UNIT I

CHAPTER 1

Introduction to Natural Language Processing

Syllabus

Introduction : Natural Language Processing, Why NLP is hard? Programming languages Vs Natural Languages, Are natural languages regular? Finite automata for NLP, Stages of NLP, Challenges and Issues(Open Problems) in NLP.

Basics of text processing : Tokenization, Stemming, Lemmatization, Part of Speech Tagging

1.1 ORIGIN AND HISTORY OF NLP

Natural language processing (NLP) is part of everday life and it is essential to our lives at home and at work. We can send voice commands to our home assistants, our smartphones, etc.

Voice enabled applications such as alexa, siri, and google assistant use NLP to answer our questions. It can add activities to our calendars and also call the contacts that we mention in our voice commands.

NLP has made our lives easier. But more than that it has revolutionised the way we work, live and play.

- Communication is an important act that agent can perform so as to exchange information with the environment. Communication can be carried out by producing and Percepting certain signs drawn from a shared system of conventional signs.
- In a partially observable world, communication can help agents to learn information that is observed or inferred by others. This information can make agent more successful.
- Language is meant for communicating about the world. By studying language, we can come to understand more about the world. We can test our theories about the world by how well they support our attempt to understand language. And, if we can succeed at building a computational model of language, we will have a powerful tool for communicating about the world. In this chapter, we look at how we can exploit knowledge about the world, in combination with linguistic facts, to build computational natural language systems.

- Natural Language Processing (NLP) refers to AI method of communicating with an intelligent systems using a natural language such as English. Processing of Natural Language is required when you want an intelligent system like robot to perform as per your instructions, when you want to hear decision from a dialogue based clinical expert system, etc.
- The field of NLP involves making computers to perform useful tasks with the natural languages humans use. The input and output of an NLP system can be : Speech, Written text.
- Natural language understanding is a subtopic of natural language processing in artificial intelligence that deals with machine reading comprehension.
- The goal of the Natural Language Processing (NLP) group is to design and build software that will analyze, understand, and generate languages that humans use naturally, so that eventually you will be able to address your computer as though you were addressing another person.

1.2 OVERVIEW OF NLP TASK

GQ. Give general approaches to natural language process. Or Write short note on NLP.

Natural language processing (NLP) is the ability of a computer program to understand human speech as it is spoken. NLP is a component of artificial intelligence (AI).

- The development of NLP applications is challenging because computers traditionally require humans to "speak" to them in a programming language that is precise, unambiguous and highly structured or, perhaps through a limited number of clearly-enunciated voice commands. Human speech, however, is not always precise - it is often ambiguous and the linguistic structure can depend on many complex variables, including slang, regional dialects and social context.
- Current approaches to NLP are based on machine learning, a type of artificial intelligence that examines and uses patterns in data to improve a program's own understanding. Most of the research being done on natural language processing revolves around search, especially enterprise search.

Common NLP tasks in software programs today include :

(1) Sentence segmentation, part-of-speech tagging and parsing.
(2) Deep analytics.
(3) Named entity extraction.
(4) Co-reference resolution.

The advantage of natural language processing can be seen when considering the following two statements :

"Cloud computing insurance should be part of every service level agreement" and

"A good SLA ensures an easier night's sleep -- even in the cloud."

If you use national language processing for search, the program will recognize that cloud computing is an entity, that cloud is an abbreviated form of cloud computing and that SLA is an industry acronym for service level agreement.

The ultimate goal of NLP is to do away with computer programming languages altogether. Instead of specialized languages such as Java or Ruby or C, there would only be "human."

1.3 EVOLUTION OF NLP SYSTEMS

GQ. Discuss the evolution of NLP systems Or Given a brief history of NLP.

- **History of NLP :** The work related to NLP was started with machine translation (MT) in 1950s. It was Allen Turing who proposed what today is called the Turing test in 1950s. It is the testing ability of the machine program to have written conversation with human.
- This program should be written so well so that one would find it difficult to determine whether the conversation is with a machine or it is with the other person actually. During the same period of cryptography and language translation took place. Later on, syntactic structures came up along with linguistics. Further, the sentences were considered with knowledge augmentation and semantics. In 1960s, KI.IZA (the most common NLP system) was developed that gained popularity.
- It was the simulation of a psychotherapist. At a very later stage, it was the case grammars that came up. Now, there has been a complete revolution in the NLP with the machine learning approaches coming up. Many NLP systems have been developed till today and a lot of competitions are being organized that are based on the Turing test.

GQ. What is pragmatic analysis in natural language processing?

Pragmatic has not been the central concern of most NLP system. Only after ambiguities arise at syntactic or semantic level are the context and purpose of the utterance considered for analysis. Considered a problem in which pragmatic has been used in this kind of "support" capacity ambiguous noun phrases.

1.3.1 Components of NLP

There are two components of NLP : Mapping the given input in the natural language into a useful representation. Different level of analysis required: morphological analysis, syntactic analysis, semantic analysis, discourse analysis.

- **Natural language generation :** Producing output in the natural language from some internal representation. Different level of synthesis required: deep planning (what to say), syntactic generation
- **NL understanding :** NL Understanding is much harder than NL Generation. But, still both of them are hard.

- **Planning :** Planning problems are hard problems. They are certainly nontrivial. Method which we focus on ways of decomposing the original problem into appropriate subparts and on ways of handling interactions among the subparts during the problem- solving process are often called as planning. Planning refers to the process of computing several steps of a problem-solving procedure before executing any of them.

1.3.2 Major Methods of NLP Analysis

There are several main techniques used in analyzing natural language processing. Some of them can be briefly described as follows :

1. **Pattern matching :** The idea here is an approach to natural language processing is to interpret input utterances as a whole further than building up their interpretation by combining the structure and meaning of words or other lower level constituents. That means the interpretations are obtained by matching patterns of words against the input utterance. For a deep level of analysis in pattern matching a large number of patterns are required even for a restricted domain. This problem can be ameliorated by hierarchical pattern matching in which the input is gradually canonical through pattern matching against sub phrases. Another way to reduce the number of patterns is by matching with semantic primitives instead of words.
2. **Syntactically driven parsing :** Syntax means ways that words can fit together to form higher level units such as phrases, clauses and sentences. Therefore syntactically driven parsing means interpretation of larger groups of words are built up out of the interpretation of their syntactic constituent words or phrases. In a way this is the opposite of pattern matching as here the interpretation of the input is done as a whole. Syntactic analysis are obtained by application of a grammar that determines what
3. **Semantic grammars :** Natural language analysis based on semantic grammar is bit similar to syntactically driven parsing except that in semantic grammar the categories used are defined semantically and syntactically. There here semantic grammar is also involved.
4. **Case frame instantiation :** Case frame instantiation is one of the major parsing techniques under active research today. The has some very useful computational properties such as its recursive nature and its ability to combine bottom-up recognition of key constituents with top-down instantiation of less structured constituents.

