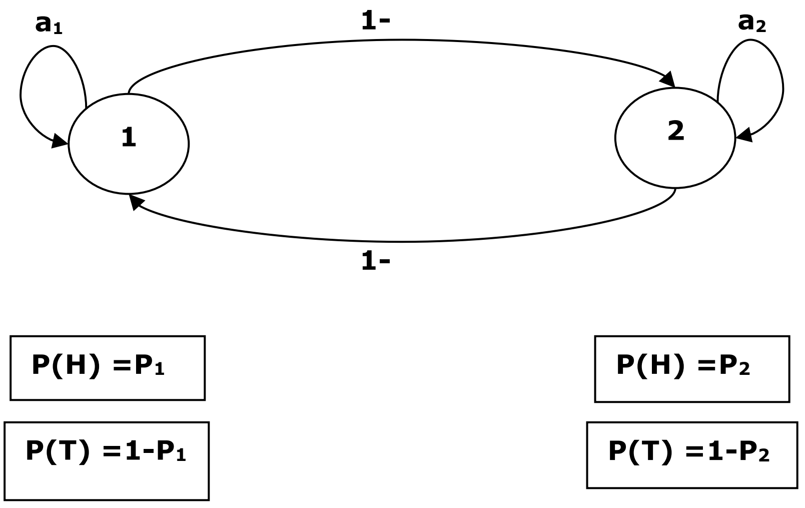
Before digging deep into HMM POS tagging, we must understand the concept of Hidden Markov Model (HMM).

Hidden Markov Model

An HMM model may be defined as the doubly-embedded stochastic model, where the underlying stochastic process is hidden. This hidden stochastic process can only be observed through another set of stochastic processes that produces the sequence of observations.

Example

For example, a sequence of hidden coin tossing experiments is done and we see only the observation sequence consisting of heads and tails. The actual details of the process – how many coins used, the order in which they are selected – are hidden from us. By observing this sequence of heads and tails, we can build several HMMs to explain the sequence. Following is one form of Hidden Markov Model for this problem:



We assumed that there are two states in the HMM and each of the state corresponds to the selection of different biased coin. Following matrix gives the state transition probabilities:

$$A = \begin{bmatrix} a11 & a12 \\ a21 & a22 \end{bmatrix}$$

Here,

- **a_{ij}** = probability of transition from one state to another – from i to j.
- **a_{11} + a_{12}** = 1 and a_{21} + a_{22} =1
- **P_1** = probability of heads of the first coin i.e. the bias of the first coin.
- **P_2** = probability of heads of the second coin i.e. the bias of the second coin.

We can also create an HMM model assuming that there are 3 coins or more.

This way, we can characterize HMM by the following elements:

- N, the number of states in the model (in the above example N =2, only two states).
- M, the number of distinct observations that can appear with each state in the above example M = 2, i.e., H or T).
- A, the state transition probability distribution – the matrix A in the above example.
- P, the probability distribution of the observable symbols in each state (in our example P1 and P2).

- I, the initial state distribution.

Use of HMM for POS Tagging

The POS tagging process is the process of finding the sequence of tags which is most likely to have generated a given word sequence. We can model this POS process by using a Hidden Markov Model (HMM), where **tags** are the **hidden states** that produced the **observable output**, i.e., the **words**.

Mathematically, in POS tagging, we are always interested in finding a tag sequence (C) which maximizes:

P (C|W)

Where,

$C = C_1, C_2, C_3 \ldots C_T$

$W = W_1, W2, W_3, W_T$

On the other side of coin, the fact is that we need a lot of statistical data to reasonably estimate such kind of sequences. However, to simplify the problem, we can apply some mathematical transformations along with some assumptions.

The use of HMM to do a POS tagging is a special case of Bayesian interference. Hence, we will start by restating the problem using Bayes' rule, which says that the above-mentioned conditional probability is equal to -

(PROB (C_1,..., CT) * PROB (W_1,..., WT | C_1,..., CT)) / PROB (W_1,..., WT)

We can eliminate the denominator in all these cases because we are interested in finding the sequence C which maximizes the above value. This will not affect our answer. Now, our problem reduces to finding the sequence C that maximizes -

PROB (C_1,..., CT) * PROB (W_1,..., WT | C_1,..., CT) (1)

Even after reducing the problem in the above expression, it would require large amount of data. We can make reasonable independence assumptions about the two probabilities in the above expression to overcome the problem.

First Assumption

The probability of a tag depends on the previous one (bigram model) or previous two (trigram model) or previous n tags (n-gram model) which, mathematically, can be explained as follows:

PROB (C_1,..., C_T) = $\prod_{i=1..T}$ PROB (C_i|$C_{i-n+1} \ldots C_{i-1}$) (n-gram model)

PROB (C_1,..., CT) = $\prod_{i=1..T}$ PROB (C_i|C_{i-1}) (bigram model)

The beginning of a sentence can be accounted for by assuming an initial probability for each tag.

PROB (C_1|C_0) = PROB $_{\text{initial}}$ (C_1)

Second Assumption

The second probability in equation (1) above can be approximated by assuming that a word appears in a category independent of the words in the preceding or succeeding categories which can be explained mathematically as follows:

PROB (W_1,..., W_T | C_1,..., C_T) = $\prod_{i=1..T}$ PROB ($W_i|C_i$)

Now, on the basis of the above two assumptions, our goal reduces to finding a sequence C which maximizes

$\prod_{i=1...T}$ **PROB($C_i|C_{i-1}$) * PROB($W_i|C_i$)**

Now the question that arises here is has converting the problem to the above form really helped us. The answer is - yes, it has. If we have a large tagged corpus, then the two probabilities in the above formula can be calculated as -

PROB ($C_{i=VERB}|C_{i-1=NOUN}$) = (# of instances where Verb follows Noun) / (# of instances where Noun appears) (2)

PROB ($W_i|C_i$) = (# of instances where W_i appears in C_i) /(# of instances where C_i appears) (3)

9. Natural Language Processing — Natural Language Inception

In this chapter, we will discuss the natural language inception in Natural Language Processing. To begin with, let us first understand what is Natural Language Grammar.