1.4 NLP IS HARD

- Natural language processing is considered a difficult problem in computer science. It is the nature of the human language that makes NLP hard.
- The rules that dictate the passing of information using natural languages are not easy for computers to understand.

- There are several factors that make this process hard. For example, there are hundreds of natural languages, each of which has different syntax rules, words can be ambiguous where their meaning is dependent on their context.
- Natural languages, such as English or Spanish, cannot be characterized as definite set of sentences. Everyone agrees that "Not to be invited is sad" is a sentence of English, but people disagree on the grammatically of " To be not invited is sad".
- Therefore it is more fruitful to define a natural language model as a probability distribution over sentences rather than a definitive set.
- Hence, instead of asking it a string of **words** is or is not a member of the set defining the language, we better ask for P (s = words) – i.e. what is the probability that a random sentence would be **words.**
- Natural languages are ambiguous, "He saw her duck" can mean either that he saw a waterfowl belonging to her, or that he saw that she is evading something. This implies that we cannot speak of a single meaning for a sentence, but rather a probability distribution over possible meaning.
- English language is phonetically also not sound. Once Bernard show asked the audience, "What is the meaning of the word Ghoti ?" . The anwer was, "There is no such word in English". Show said, " yes, it is, Ghoti means fish." The Whole audience was shocked. But then he explained. "Enough is the word in English. But we pronounce 'gh' as 'F' woman is a word in English but we pronounce 'i' as 'i'. Nation is a word in English, but we pronounce 'ti' as sh.

  Therefore **Ghoti** means **Fish."**
- Words can be ambinguous where their meaning is dependent on their content.
- Here, we study a few of the more significant problem area :
- At the character level, there are several factors that need to be considered. For example, the encoding scheme used for a document is to be considered.
- Text can be encoded using schemes such as UTF – 16 or Latin – 1.Other factor to be considered as : Whether the text should be treated as case-sensitive or not. Special processing is required for punctuations and numbers.
- Sometimes we have to consider the use of emoticons (character combinations and special character images), hyperlinks, repeated punctuations, file extension, and usernames with embedded periods.
- When we tokenize text, it means that we are breaking up the text into a sequence of words. These words are called tokens. The process is called as Tokenization.
- With a language like Chinese, it can be quite difficult since it uses unique symbols for words. Words and morphemes are assigned a part of speech label identifying what type of unit it is.
- A morpheme is the smallest division text that has meaning. Prefixes and suffixes are examples of morphemes. We also consider synonyms. acronyms, abbreviations and spellings when we work with words.

- We apply another task, called as stemming stemming, is the process of finding the word stem of a word. For example, words such as "running", "runs" or "run" have the word stem "run".
- Lemmatization is a more refined process than stemming and uses vocabulary and morphological techniques to find a **lemma**. This process determines the base form of a word called its **lemma.**
- For example, for the word "operating", its stem is "oper" but its lemma is "operate". Lemmatization results in more precise analysis is some situations.
- Words are combined into phrases and sentences. Sentence detection can be problematic and is not a simples as looking for the periods at the end of a sentence.
- We need to understand which words in a sentence are nouns and which are verbs.
- We are concerned with the relationship between words.
- For example, **conferences resolutions** determines the relationship between certain words in one or more sentences.

Consider the sentence

"The city is large but beautiful, it fills the entire valley".

The word "it" is the conference to city. When a word has multiple meanings we perform **'word sense Disambiguation'** to determine the actual meaning.

Sometimes this is difficult to do. For example, "Arjun went back home."

Does the home refer to a house, a city, or some other unit? Its meaning can be inferred from the context in which it is used. For example, "Arjun went back home. It was situated at the end of Rasta Peth".

1.4.1 Performance of NLP

- In spite of these difficulties, NLP performs these tasks reasonably well in most situations and also adds value to many problems – domains.
- For example, sentiment analysis can be performed on customer-tweets resulting in possible free product offers for dissatisfied customers.
- Medical documents can be readily summered to highlight the relevant topics. Summarisation is the process of producing a short description of different units. These units include multiple sentences, paragraphs, a document, or multiple documents. The content of the text is important in accomplishing this task.
- Finally, natural languages are difficult to deal with because they are very large, and constantly changing.
- Thus, our language models are an approximation. We begin with the simplest possible approximations and more up from there.

1.5 PROGRAMMING LANGUAGE VS NATURAL LANGUAGE

- Natural language is the language spoken by the people while programming language is intended for machines.
- There are important similarities between both languages, such as the differentiation they make between syntax and semantics, their purpose to communicate and the existence of a basic composition.
- Both the types were created to communicate ideas, expressions and instructions.

1.5.1 Difference between Programming Language and Natural Language

Sr. No.	Programming language	Natural language
(1)	Programming language is stricter and less tolerant. Programming languages have practically no redundancy, otherwise it would be very easy to fall into ambiguity and may not indicate correct command.	Natural language is not that strict and is somewhat tolerant. It is because human languages have a built- in redundancy that allows some ambiguity to be resolved using context.
(2)	Programming Languages are stricter because computers are very precise in addition, machines do not have the ability to clarify the meaning of an expression as a human being would.	In computing, natural language refers to a human language such as English, Russian, German or Japanses as distinct from the typically artificial command

1.5.2 Main Features for Programming Languages

- The popularity of a programming language depends on the features and utilities it provides to programmers.
- We mention below the features that a programming language must possess.

(i) **Simplicity :** The language must offer clear and simple concepts that facilitate learning and application, in a way that is simple to understand and maintain. Simplicity is a difficult balance to strike without compromise the overall capability,

(ii) **Naturalness :** It implies that its application in the area for which it is designed must be done naturally, providing operators, structures and syntax for operators to work efficiently.

(iii) **Abstraction :** It is the ability to define and to use complicated structures or operations while ignoring certain low level details.

(iv) **Efficiency :** Programming languages must be translated and executed efficiently so as not to consume too much memory or require too much time.

(v) **Structuring :** In order to avoid creating errors, the language allows programmers to write the code according to the structured programming concepts.

(vi) **Compacters :** Using this characteristics, it is possible to express operations concisely, without having to write too many details.

(vii) **Locality :** It refers to the code concentrating on the part of the program with which one is working at a given time.

In programming language one usually talks to a computer. The terms usually refers to a written language but might also apply to spoken language.

(viii) **Natural language not to be used for programming.**

Natural language programming is not to be mixed up with natural language interfacing. In NLP the functionality of a program is organised only for the definition of the meaning of sentences.

(ix) **Expertise :** Programming languages need a high degree of expertise, completeness and precision because computer cannot think outside the statement while in speaking some minor errors are neglected.

(x) **Replacement :** Programming languages cannot be replaced by natural languages. Programming languages are more like natural languages only in the sense of "words we have in English." A key feature of programming language is that, when one writes a program and executes it, it has a well-defined meaning, which is its behaviour.

1.5.3 Ontology-assisted NLP

- NLP is an ontology-assisted way of programming in terms of natural language sentences, e.g. English.
- A structured document with content, sections and subsections for explanations of sentences forms a NLP document, which is really a computer program.
- Natural language programming is first written and then communicated with through natural language using an interface added on.
- In NLP the functionality of a program is organized only for the definition of the meaning of sentences.
- For example, NLP can be used to represent all the knowledge of an autonomous robot. After that, its tasks can be scripted by its users so that the robot can execute them autonomously while keeping to prescribed rules of behavior as determined by the robot's users. Such robots are called as **transparent robots**. It is because their reasoning is transparent to users and this develops trust in robots.

- Some methods for program synthesis are based on natural language programming.

1.5.4 Interpretation

The smallest unit of statement in NLP is a **sentence.**

Each sentence is stated in terms of concepts form the underlying ontology, attributes in that ontology are named objects in **Capital Letters.**

- In an NLP text every sentence unambiguously compiles into a **procedure call** in the underlying high-level programming language such as MATLAB, Octave, SciLab, Pythan etc.

1.5.5 Software Paradigm

Natural-language programming is a top-down method of writing software.

We mention below the stages.

(i) **Definition of Ontology :** Taxonomy – of concepts needed to describe tasks in the topic addressed.
- Each concept and all their attributes are defined in natural language words.
- This ontology will define the data structures which the NLP can use in sentences.

(ii) Definition of one or more top-level sentences in terms of concepts from the ontology. These sentences are later used to invoke the most important activities in the topic.

(iii) Defining of each of the **top-level sentences** in terms of a sequence of sentences.

(iv) Defining each of the lower-level sentences in terms of other sentences or by a simple sentence of the form **Execute code** "....." where, ... stands for a code in terms of the associated high-level programming language.

(v) Repeating the previous step till no sentence is left undefined. During this process each of the sentences can be classified to belong to a section of the document to be produced in HTML or Latex format to form the final-language program.

(vi) Using testing objects, to test the meaning of each sentence by executing its code.

(vii) Providing a library of procedure calls (in the underlying high-level language) which are needed in the code definition of some low-levle sentence meanings.

(viii) Providing a title, author data and compiling the sentences into an HTML or LateX file.

(ix) Publishing the natural language program as a webpage on the internet or as a PDF file compiled from the LaTeX document.

1.5.6 Publication Valve of Natural – Language Programs and Documents

- A natural-language program is a precise description of some procedure, that is created by its author.

- It is human readable and it can also be read by a suitable software agent. For example, a web page in an NLP format can be read by a software personal assistant to a person and he or she can ask the agent to execute some sentences, i.e., carry out some task or answer a question.
- There is a reader agent available for English interpretation of HTML based NLP documents that a person can run on the personal computer.

1.5.7 Contribution to Machine Knowledge

- An ontology class is a natural-language program is not a concept in the sense that we use it in our regular course.
- **Concepts** in an NLP are examples or samples of **generic** human concepts.
- Each sentence in a natural language program is either :
  (i) Stating a relationship is a world model or
  (ii) carries out an action in the environment, or
  (iii) carries out a computational procedure or
  (iv) invokes an answering mechanism in response to a question.
- A set of NLP sentences, with their associated ontology, can also be used as a **pseudo-code** that does not provide the details in any underlying high level programming language.
- In such an application the sentences used become high level abstractions of computing procedures that are computer language and machine independent.

1.5.8 The Theory

- Consider, for examples, a father saying to his baby son.
  "Want to suck on this bottle, dear bay?"
  and the kid says...
- "blah, suck, blah, blah, BOTTLE, blah, blah" .....but he properly responds because he has got a 'picture' of a bottle in the right side of his head connected to the word "bottle" on the left-side, and an existing "skill" near the back of his neck connected to the term "suck".
- In other words, the kid matches when he can with the pictures (types) and skills (routines), he has accumulated and he simply disregards the rest.
- Out compiles does very much the same thing, with new pictures (types) and skills (routines) being defined (–) not by us but – by the programmer, as he writes new applications code.

1.5.9 Web Programming Language

- Web development can be done through different programming languages that allows to build a site or design an applicaton.

The following options are :

(i) **Java :** multipurpose language that adjusts efficiently to web development.

(ii) **Go :** This is a flexible language that facilitates the creation of applications.

(iii) **Ruby on Rails :** It allows to design web applications quickly.

(iv) **Python :** It works on a wide variety of contents and on the web has technical advantages.

(v) **Java script :** It is an the client's side and can be extended to the server for different functions.

1.6 ARE NATURAL LANGUAGE REGULAR ?

- In theoretical computer science and formal language theory, a regular language is also called as **Rational language (RL).**
- It is defined by a **regular expression** in the strict sense in theoretical computer science.
- A regular expression (RE) is a language for specifying **text search strings.**
- RE helps us to match other strings or sets of strings. It uses a specialised syntax held in a pattern.
- Regular expressions are used to search in identical way.

1.6.1 Properties of Regular Expressions

We mention below some of the important properties of RE.

(i) RE is an algebraic notation for characterising a set of strings.
It is a formula in a special language and it can be used for specifying simple classes of strings, a sequence of symbols.

(ii) Regular expression requires two things, one is the pattern that is to be searched and the other is a corpus of text from which we need to search.

1.6.2 Mathematical Definition of RE

Mathematically, A regular expression can be defined as :

(i) 'ϵ' is a regular expression, which indicates that the language is having an empty string.

(ii) ϕ is a Regular Expression which denotes that it is an empty language.

(iii) If X and Y are Regular expressions, then

1. X, Y
2. X. Y. (or X $\cap$ Y – concatenation of XY)
3. X + Y (or X $\cup$ Y – union of X and Y)
4. X*, Y* (Closure of X and Y; also called as Kleen closure)

are also regular expressions.

(iv) If a string is derived from above rules then that is also a regular expression.

1.6.3 Examples of Regular Expressions

We mention below a few examples of Regular expressions in the following table.

Regular expressions	Regular set
(0 + 10*)	{0, 1, 10, 100, 1000, 10000, ...}
(0* 10*)	{1, 01, 10, 010, 0010, ...}
(0 + ∈) (1 + ∈)	{∈, 0, 1, 0, 1}
(a + b)*	If would be set of strings of a's and b's of any length which also includes the null string i.e. {∈, 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. {∈, 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, aaaabbbb, ...)
(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, ...)

1.6.4 Properties of Regular Sets

(i) If we do the union of two regular sets, then the resulting set would also be regular.

(ii) It we do the intersection of two regular sets, then the resulting set would also be regular.

(iii) If we carry out the complement of regular sets, then the resulting set would also be regular.

(iv) If we carry the difference of two regular sets, then the resulting set would also be regular.