Natural Language Grammar

For linguistics, language is a group of arbitrary vocal signs. We may say that language is creative, governed by rules, innate as well as universal at the same time. On the other hand, it is humanly too. The nature of the language is different for different people. There is a lot of misconception about the nature of the language. That is why it is very important to understand the meaning of the ambiguous term '***grammar***'. In linguistics, the term grammar may be defined as the rules or principles with the help of which language works. In broad sense, we can divide grammar in two categories:

Descriptive Grammar

The set of rules, where linguistics and grammarians formulate the speaker's grammar is called descriptive grammar.

Perspective Grammar

It is a very different sense of grammar, which attempts to maintain a standard of correctness in the language. This category has little to do with the actual working of the language.

Components of Language

The language of study is divided into the interrelated components, which are conventional as well as arbitrary divisions of linguistic investigation. The explanation of these components is as follows:

Phonology

The very first component of language is phonology. It is the study of the speech sounds of a particular language. The origin of the word can be traced to Greek language, where 'phone' means sound or voice. Phonetics, a subdivision of phonology is the study of the speech sounds of human language from the perspective of their production, perception or their physical properties. IPA (International Phonetic Alphabet) is a tool that represents human sounds in a regular way while studying phonology. In IPA, every written symbol represents one and only one speech sound and vice-versa.

Phonemes

It may be defined as one of the units of sound that differentiate one word from other in a language. In linguistic, phonemes are written between slashes. For example, phoneme **/k/** occurs in the words such as kit, skit.

Morphology

It is the second component of language. It is the study of the structure and classification of the words in a particular language. The origin of the word is from Greek language, where the word 'morphe' means 'form'. Morphology considers the principles of formation of words in a language. In other words, how sounds combine into meaningful units like prefixes, suffixes and roots. It also considers how words can be grouped into parts of speech.

Lexeme

In linguistics, the abstract unit of morphological analysis that corresponds to a set of forms taken by a single word is called lexeme. The way in which a lexeme is used in a sentence is determined by its grammatical category. Lexeme can be individual word or multiword. For example, the word talk is an example of an individual word lexeme, which may have many grammatical variants like talks, talked and talking. Multiword lexeme can be made up of more than one orthographic word. For example, speak up, pull through, etc. are the examples of multiword lexemes.

Syntax

It is the third component of language. It is the study of the order and arrangement of the words into larger units. The word can be traced to Greek language, where the word suntassein means 'to put in order'. It studies the type of sentences and their structure, of clauses, of phrases.

Semantics

It is the fourth component of language. It is the study of how meaning is conveyed. The meaning can be related to the outside world or can be related to the grammar of the sentence. The word can be traced to Greek language, where the word semainein means means 'to signify', 'show', 'signal'.

Pragmatics

It is the fifth component of language. It is the study of the functions of the language and its use in context. The origin of the word can be traced to Greek language where the word 'pragma' means 'deed', 'affair'.

Grammatical Categories

A grammatical category may be defined as a class of units or features within the grammar of a language. These units are the building blocks of language and share a common set of characteristics. Grammatical categories are also called grammatical features

The inventory of grammatical categories is described below:

Number

It is the simplest grammatical category. We have two terms related to this category – singular and plural. Singular is the concept of 'one' whereas, plural is the concept of 'more than one'. For example, dog/dogs, this/these.

Gender

Grammatical gender is expressed by variation in personal pronouns and 3rd person. Examples of grammatical genders are singular – he, she, it; the first and second person forms – I, we and you; the 3rd person plural form they, is either common gender or neuter gender.

Person

Another simple grammatical category is person. Under this, following three terms are recognized:

- **1st person:** The person who is speaking is recognized as 1st person.
- **2nd person:** The person who is the hearer or the person spoken to is recognized as 2nd person.
- **3rd person:** The person or thing about whom we are speaking is recognized as 3rd person.

Case

It is one of the most difficult grammatical categories. It may be defined as an indication of the function of a noun phrase (NP) or the relationship of a noun phrase to a verb or to the other noun phrases in the sentence. We have the following three cases expressed in personal and interrogative pronouns:

- Nominative case: It is the function of subject. For example, I, we, you, he, she, it, they and who are nominative.
- Genitive case: It is the function of possessor. For example, my/mine, our/ours, his, her/hers, its, their/theirs, whose are genitive.
- Objective case: It is the function of object. For example, me, us, you, him, her, them, whom are objective.

Degree

This grammatical category is related to adjectives and adverbs. It has the following three terms:

- **Positive degree:** It expresses a quality. For example, big, fast, beautiful are positive degrees.
- **Comparative degree:** It expresses greater degree or intensity of the quality in one of two items. For example, bigger, faster, more beautiful are comparative degrees.
- **Superlative degree:** It expresses greatest degree or intensity of the quality in one of three or more items. For example, biggest, fastest, most beautiful are superlative degrees.

Definiteness and Indefiniteness

Both these concepts are very simple. Definiteness as we know represents a referent, which is known, familiar or identifiable by the speaker or hearer. On the other hand, indefiniteness represents a referent that is not known, or is unfamiliar. The concept can be understood in the co-occurrence of an article with a noun -

- **definite article:** *the*
- **indefinite article:** *a/an*

Tense

This grammatical category is related to verb and can be defined as the linguistic indication of the time of an action. A tense establishes a relation because it indicates the time of an event with respect to the moment of speaking. Broadly, it is of the following three types:

- **Present tense:** Represents the occurrence of an action in the present moment. For example, Ram works hard.

- **Past tense:** Represents the occurrence of an action before the present moment. For example, it rained.

- **Future tense:** Represents the occurrence of an action after the present moment. For example, it will rain.

Aspect

This grammatical category may be defined as the view taken of an event. It can be of the following types:

- **Perfective aspect:** The view is taken as whole and complete in the aspect. For example, the simple past tense like **yesterday I met my friend**, in English is perfective in aspect as it views the event as complete and whole.