(v) If we carry out the reversal of regular sets, then the resulting set would also be regular.

(vi) If we take the closure of regular sets, then the resulting set would also be regular.

(vii) If we carry out the concatenation of two regular sets, then the resulting set would also be regular.

(viii) A simple example of a language that is not regular is the set of strings $\{a^n b^n \mid n \geq 0\}$. It cannot be recognised with a finite automation, since a finite automation has finite memory and it cannot remember the enact number of a's.

(ix) It is the language accepted by a **Deterministic Finite Automation (DFA).**

(x) It is the language accepted by a nondeterministic finite automation (NFA).

(xi) It can be generated by a regular grammar.

(xii) It can be generated by a **prefix grammar.**

(xiii) It can be accepted by a read-only.

1.6.5 The Number of Words in a Regular Language

Let $S_L(n)$ denote the number of words of length n in L.

The **ordinary generating function** for L is the **formal power series :**

$$S_L(z) = \sum_{n \geq 0} S_L(n)\, z^n.$$

The generating function of a language L is a rational function if **L is regular.**

Thus for every regular language L, the sequence $S_L(n)_{n \geq 0}$ is constant – recursive.

This implies that there exists an integer constant no, complex constants λ_1, ...λ_K and complex polynomials $p_1(x)$, $p_2(x)$, ..., $p_k(x)$. Such that for every $n \geq n_o$, the number $S_L(n)$ of words of length n in L is.

$$S_L(n) = p_1(n)\, \lambda_1^n + p_2(n)\, \lambda_2^n + ... + p_K(n)\, \lambda_K^n.$$

Thus, non-regularity of certain languages L' can be proved by counting the words of a given length in L'.

Consider, for example, the Dyck language of strings of balanced parentheses. The number of words of length 2n in the Dyck language is equal to $C_n \simeq \frac{4^n}{n^{3/2}\sqrt{\pi}}$, and this is not of the form $p(n)\, \lambda^n$. Hence the Dyck language is not regular.

1.6.6 English is Not a Regular Language

- The English language is **regular** if one considers if as a set of single words. But English is more than a set of words in a dictionary.
- English grammar is the non-regular part. Given a paragraph, there is no DFA which will decide whether it is a well-written paragraph in the English language.
- Of course, it can say whether each word is an English word or not, but it cannot judge whole paragraphs.
- In particular, the standard example is that one can build sentences of the form "the mouse escaped" " The mouse the cat chased escaped. "The mouse the cat the man owned chase escaped" that are grammatical and are arbitrarily long, but are **irregular.**

1.7 FINITE AUTOMATION OF NLP

The term automata means 'self acting.' It is the plural of automation.

Automation is defined as a self-propelled computing device, which follows a predetermined sequence of operations automatically.

An **automation** having a finite number of states is called a **Finite Automation (FA) or Finite State Automation (FSA).**

Mathematically, an automation can be represented by a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

(i) Q is a finite set of states

(ii) Σ is a finite set of symbols, called the alphabet of the automation,

(iii) δ is the transition function,

(iv) q_0 is the initial state from where any input is processed (i.e. $q_0 \in Q$).

(v) F is a set of final state/states of Q ($F \subseteq Q$).

1.7.1 Relation between Finite Automata, Regular Grammars and Regular Expressions

We mention below the points which will give us a clear idea about the relationship between finite automata, regular grammars and regular expressions.

(i) Finite state automata are the theoretical foundation of computational work and regular expression is one way of describing them.

(ii) Any regular expression can be implemented as FSA and any FSA can be described with a regular expression.

(iii) Since regular expression is a way to characterise a kind of language called regular language so we can say that regular language can be described with the help of both FSA and regular expression.

(iv) Regular grammar, a formal grammar that can be right-regular or left-regular, is a way to characterise regular language.

We mention below the diagram to show that finite automata, regular expressions and regular grammars are the equivalent ways of describing regular languages.

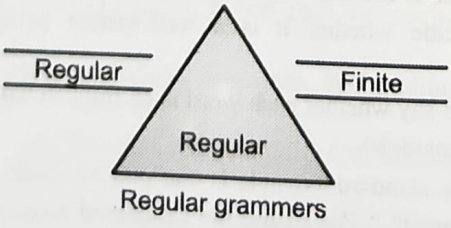


Fig. 1.7.1 : Regular Grammars

1.7.2 Types of Finit

There are two types of finite sta

1.7.2.1(A) Determi

It is defined as the type of fini
can determine the state to which the
Hence the machine is called De

Mathematically, a DFA can
where :

(i) Q is a finite set of states

(ii) Σ is a finite set of symbols, ca

(iii) δ is the transition function, wh

(iv) q_0 is the initial state from whe

(v) F is a set of final state/states o

Graphically, a DFA can be r
where :

(i) The states are represented by

(ii) The transitions are shown by

(iii) The initial state is represented

(iv) The final state is represented

1.7.2.2 Example o

Suppose a DFA be

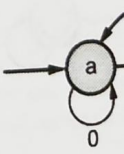
Q = {l, m, n}

q_0 = {l}

Transition function δ is as sh

Current state	Next s
L	
M	
N	

The graphical representation



1.7.2 Types of Finite State Automation (FSA)

There are two types of finite state automation :

1.7.2.1(A) Deterministic finite automation (DFA)

It is defined as the type of finite automation where, for every input symbol we can determine the state to which the machine will move. It has finite number of states.

Hence the machine is called Deterministic Finite Automation (DFA).

Mathematically, a DFA can be represented by a 5-typle $(Q, \sum, \delta, q_0, F)$, where :

(i) Q is a finite set of states

(ii) $\sum$ is a finite set of symbols, called the alphabet of the automation,

(iii) δ is the transition function, where $\delta : Q \times \sum \rightarrow Q$.

(iv) q_0 is the initial state from where any input is processed ($q_0 \in Q$).

(v) F is a set of final state/states of Q ($F \subseteq Q$).

Graphically, a DFA can be represented by diagraphs, called as state diagram, where :

(i) The states are represented by **vertices.**

(ii) The transitions are shown by labelled **arcs.**

(iii) The initial state is represented by an **empty incoming arc.**

(iv) The final state is represented by **double circle.**

1.7.2.2 Example of DFA

Suppose a DFA be

$Q = \{l, m, n\}$ $\quad \sum = \{0, 1\}$

$q_0 = \{l\}$ $\quad F = \{n\}$

Transition function δ is as shown in the table below.

Current state	Next state for input O	Next state for input 1
L	*l*	M
M	M	L
N	N	N

The graphical representation of this DFA is :

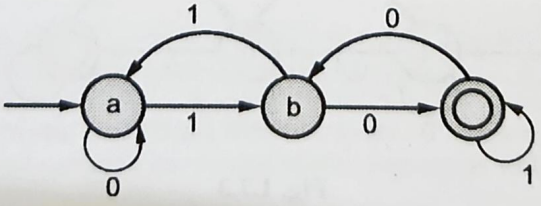


Fig. 1.7.2

1.7.2.3 Non-Deterministic Finite Automation (NDFA)

It is defined as the type of finite automation where for every input symbol we cannot determine the state to which the machine will more. It means that machine can move to any combination of states.

Since it has finite number of states, it is called as Non-deterministic finite Automation (NDFA) machine.

Again, as usual, NDFA can be represented mathematically by a 5-typle (Q, Σ, d, q_0, F), where

(i) Q is a finite set of states,

(ii) Σ is a finite set of symbols, called the alphabet of the automation.

(iii) δ : is the transition function where $\delta : Q \times \Sigma \rightarrow 2\ Q$

(iv) q_0 : is the initial state from where any input is processed ($q_0 \in Q$),

(v) F : is a set of final state / states of Q ($F \subseteq Q$).

Graphically, NDFA can be represented by diagraphs. (same as DFA) and is called as state diagrams where :

(i) The states are represented by **vertices.**

(ii) The transition are shown by labelled **arcs.**

(iii) The initial state is represented by an **empty incoming arc.**

(iv) The final state is represented by double **circle.**

1.7.2.4 Example of NDFA

Let NDFA be :

$Q = \{a, b, c\}$ $\quad \Sigma = \{0, 1\}$

$q_0 = \{a\}$, $\quad f = \{c\}$

We exhibit transition function δ in the table as :

Current state	Next state for input O	Next state for input 1
A	a, b	B
B	C	a, c
c	b, c	C

The graphical representation of NDFA is :

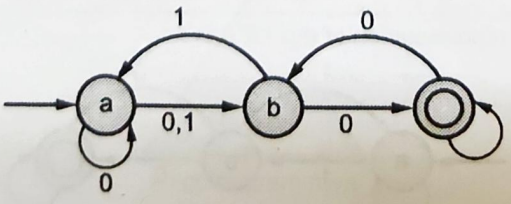


Fig. 1.7.3

1.7.3 Regular Expression and Automate

(I) Role of regular expression and automata in NLP system

The key concept of Natural language Processing is that every NLP expert should be proficient in Regular Expressions. They are used in various tasks such as **data-processing, rule-based information mining systems, pattern matching, text, feature engineering, web scraping, data extraction** etc.

(II) Bag of words in NLP

A bag of words is a representation of text that describes the occurrence of words within a document.

We keep the track of word-counts and disregard the grammatical details and the word order. It is called a "bag" of words because any information about the order or structure of words in the document is discarded.

1.7.4 : (I) Applications and Limitations of Finite Automata

(i) Applications are as follows

(i) For the designing of lexical analysis of a compiler.
(ii) For recognising the pattern using regular expressions.
(iii) For the designing of the combination and sequential circuits using mealy and Moore machines.
(iv) Used in text editors.
(v) for the implementation of spell checkers.

(II) Limitations of Finite Automata

(i) FA can only count finite input,
(ii) There is no finite automata that can find and recognise set of binary string of equal 0s and 1s.
(iii) Set of strings over "("and")" and have balanced parenthesis.
(iv) Input tape is read only and only memory it has is, state of state.
(v) It can have only string pattern.

1.8 LEVELS AND TASKS OF NLP

GQ. Briefly explain the NLP tasks and write the different levels of NLP. Or Explain the synthetic and semantic analysis in NLP.

NLP problem can be divided into two tasks : Processing written text, using lexical, syntactic and semantic knowledge of the language as well as the required real world information.

Processing spoken language, using all the information needed above plus additional knowledge about phonology as well as enough added information to handle the further ambiguities that arise in speech.

Level of NLP

1. Morphology

It is the analysis of individual words that consist of morphemes the smallest grammatical unit. Generally, words with 'ing', 'ed' change the meaning of the word. This analysis becomes necessary in the determination of tense as well.

2. Syntax

Syntax is concerned with the rules. It includes legal formulation of the sentences to check the structures. (Some aspects are covered in compiler's phase of syntax analysis that you must have studied). For example, 'Hari is good not to.' The sentence structure is totally invalid here.

3. Semantic

During this phase, meaning check is carried out- The way in which the meaning is conveyed is analyzed. The previous example is syntactically as well as semantically wrong. Now, consider one more example, i.e., "The table is on the ceiling." This is syntactically correct, but semantically wrong.

4. Discourse integration

In communication or even in text formats, often the meaning of the current sentence is dependent on the one that is prior to it. Disburse analysis deals with the identification of discourse structure.

5. Pragmatic

In this phase, analysis of the response from the user with reference to what actually the language meant to convey is handled. So, it deals with the mapping for what the user has interpreted from the conveyed part arid what was actually expected. For a question like "Do you know how long it will take to complete the job?", the expected answer is the number of hours rather than a yes or no.

6. Prosody

It is an analysis phase that handles rhythm. This is the most difficult analysis that plays an important role in the poetry or shlokus (chants involving the name of God) that follow a rhythm.

7. Phonology

This involves analysis of the different kinds of sounds that are combined. It is concerned with speech recognition. Can the analysis levels discussed be overlapped or interrelated? Yes. It is very much possible to have an analysis actually forming a fuzzy structure. They can work in stages, where the second level makes use of the analysis or the outcomes of the first level. We now study them in detail.

1.9 STAGES IN NLP

There are five phases of NLP (Refer Fig. 1.9.1)

1. Lexical analysis and morphological

- Lexical analysis is the first phased NLP this phase scans the source code os a stream of characters. Then it converts into meaningful lexemes. It divides the whole text into paragraphs, sentence and words.
- It studies the patterns of formation of words. It combines sounds into minimal distinctive units of meaning.

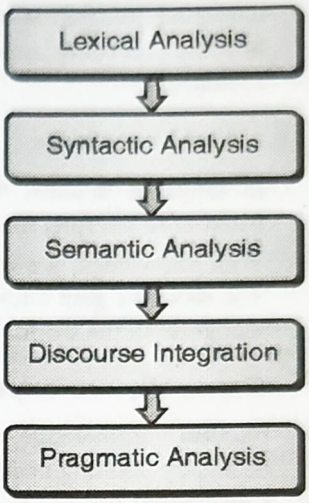


Fig. 1.9.1 : Basic steps of NLP

2. Syntactic analysis (parsing)

- Syntactic analysis is used to check grammar, word arrangements, and shows the relationship among the words. Hence words are collected to form phrases, phrases get converted to clauses and clauses form sentences. It shows the relationship among words.
- **Example :** Pune goes to gopal.
- Pune goes to gopal, does not make any sense, so this sentence is rejected by the syntactic analyser.

3. Semantic analysis

- Semantic analysis is concerned with the meaning representation. It focus on the literal meaning of words, phrases, and sentences. It studies meaning of the words independent of context of the sentence.
- Hence it may involve ambiguities to some extent.