- **Imperfective aspect:** The view is taken as ongoing and incomplete in the aspect. For example, the present participle tense like **I am working on this problem**, in English is imperfective in aspect as it views the event as incomplete and ongoing.

Mood

This grammatical category is a bit difficult to define but it can be simply stated as the indication of the speaker's attitude towards what he/she is talking about. It is also the grammatical feature of verbs. It is distinct from grammatical tenses and grammatical aspect. The examples of moods are indicative, interrogative, imperative, injunctive, subjunctive, potential, optative, gerunds and participles.

Agreement

It is also called concord. It happens when a word changes from depending on the other words to which it relates. In other words, it involves making the value of some grammatical category agree between different words or part of speech. Followings are the agreements based on other grammatical categories:

- **Agreement based on Person:** It is the agreement between subject and the verb. For example, we always use "I am" and "He is" but never "He am" and "I is".

- **Agreement based on Number:** This agreement is between subject and the verb. In this case, there are specific verb forms for first person singular, second person plural and so on. For example, 1st person singular: I really am, 2nd person plural: We really are, 3rd person singular: The boy sings, 3rd person plural: The boys sing.

- **Agreement based on Gender:** In English, there is agreement in gender between pronouns and antecedents. For example, He reached his destination. The ship reached her destination.

- **Agreement based on Case:** This kind of agreement is not a significant feature of English. For example, who came first - he or his sister?

Spoken Language Syntax

The written English and spoken English grammar have many common features but along with that, they also differ in a number of aspects. The following features distinguish between the spoken and written English grammar:

Disfluencies and Repair

This striking feature makes spoken and written English grammar different from each other. It is individually known as phenomena of disfluencies and collectively as phenomena of repair. Disfluencies include the use of following:

- **Fillers words:** Sometimes in between the sentence, we use some filler words. They are called fillers of filler pause. Examples of such words are uh and um.

- **Reparandum and repair:** The repeated segment of words in between the sentence is called reparandum. In the same segment, the changed word is called repair. Consider the following example to understand this:

Does ABC airlines offer any one-way flights uh one-way fares for 5000 rupees?

In the above sentence, one-way flight is a reparadum and one-way flights is a repair.

Restarts

After the filler pause, restarts occurs. For example, in the above sentence, restarts occur when the speaker starts asking about one-way flights then stops, correct himself by filler pause and then restarting asking about one-way fares.

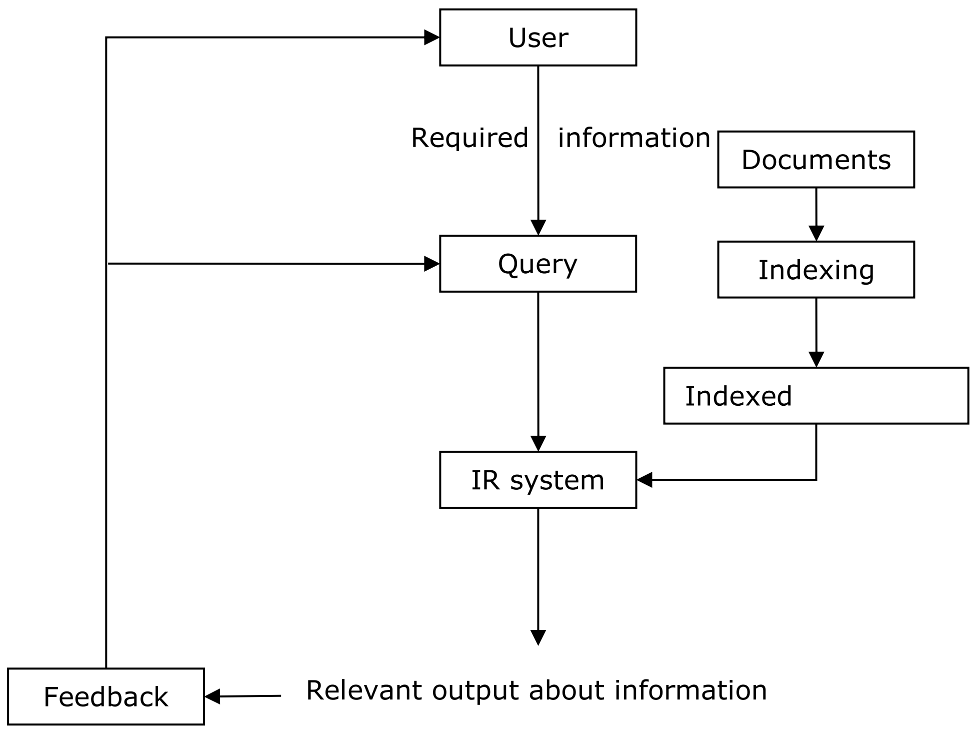
Word Fragments

Sometimes we speak the sentences with smaller fragments of words. For example, **w-wha-what is the time?** Here the words **w-wha** are word fragments.

10. Natural Language Processing — Information Retrieval

Information retrieval (IR) may be defined as a software program that deals with the organization, storage, retrieval and evaluation of information from document repositories particularly textual information. The system assists users in finding the information they require but it does not explicitly return the answers of the questions. It informs the existence and location of documents that might consist of the required information. The documents that satisfy user's requirement are called relevant documents. A perfect IR system will retrieve only relevant documents.

With the help of the following diagram, we can understand the process of information retrieval (IR):



It is clear from the above diagram that a user who needs information will have to formulate a request in the form of query in natural language. Then the IR system will respond by retrieving the relevant output, in the form of documents, about the required information.

Classical Problem in Information Retrieval (IR) System

The main goal of IR research is to develop a model for retrieving information from the repositories of documents. Here, we are going to discuss a classical problem, named **ad-hoc retrieval problem**, related to the IR system.

In ad-hoc retrieval, the user must enter a query in natural language that describes the required information. Then the IR system will return the required documents related to the desired information. For example, suppose we are searching something on the Internet and it gives some exact pages that are relevant as per our requirement but there can be some non-relevant pages too. This is due to the ad-hoc retrieval problem.

Aspects of Ad-hoc Retrieval

Followings are some aspects of ad-hoc retrieval that are addressed in IR research:

- How users with the help of relevance feedback can improve original formulation of a query?
- How to implement database merging, i.e., how results from different text databases can be merged into one result set?
- How to handle partly corrupted data? Which models are appropriate for the same?

Information Retrieval (IR) Model

Mathematically, models are used in many scientific areas having objective to understand some phenomenon in the real world. A model of information retrieval predicts and explains what a user will find in relevance to the given query. IR model is basically a pattern that defines the above-mentioned aspects of retrieval procedure and consists of the following:

- A model for documents.
- A model for queries.
- A matching function that compares queries to documents.

Mathematically, a retrieval model consists of:

D: Representation for documents.

R: Representation for queries.

F: The modeling framework for D, Q along with relationship between them.

R (q,d_i): A similarity function which orders the documents with respect to the query. It is also called ranking.

Types of Information Retrieval (IR) Model

An information model (IR) model can be classified into the following three models:

Classical IR Model

It is the simplest and easy to implement IR model. This model is based on mathematical knowledge that was easily recognized and understood as well. Boolean, Vector and Probabilistic are the three classical IR models.

Non-Classical IR Model

It is completely opposite to classical IR model. Such kind of IR models are based on principles other than similarity, probability, Boolean operations. Information logic model, situation theory model and interaction models are the examples of non-classical IR model.

Alternative IR Model

It is the enhancement of classical IR model making use of some specific techniques from some other fields. Cluster model, fuzzy model and latent semantic indexing (LSI) models are the example of alternative IR model.

Design features of Information retrieval (IR) systems

Let us now learn about the design features of IR systems:

Inverted Index

The primary data structure of most of the IR systems is in the form of inverted index. We can define an inverted index as a data structure that list, for every word, all documents that contain it and frequency of the occurrences in document. It makes it easy to search for 'hits' of a query word.

Stop Word Elimination

Stop words are those high frequency words that are deemed unlikely to be useful for searching. They have less semantic weights. All such kind of words are in a list called stop list. For example, articles "a", "an", "the" and prepositions like "in", "of", "for", "at" etc. are the examples of stop words. The size of the inverted index can be significantly reduced by stop list. As per Zipf's law, a stop list covering a few dozen words reduces the size of inverted index by almost half. On the other hand, sometimes the elimination of stop word may cause elimination of the term that is useful for searching. For example, if we eliminate the alphabet "A" from "Vitamin A" then it would have no significance.

Stemming

Stemming, the simplified form of morphological analysis, is the heuristic process of extracting the base form of words by chopping off the ends of words. For example, the words laughing, laughs, laughed would be stemmed to the root word laugh.

In our subsequent sections, we will discuss about some important and useful IR models.

The Boolean Model

It is the oldest information retrieval (IR) model. The model is based on set theory and the Boolean algebra, where documents are sets of terms and queries are Boolean expressions on terms. The Boolean model can be defined as:

- **D:** A set of words, i.e., the indexing terms present in a document. Here, each term is either present (1) or absent (0).

- **Q:** A Boolean expression, where terms are the index terms and operators are logical products – AND, logical sum – OR and logical difference – NOT.

- **F:** Boolean algebra over sets of terms as well as over sets of documents.

  If we talk about the relevance feedback, then in Boolean IR model the Relevance prediction can be defined as follows:

- **R:** A document is predicted as relevant to the query expression if and only if it satisfies the query expression as -

$$((text \vee information) \wedge rerieval \wedge \tilde{}\ theory)$$

We can explain this model by a query term as an unambiguous definition of a set of documents.

For example, the query term ***"economic"*** defines the set of documents that are indexed with the term ***"economic".***

Now, what would be the result after combining terms with Boolean AND Operator? It will define a document set that is smaller than or equal to the document sets of any of the single terms. For example, the query with terms "***social"*** and "***economic"*** will produce the documents set of documents that are indexed with both the terms. In other words, document set with the intersection of both the sets.

Now, what would be the result after combining terms with Boolean OR operator? It will define a document set that is bigger than or equal to the document sets of any of the single terms. For example, the query with terms ***"social"*** or "***economic"*** will produce the documents set of documents that are indexed with either the term ***"social"*** or "***economic"***. In other words, document set with the union of both the sets.

Advantages of the Boolean Model

The advantages of the Boolean model are as follows:

- The simplest model, which is based on sets.
- Easy to understand and implement.
- It only retrieves exact matches.
- It gives the user, a sense of control over the system.

Disadvantages of the Boolean Model

The disadvantages of the Boolean model are as follows:

- The model's similarity function is Boolean. Hence, there would be no partial matches. This can be annoying for the users.
- In this model, the Boolean operator usage has much more influence than a critical word.
- The query language is expressive, but it is complicated too.
- No ranking for retrieved documents.

Vector Space Model

Due to the above disadvantages of the Boolean model, Gerard Salton and his colleagues suggested a model, which is based on Luhn's similarity criterion. The similarity criterion formulated by Luhn states, "the more two representations agreed in given elements and their distribution, the higher would be the probability of their representing similar information."

Consider the following important points to understand more about the Vector Space Model:

- The index representations (documents) and the queries are considered as vectors embedded in a high dimensional Euclidean space.
- The similarity measure of a document vector to a query vector is usually the cosine of the angle between them.