4. Pragmatic Knowledge

- Praggmatic is the last phase of NLP. It helps one to discover the intended effect by applying a set of rules that characterise cooperative dialogues.
- It is mainly concerned with how the sentences are used and what the inner meaning of the sentence is
- **For example :** "Open the door" is interpreted as a request instead of an order.

5. Discourse integration

- Discourse integration depends upon the sentences that proceeds it and also invokes the meaning of the sentences that follow it. It connects sentences.
- Discourse integration mainly studies the inter-sentential connections. It studies how the preceding sentence can change the inter pretation of the next following sentence.

6. World knowledge

- In language studies, the non-linguistic information that helps a reader on listerner to interpret the meanings of words and sentences.
- With knowledge, we are able to recognise things and people around the world. The more we gain knowledge, the more things and people we should be able to recognise in the world.

☞ Generally we experience four types of knowledge

(i) **Factual knowledge :** These are the terminologies, glossaries, details and necessary building details of any professional domain.

(ii) **Conceptual knowledge :** This knowledge is the understanding of the principles and relationships that underlie a domain.

(iii) **Procedural knowledge :** This knowledge refers to the knowledge of how to perform a specific skill or task, and is considered knowledge related to methods, procedures, or operation of equipment.

Procedural knowledge is also referred to as implicit knowledge or know-how.

(iv) **Meta cognitive knowledge :** This knowledge refers to what learners know about learning.

This includes : The learner's knowledge of their our cognitive abilities (e.g 'I have trouble remembering dates in history') the learner's knowledge of particular tasks (e.g. 'the ideas in this chapter that I am going to read ar complex.').

1.9.1 Phonetic and Phonological Knowledge

- Phonetic knowledge is the knowledge of sound-symbol relations and sound patterns represented in a language.
- It is when a child is learning to talk, communicate and then they develop phonemic awareness, which is an awareness of distinctive speech sounds and they use phonemes (smallest unit of sound) to create words.
- The primary differences between phonological and phonemic awareness is that phonological awareness is the ability to recognise words made up of different sounds.
- In contrast, phonemic awareness is the ability to understand how sound functions in words.

Example of phonological knowledge

- Counting the number of syllables in a name, recognising alterations, segmenting a sentence into words, and identifying the syllables in a word.
  Example of phonemic knowledge.
- Counting the number of sounds a word would be a phonemic awareness activity.
- Information retrieval, information extraction and question answering Information retrieval involves returning a set of documents in response to a user query :

Internet search engines are a form of IR. However, one change from classical IR is that Internet search now uses techniques that rank documents according to how many links there are to them (e.g., Google's PageRank) as well as the presence of search terms. Information extraction involves trying to discover specific information from a set of documents. The information required can be described as a template. For instance, for company joint ventures, the template might have slots for the companies, the dates, the products, the amount of money involved. The slot fillers are generally strings. Question answering attempts to find a specific answer to a specific question from a set of documents, or at least a short piece of text that contains the answer. What is the capital of France? Paris has been the French capital for many centuries. There are some question-answering systems on the Web, but most use very basic techniques. For instance, Ask Jeeves relies on a fairly large staff of people who search the web to find pages which are answers to potential questions. The system performs very limited manipulation on the input to map to a known question. The same basic technique is used in many online help systems.

1.10 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 :

(1) **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.

(2) **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.

(3) **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".

(4) **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.

(5) **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).

1.10.1 NLP for Indian Regional Languages

(1) One might think that people who are acquainted with computers are already familiar with the English interface. However, it's worth noting that majority of the Indian population in India is still based in rural areas where teaching and learning would be in local languages, where communities are literate, but still are not familiar with English.

So, yes, it is a worthwhile effort to upscale NLP research in India.

(2) The dream of an all-inclusive Digital India cannot be realized without bringing NLP research and application in India at par with that of languages like English. When engaging with smartphones, the language barrier can be a huge obstacle to many.

(3) Take the case of farmers and agriculture which has long been considered the backbone of India. Farmers play an obviously important role in feeding the country. Helping such farmers improve their methods (through precision agriculture, farmer helplines, chatbots, etc) has been an aim of development projects and an important part of the fight against global hunger. But many small farmers are not knowledgeable in English, meaning it is difficult for them to share and learn about new farming practices since most of the information is in English.

4) Can you imagine a mobile application like Google assistant but tailor-made for Indian farmers? It'd allow them to ask their questions in their native tongue, the system would understand their query and suggest relevant information from around the globe <.

(5) Do you think this is possible to do without NLP for Indian regional languages?

And, this is just one possible use-case. From making information more accessible to understanding farmer suicides [4], NLP has a huge role to play.

Thus, there is a clear need to bolster NLP research for Indian languages so that such people who don't know English can get "online" in the true sense of the word, ask questions, in their mother tongue and get answers.

The need also becomes clear when we look at some of the applications of NLP in India.

1.10.2 Applications of NLP in India

They are :

(1) Smartphone users in India crossed 500 million in 2019. Businesses are feeling a need to increase user engagement at the local level. NLP can go a long way in achieving that-by improving search accuracy(Google Assistant now supports multiple Indian Languages), chatbots and virtual agents, etc.

(2) NLP has huge application in helping people with disabilities-interpretation of sign languages, text to speech, speech to text, etc.

(3) Digitisation of Indian Manuscripts to preserve knowledge contained in them.

(4) Signboard Translation from Vernacular Languages to make travel more accessible.

(5) Fonts for Indian Scripts for improving the impact/readability of advertisements, signboards, presentations, reports, etc.

(6) There are many more. The ideal scenario would be to have corpora and tools available in as good quality as they are for English to support work in these areas.

1.11 CHALLENGES OF NLP

If we have to progress in terms of the potential applications and overall capabilities of NLP, these are the important issues we need to resolve :

(1) **Language Differences :** If we speak English and if we are thinking of reaching an international and / or multicultural andience, we shall need to provide support for multiple languages.

Different languages have not only vastly different sets of vocabulary, but also different types of phrasing, different modes of inflection and different cultural expectations. We shall need to spend time retraining our NLP system for each new languages.

(2) **Training Data :** NLP is all about analysing language to better understand it. One must spend years constantly to become fluent in a language. One must spend a significant amount of time reading, listening to, and utilising a language.

The abilities of an NLP system depends on the training data provided to it.

If questionable data is fed to the system it is going to learn wrong things, or learn in an inefficient way

(3) **Development Time :** One also must think about the development time for an NLP system. With a distributed deep learning mode and multiple GPUS working in coordination, one can trim down the training time to just a few hours.

(4) **Phrasing Ambiguities :** Sometimes, it is hard even for another human being to parse out what someone means when they say something ambiguous. There may not be a clear, concise meaning to be found in a strict analysis of their words.