Cosine Similarity Measure Formula

Cosine is a normalized dot product, which can be calculated with the help of the following formula:

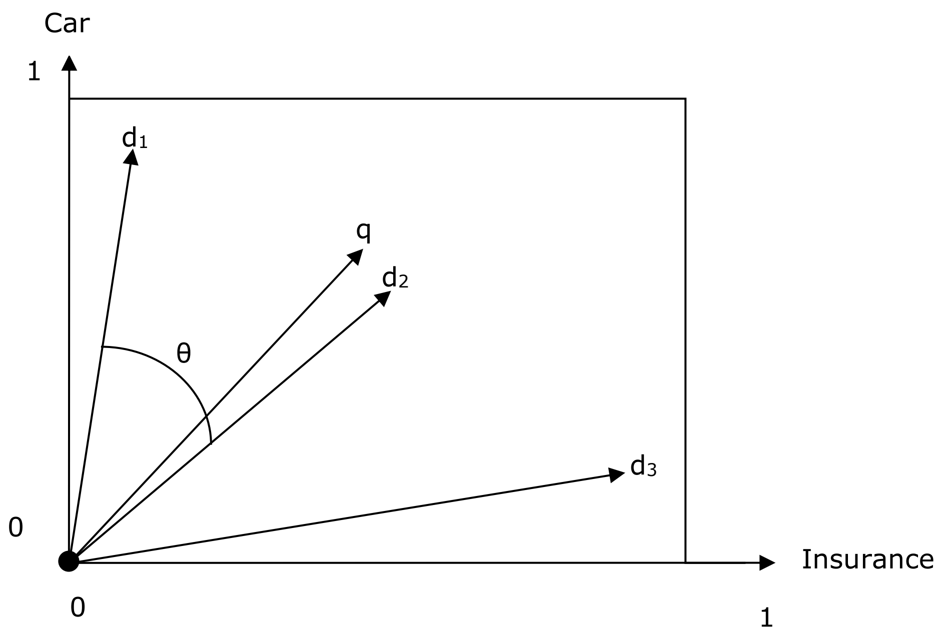
$$Score(\vec{d},\vec{q}) = \frac{\sum_{k=1}^{m} d_k \cdot q_k}{\sqrt{\sum_{k=1}^{m}(d_k)^2} \cdot \sqrt{\sum_{k=1}^{m}(q_k)^2}}$$

$$Score(\vec{d},\vec{q}) = 1 \ when \ d = q$$

$$Score(\vec{d},\vec{q}) = 0 \ when \ d \ and \ q \ share \ no \ items$$

Vector Space Representation with Query and Document

The query and documents are represented by a two-dimensional vector space. The terms are ***car*** and ***insurance***. There is one query and three documents in the vector space.



The top ranked document in response to the terms car and insurance will be the document **d₂** because the angle between **q** and **d₂** is the smallest. The reason behind this is that both the concepts car and insurance are salient in d₂ and hence have the high weights. On the other side, **d_1** and **d_3** also mention both the terms but in each case, one of them is not a centrally important term in the document.

Term Weighting

Term weighting means the weights on the terms in vector space. Higher the weight of the term, greater would be the impact of the term on cosine. More weights should be assigned to the more important terms in the model. Now the question that arises here is how can we model this.

One way to do this is to count the words in a document as its term weight. However, do you think it would be effective method?

Another method, which is more effective, is to use **term frequency (tf_{ij})**, **document frequency (df_i)** and **collection frequency (cf_i)**.

Term Frequency (tf_{ij})

It may be defined as the number of occurrences of **w_i** in **d_j.** The information that is captured by term frequency is how salient a word is within the given document or in other words we can say that the higher the term frequency the more that word is a good description of the content of that document.

Document Frequency (df_i)

It may be defined as the total number of documents in the collection in which **w_i** occurs. It is an indicator of informativeness. Semantically focused words will occur several times in the document unlike the semantically unfocused words.

Collection Frequency (cf_i)

It may be defined as the total number of occurrences of **w_i** in the collection.

Mathematically, $df_i \leq cf_i$ and $\sum_j tf_{ij} = cf_i$

Forms of Document Frequency Weighting

Let us now learn about the different forms of document frequency weighting. The forms are described below:

Term Frequency Factor

This is also classified as the term frequency factor, which means that if a term ***t*** appears often in a document then a query containing ***t*** should retrieve that document. We can combine word's **term frequency (tf_{ij})** and **document frequency (df_i)** into a single weight as follows:

$$weight(i,j) = \begin{cases} (1+\log(tf_{i,j}))log\frac{N}{df_i} & if\ tf_{i,j} \geq 1 \\ 0 & if\ tf_{i,j} = 0 \end{cases}$$

Here N is the total number of documents.

Inverse Document Frequency (idf)

This is another form of document frequency weighting and often called **idf** weighting or inverse document frequency weighting. The important point of idf weighting is that the term's scarcity across the collection is a measure of its importance and importance is inversely proportional to frequency of occurrence.

Mathematically,

$$idf_t = \log\left(1 + \frac{N}{n_t}\right)$$

$$idf_t = \log\left(\frac{N - n_t}{n_t}\right)$$

Here,

N = documents in the collection

n_t = documents containing term *t*

User Query Improvement

The primary goal of any information retrieval system must be accuracy – to produce relevant documents as per the user's requirement. However, the question that arises here is how can we improve the output by improving user's query formation style. Certainly, the output of any IR system is dependent on the user's query and a well-formatted query will produce more accurate results. The user can improve his/her query with the help of ***relevance feedback,*** an important aspect of any IR model.

Relevance Feedback

Relevance feedback takes the output that is initially returned from the given query. This initial output can be used to gather user information and to know whether that output is relevant to perform a new query or not. The feedbacks can be classified as follows:

Explicit Feedback

It may be defined as the feedback that is obtained from the assessors of relevance. These assessors will also indicate the relevance of a document retrieved from the query. In order to improve query retrieval performance, the relevance feedback information needs to be interpolated with the original query.

Assessors or other users of the system may indicate the relevance explicitly by using the following relevance systems:

- **Binary relevance system:** This relevance feedback system indicates that a document is either relevant (1) or irrelevant (0) for a given query.

- **Graded relevance system:** The graded relevance feedback system indicates the relevance of a document, for a given query, on the basis of grading by using numbers, letters or descriptions. The description can be like "not relevant", "somewhat relevant", "very relevant" or "relevant".

Implicit Feedback

It is the feedback that is inferred from user behavior. The behavior includes the duration of time user spent viewing a document, which document is selected for viewing and which is not, page browsing and scrolling actions, etc. One of the best examples of implicit feedback is ***dwell time,*** which is a measure of how much time a user spends viewing the page linked to in a search result.

Pseudo Feedback

It is also called Blind feedback. It provides a method for automatic local analysis. The manual part of relevance feedback is automated with the help of Pseudo relevance feedback so that the user gets improved retrieval performance without an extended interaction. The main advantage of this feedback system is that it does not require assessors like in explicit relevance feedback system.

Consider the following steps to implement this feedback:

Step 1: First, the result returned by initial query must be taken as relevant result. The range of relevant result must be in top 10-50 results.

Step 2: Now, select the top 20-30 terms from the documents using for instance term frequency(tf)-inverse document frequency(idf) weight.

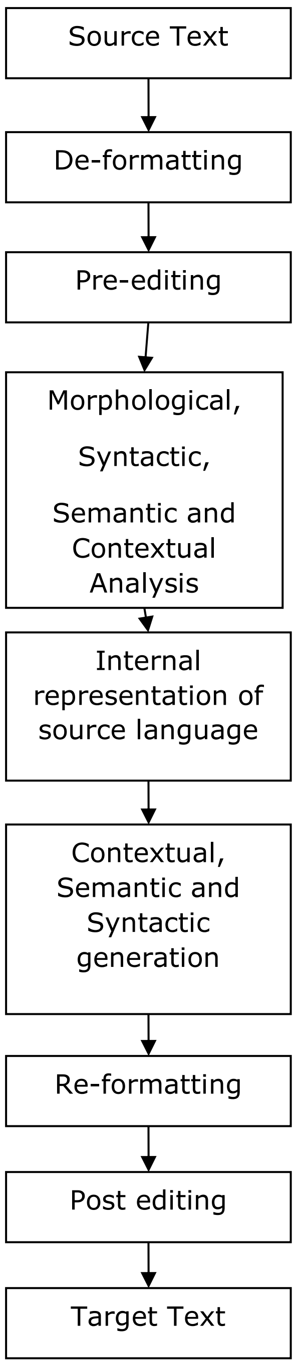
Step 3: Add these terms to the query and match the returned documents. Then return the most relevant documents.

11. Natural Language Processing — Applications of NLP

Natural Language Processing (NLP) is an emerging technology that derives various forms of AI that we see in the present times and its use for creating a seamless as well as interactive interface between humans and machines will continue to be a top priority for today's and tomorrow's increasingly cognitive applications. Here, we are going to discuss about some of the very useful applications of NLP.

Machine Translation

Machine translation (MT), process of translating one source language or text into another language, is one of the most important applications of NLP. We can understand the process of machine translation with the help of the following flowchart:


Source Text
De-formatting
Pre-editing
Morphological,
Syntactic,
Semantic and Contextual Analysis
Internal representation of source language
Contextual, Semantic and Syntactic generation
Re-formatting
Post editing
Target Text

Types of Machine Translation Systems

There are different types of machine translation systems. Let us see what the different types are.

Bilingual MT System

Bilingual MT systems produce translations between two particular languages.

Multilingual MT System

Multilingual MT systems produce translations between any pair of languages. They may be either uni-directional or bi-directional in nature.

Approaches to Machine Translation (MT)

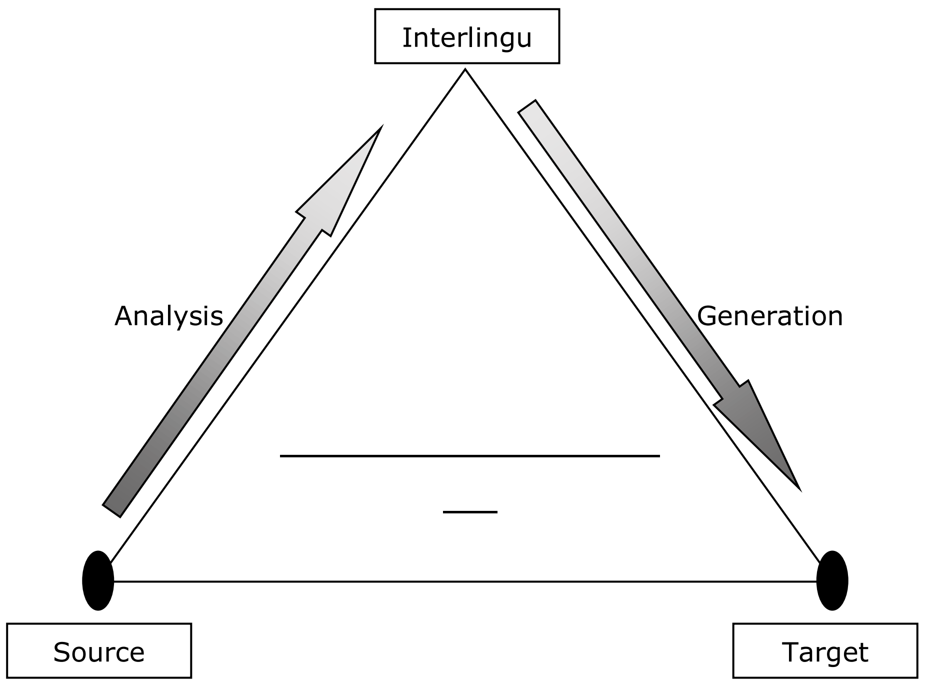
Let us now learn about the important approaches to Machine Translation. The approaches to MT are as follows:

Direct MT Approach

It is less popular but the oldest approach of MT. The systems that use this approach are capable of translating SL (source language) directly to TL (target language). Such systems are bi-lingual and uni-directional in nature.

Interlingua Approach

The systems that use Interlingua approach translate SL to an intermediate language called Interlingua (IL) and then translate IL to TL. The Interlingua approach can be understood with the help of the following MT pyramid:



Transfer Approach

Three stages are involved with this approach.

- In the first stage, source language (SL) texts are converted to abstract SL-oriented representations.
- In the second stage, SL-oriented representations are converted into equivalent target language (TL)-oriented representations.
- In the third stage, the final text is generated.

Empirical MT Approach

This is an emerging approach for MT. Basically, it uses large amount of raw data in the form of parallel corpora. The raw data consists of the text and their translations. Analogy-based, example-based, memory-based machine translation techniques use empirical MTapproach.