In order to resolve this, an NLP system must be able to seek context that can help it understand the phrasing. It may also need to ask the user for clarity.

(5) **Misspelling :** Misspellings are a simple problem for human beings, but for a machine, misspellings can be harder to identify. One should use an NLP tool with capabilities to recognise common misspellings of words, and move beyond them.

(6) **Innate Biases :** In some cases, NLP tools can carry the biases of their programmers as well as biases within the data sets.

Depending on the application, an NLP could provide a better experience to certain types of users over others.

It is challenging to make a system that works equally well in all situations, with all people.

(7) **Words with Multiple Meaning :** Most of the languages have words that could have multiple meanings, depending on the context. For example, a user who asks, "how are you" has a totally different goal than a user.

Good NLP tools should be able to differentiate between these phrases with the help of context.

(8) **Phrases with Multiple Intentions :** Some phrases and questions actually have multiple intentions, so the NLP system cannot over simplify the situation by interpreting only one of those intentions.

For example, a user may prompt the Chabot with something like, "I need to cancel any previous order and update my card on file."

The AI needs to be able to distinguish these intentions separately.

(9) **False Positives ad Uncertainty :** A false positive occurs when an NLP notices a phrase that should be understandable but cannot be sufficiently answered.

The solution here is to develop a NLP system that can recognise its own limitations, and use questions to clear up the ambiguity.

(10) **Keeping a conversation moving :** Many modern NLP applications are built on dialogue between a human and a machine.

Accordingly, your NLP AI needs to be able to keep the conversation moving, providing additional questions to collect more information and always pointing towards a solution.

1.12 GENERAL APPLICATIONS OF NLP

- Natural Language processing, machine learning and artificial intelligence are used interchangeably. AI is regarded as an umbrella-term for machines that can simulate human intelligenic NLP and ML are regarded as subsets of AI.
- Natural language processing is a form of AI that gives machines the ability to not just read, but to understand and interpret human language.
- With NLP, machines can make sense of written or spoken text and perform tasks including speech recognition, sentiments analysis, and automatic test summarisation.
- Thus, we can note that NLP and ML are parts of AI and both subsets share techniques, algorithms and knowledge.

- Some NLP-based solutions include translation, speech recognition, sentiment analysis, question/answer systems, chatbots, automatic test summarisation, market intelligence, automatic text classification, and automatic grammar checking.

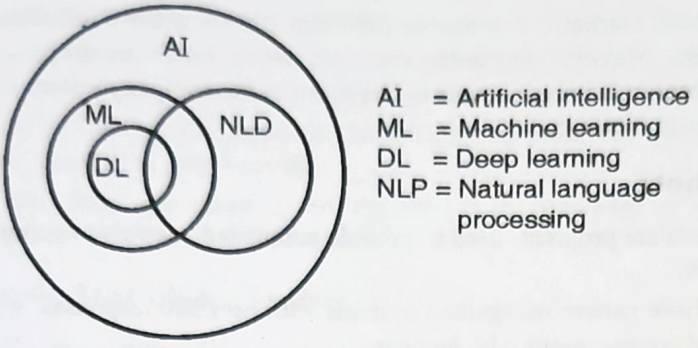


Fig. 1.12.1

- These technologies help organisations to analyse data, discover insights, automate time-consuming processes, and/or gain competitive advantages.

(1) Translation

- Translating languages is more complex task than a simple word-to-word replacement method. Sine each language has grammar rules, the challenge of translating a text is to be done without changing its meaning and style.
- Since computers do not understand grammar, they need a process in which they can deconstruct a sentence, then again reconstruct it in another language in a way that makes sense.
- Google translate is one of the most well-known online translation tools. Google Translate once used phrase-based machine Translation (PBMT), which looks for similar phrases between different languages.
- At present Google uses Google neural machine translation (GNMT), which useds ML with NLP to look for patterns in languages.

(2) Speech Recognition

- Speech recognition is a machine's ability to identify and interpret phrases and words from spoken language and convert them into a machine- readable format.
- It uses NLP to allow computers to collect human interaction, and ML to respond in a way that copies human responses.
- Google Now, alexa, and Siri are some of the most popular examples of speech recognition. Simply by saying 'call Ravi', a mobile recognises what the command means and it makes a call to the contach saved as 'Ravi',

(3) Sentiment Analysis

- Sentiment analysis uses NLP to interpret and analyse emotions in subjective data like news articles and tweets.

- Positive, negative and neutral opinions can be identified to determine a customer's sentiment towards a brand, product, or service.
- Sentiment analysis is used to measure public opinion, monitor brand reputation, and better understand customer experiences.
- The stock market is a sensitive field that can be heavily influenced by human emotion. Negative sentiment con lead stock prices to drop, while positive sentiment may trigger people to buy more of the company's stock, causing stock princes to increase.

(4) Chatbots

- Chatbots are programs used to provide automated answers to common customer queries.
- They have pattern recognition systems with heuristic responses, which are used to hold conversations with humans.
- Initially, chatbots were used to answer basic questions to alleviate heavy volume call centres and offer quick customer support services. AI – powered chatbots are designed to handle more complicated request making conversational experiences increasingly original.
- Chatbots in health-care can collect intake data, help patients to assess their symptoms, and determine next steps. These chatbots can set up appointments with the right doctor and even recommend treatments.

(5) Question- Answer systems

- Question – Answer systems are intelligent systems that can provide answers to customer queries.
- Other than cabtbost, question-answer systems have a huge array of knowledge and good language understanding rather than canned answers. They can answer questions like "When as Indira Gandhi assassinated ?", or "How do I go to the Airport ?" and it can be created to deal with textual data, and audio, images and videos.
- Question – answer systems can be found in social media chats and tools such as siri and IBM's Waston.
- In 2011, IBM's Watson computer competed on Jeopardy, a game show during which answers are given first, and the contestants supply the questions. The computer connected against the show's two biggest all time champions and astounded the tech industry as it won first place.

(6) Automatic Text Summarisation

- Automatic text summarisation is the task of condensing a piece of text to a shorter version. It extracts its main ideas and preserving the meaning of content.
- This application of NLP is used in news headlines result snipers in web search, and buttetins of market reports.

(7) Market Intelligence

- Market intelligence is the gathering of valuable insights surrounding trends, consumers, products and competitors. It extracts action able information that can be used for strategic decision-making.
- Market intelligence can analyse topics, sentiment, keywords, and intent in unstructured data and is less time consuming than traditional desk research.
- Using Market intelligence, organizations can pick up on search queries and add relevant synonyms to search results.
- It can also help organisations to decide which products or services to discontinue or what to target to customers.

(8) Automatic Text Classification

- Automatic text classification is another fundamental solution of NLP. It is the process of assigning tags to text according to its content and semantics. It allows for rapid, easy collection of information in the search phase.
- This NLP application can differentiate span from non-spam based on it content.