Fighting Spam

One of the most common problems these days is unwanted emails. This makes Spam filters all the more important because it is the first line of defense against this problem.

Spam filtering system can be developed by using NLP functionality by considering the major false-positive and false-negative issues.

Existing NLP models for spam filtering

Followings are some existing NLP models for spam filtering:

N-gram Modeling

An N-Gram model is an N-character slice of a longer string. In this model, N-grams of several different lengths are used simultaneously in processing and detecting spam emails.

Word Stemming

Spammers, generators of spam emails, usually change one or more characters of attacking words in their spams so that they can breach content-based spam filters. That is why we can say that content-based filters are not useful if they cannot understand the meaning of the words or phrases in the email. In order to eliminate such issues in spam filtering, a rule-based word stemming technique, that can match words which look alike and sound alike, is developed.

Bayesian Classification

This has now become a widely-used technology for spam filtering. The incidence of the words in an email is measured against its typical occurrence in a database of unsolicited (spam) and legitimate (ham) email messages in a statistical technique.

Automatic Summarization

In this digital era, the most valuable thing is data, or you can say information. However, do we really get useful as well as the required amount of information? The answer is 'NO' because the information is overloaded and our access to knowledge and information far exceeds our capacity to understand it. We are in a serious need of automatic text summarization and information because the flood of information over internet is not going to stop.

Text summarization may be defined as the technique to create short, accurate summary of longer text documents. Automatic text summarization will help us with relevant information in less time. Natural language processing (NLP) plays an important role in developing an automatic text summarization.

Question-answering

Another main application of natural language processing (NLP) is question-answering. Search engines put the information of the world at our fingertips, but they are still lacking when it comes to answer the questions posted by human beings in their natural language. We have big tech companies like Google are also working in this direction.

Question-answering is a Computer Science discipline within the fields of AI and NLP. It focuses on building systems that automatically answer questions posted by human beings in their natural language. A computer system that understands the natural language has the capability of a program system to translate the sentences written by humans into an internal representation so that the valid answers can be generated by the system. The exact answers can be generated by doing syntax and semantic analysis of the questions. Lexical gap, ambiguity and multilingualism are some of the challenges for NLP in building good question answering system.

Sentiment Analysis

Another important application of natural language processing (NLP) is sentiment analysis. As the name suggests, sentiment analysis is used to identify the sentiments among several posts. It is also used to identify the sentiment where the emotions are not expressed explicitly. Companies are using sentiment analysis, an application of natural language processing (NLP) to identify the opinion and sentiment of their customers online. It will help companies to understand what their customers think about the products and services. Companies can judge their overall reputation from customer posts with the help of sentiment analysis. In this way, we can say that beyond determining simple polarity, sentiment analysis understands sentiments in context to help us better understand what is behind the expressed opinion.

12. Natural Language Processing — Language Processing and Python

In this chapter, we will learn about language processing using Python.

The following features make Python different from other languages:

- **Python is interpreted:** We do not need to compile our Python program before executing it because the interpreter processes Python at runtime.

- **Interactive:** We can directly interact with the interpreter to write our Python programs.

- **Object-oriented:** Python is object-oriented in nature and it makes this language easier to write programs because with the help of this technique of programming it encapsulates code within objects.

- **Beginner can easily learn:** Python is also called beginner's language because it is very easy to understand, and it supports the development of a wide range of applications.

Prerequisites

The latest version of Python 3 released is Python 3.7.1 is available for Windows, Mac OS and most of the flavors of Linux OS.

- For windows, we can go to the link https://www.python.org/downloads/windows/ to download and install Python.

- For MAC OS, we can use the link https://www.python.org/downloads/mac-osx/.

- In case of Linux, different flavors of Linux use different package managers for installation of new packages.
  - For example, to install Python 3 on Ubuntu Linux, we can use the following command from terminal:

```
$sudo apt-get install python3-minimal
```

To study more about Python programming, read Python 3 basic tutorial – https://www.tutorialspoint.com/python3/index.htm.

Getting Started with NLTK

We will be using Python library NLTK (Natural Language Toolkit) for doing text analysis in English Language. The Natural language toolkit (NLTK) is a collection of Python libraries designed especially for identifying and tag parts of speech found in the text of natural language like English.

Installing NLTK

Before starting to use NLTK, we need to install it. With the help of following command, we can install it in our Python environment:

```
pip install nltk
```

If we are using Anaconda, then a Conda package for NLTK can be built by using the following command:

```
conda install -c anaconda nltk
```

Downloading NLTK's Data

After installing NLTK, another important task is to download its preset text repositories so that it can be easily used. However, before that we need to import NLTK the way we import any other Python module. The following command will help us in importing NLTK:

```
import nltk
```

Now, download NLTK data with the help of the following command:

```
nltk.download()
```

It will take some time to install all available packages of NLTK.

Other Necessary Packages

Some other Python packages like **gensim** and **pattern** are also very necessary for text analysis as well as building natural language processing applications by using NLTK. the packages can be installed as shown below:

gensim

gensim is a robust semantic modeling library which can be used for many applications. We can install it by following command:

```
pip install gensim
```

pattern

It can be used to make **gensim** package work properly. The following command helps iin installing pattern:

```
pip install pattern
```

Tokenization

Tokenization may be defined as the Process of breaking the given text, into smaller units called tokens. Words, numbers or punctuation marks can be tokens. It may also be called word segmentation.

Example

Input: Bed and chair are types of furniture.

Output:

	Bed	and	chair	are	types	of	furniture

We have different packages for tokenization provided by NLTK. We can use these packages based on our requirements. The packages and the details of their installation are as follows:

sent_tokenize package

This package can be used to divide the input text into sentences. We can import it by using the following command:

```
from nltk.tokenize import sent_tokenize
```

word_tokenize package

his package can be used to divide the input text into words. We can import it by using the following command:

```
from nltk.tokenize import word_tokenize
```

WordPunctTokenizer package

This package can be used to divide the input text into words and punctuation marks. We can import it by using the following command:

```
from nltk.tokenize import WordPuncttokenizer
```

Stemming

Due to grammatical reasons, language includes lots of variations. Variations in the sense that the language, English as well as other languages too, have different forms of a word. For example, the words like ***democracy***, ***democratic***, and ***democratization***. For machine learning projects, it is very important for machines to understand that these different words, like above, have the same base form. That is why it is very useful to extract the base forms of the words while analyzing the text.

Stemming is a heuristic process that helps in extracting the base forms of the words by chopping of their ends.

The different packages for stemming provided by NLTK module are as follows:

PorterStemmer package

Porter's algorithm is used by this stemming package to extract the base form of the words. With the help of the following command, we can import this package:

```
from nltk.stem.porter import PorterStemmer
```

For example, **'write'** would be the output of the word **'writing'** given as the input to this stemmer.

LancasterStemmer package

Lancaster's algorithm is used by this stemming package to extract the base form of the words. With the help of following command, we can import this package:

```
from nltk.stem.lancaster import LancasterStemmer
```

For example, **'writ'** would be the output of the word **'writing'** given as the input to this stemmer.

SnowballStemmer package

Snowball's algorithm is used by this stemming package to extract the base form of the words. With the help of following command, we can import this package:

```
from nltk.stem.snowball import SnowballStemmer
```

For example, **'write'** would be the output of the word **'writing'** given as the input to this stemmer.

Lemmatization

It is another way to extract the base form of words, normally aiming to remove inflectional endings by using vocabulary and morphological analysis. After lemmatization, the base form of any word is called lemma.

NLTK module provides the following package for lemmatization:

WordNetLemmatizer package

This package will extract the base form of the word depending upon whether it is used as a noun or as a verb. The following command can be used to import this package:

```
from nltk.stem import WordNetLemmatizer
```

Counting POS Tags – Chunking

The identification of parts of speech (POS) and short phrases can be done with the help of chunking. It is one of the important processes in natural language processing. As we are aware about the process of tokenization for the creation of tokens, chunking actually is to do the labeling of those tokens. In other words, we can say that we can get the structure of the sentence with the help of chunking process.

Example

In the following example, we will implement Noun-Phrase chunking, a category of chunking which will find the noun phrase chunks in the sentence, by using NLTK Python module.

Consider the following steps to implement noun-phrase chunking:

Step 1: Chunk grammar definition

In this step, we need to define the grammar for chunking. It would consist of the rules, which we need to follow.

Step 2: **Chunk parser creation**

Next, we need to create a chunk parser. It would parse the grammar and give the output.

Step 3: The Output

In this step, we will get the output in a tree format.

Running the NLP Script

Start by importing the the NLTK package:

```
import nltk
```

Now, we need to define the sentence.

Here,

- DT is the determinant
- VBP is the verb
- JJ is the adjective
- IN is the preposition
- NN is the noun

```
sentence = [("a", "DT"),("clever",
"JJ"),("fox","NN"),("was","VBP"),("jumping","VBP"),("over","IN"),("the","DT"),(
"wall","NN")]
```

Next, the grammar should be given in the form of regular expression.

```
grammar = "NP:{<DT>?<JJ>*<NN>}"
```

Now, we need to define a parser for parsing the grammar.

```
parser_chunking=nltk.RegexpParser(grammar)
```

Now, the parser will parse the sentence as follows:

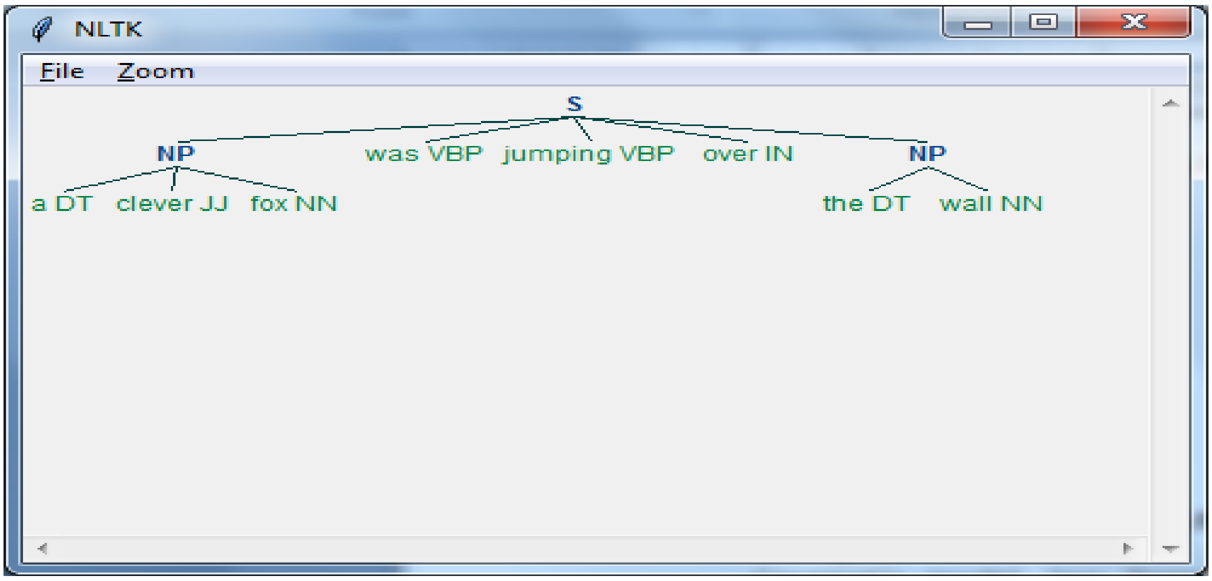
```
parser_chunking.parse(sentence)
```

Next, the output will be in the variable as follows:-

```
Output=parser_chunking.parse(sentence)
```

Now, the following code will help you draw your output in the form of a tree.

```
output.draw()
```


NLTK
File
Zoom
S
NP
was VBP
jumping VBP
over IN
NP
a DT
clever JJ
fox NN
the DT
wall NN