(9) Automatic Grammar Checking

- Automatic grammar checking is the task of detecting and correcting grammatical errors and spelling mistakes in text depending on context, is another major part of NLP.
- Automatic grammar checking will make one alert to a possible error by underling the word in **red**.

(10) Span Detection

Span detection is used to detect unwanted e-mails getting to a user's inbox. Refer Fig. 1.12.2

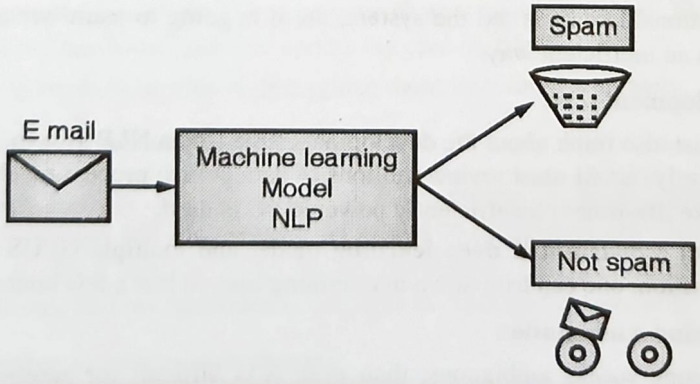


Fig. 1.12.2

(11) Information extraction

Information extraction is one of the most important applications of NLP.

It is used for extracting structured information from unstructured or semi-structural machine-readable documents.

(12) Natural Language Understand (NIU)

It converts a large set of text into more formal representations such as first-order logic structures that are easier for the computer programs to manipulate notations of the natural language processing.

1.13 ISSUES IN NLP

With the help of complex algorithms and intelligent analysis, NLP tools pave the way for digital assistants, chatbots, voice search, and dozens of applications. Even then there are some of the most important issues that we have to resolve. They are as follows :

1. Language differences

- In USA, people speak English, but if we are thinking of reaching multicultural audience, we shall need to provide support for multiple languages.
- Different languages have not only vastly different sets of vocabulary but also different types of phrasing, different modes of inflection, and different cultural expectations.
- This issue can be resolved with the help of "Universal" models that can transfer at least some learning to other languages. But we shall need some time to train NLP system for each new language.

2. Training data

- The main aim of NLP is to analyse language to better understand it. To be fluent in a language, one must immerse in a language constantly for a period of years.
- Similarly, even the best AI must also spend a significant amount of time reading, listening to and utilising a language.
- The ability of an NLP system depends on the training data provided to it. If bad or questionable data is fed the system, the it is going to learn wrong things, or learn in an inefficient way.

3. Development time

- One must also think about the development time for an NLP system. To train AI sufficiently, an AI must review millions of data points; processing all those data may take life-time if insufficiently powered PC is used.
- But with a **distributed deep learning** model and multiple GPUS working in coordination, one can trim down that training time to just a few hours.

4. Phrasing Ambiguities

- If someone speaks ambiguous, then even it is difficult for another person to parse out what one means.
- There may not be a clear, concise meaning to be found in a strict analysis of their words, In order to resolve this, an NLP system must be able to seek content that can help it understand the phrasing. one may also need to ask the user for clarity.

5. Misspellings

- For human beings, misspellings are not a very big problem. One can easily associate a misspelled word with its properly spelled counterpart, and understand the vest of the sentence in which it is used.
- But for a machine, misspellings can be harder to identity.
- Hence we need to use an NLP tool with capabilities to recognise common misspellings of words, and more beyond them.

6. Innate biases

- In some cases, NLP tools can carry the biases of their programmers, as well as biases within the data sets, that are used to train them.
- Depending on the application, an NLP could exploit certain biases. It is challenging to make a system that works equally well in all situations, with all people.

7. Words with Multiple Meanings

- Many of the languages have words that have multiple meanings, depending upon the context.
- For example, a user who asks, "now are you" has a totally different goal than a user who asks something like "how do I add a new debit card ?"
- Good NLP tools should be able to differentiate between these phrases with the help of content.

8. Phrases with Multiple Intentions

- Some questions and phrases have multiple intentions. In such a case NLP system cannot oversimplify the situation by interpreting only one of those intentions.
- For example, a user may prompt your chatbot with something like, "I need to cancel my previous order and update my card on file."
- Here AI needs to be able to distinguish these intentions separately.

9. False positive and uncertainty

- A false positive occurs when an NLP notices a phrase that should be understandable and addressable, but cannot be sufficiently answered.
- Here an NLP system is to be so developed, that can recognise its own limitations, and use questions or prompts to clear up the ambiguity.

10. Keeping a conversation moving

- Many of the modern NLP applications are built an dialogue between a human and a machine. Hence, our NLP AI needs to be able to keep the conversation moving, providing additional questions to collect more information and always pointing towards a solution.
- Here, we have discussed the major challenges of using NLP.

1.14 TOKENIZATION

- Tokenization is a common task in Natural Language Processing (NLP). It's a fundamental step in both traditional NLP methods like Count Vectorizer and Advanced Deep Learning-based architectures like Transformers.
- Tokens are the building blocks of Natural Language.
- Tokenization is a way of separating a piece of text into smaller units called tokens. Here, tokens can be either words, characters, or subwords. Hence, tokenization can be broadly classified into 3 types – word, character, and subword (n-gram characters) tokenization.
- For example, consider the sentence: "Never give up".
- The most common way of forming tokens is based on space. Assuming space as a delimiter, the tokenization of the sentence results in 3 tokens – Never-give-up. As each token is a word, it becomes an example of Word tokenization.
- Similarly, tokens can be either characters or subwords. For example, let us consider "smarter":
  1. Character tokens: s-m-a-r-t-e-r
  2. Subword tokens: smart-er
- But then is this necessary? Do we really need tokenization to do all of this?

1.14.1 Reasons behind Tokenization

- As tokens are the building blocks of Natural Language, the most common way of processing the raw text happens at the token level.
- For example, Transformer based models – the State of The Art (SOTA) Deep Learning architectures in NLP – process the raw text at the token level. Similarly, the most popular deep learning architectures for NLP like RNN, GRU, and LSTM also process the raw text at the token level.

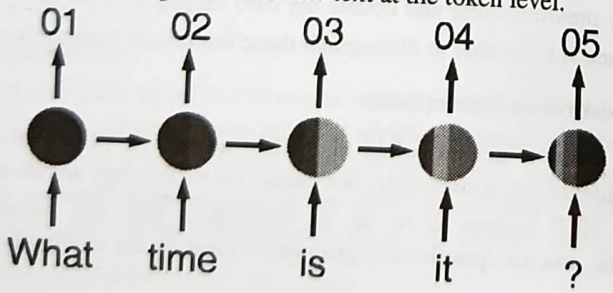


Fig. 1.14.1

Examples

- Tokenization is the process of replacing sensitive data with symbols that preserve all the essential information about the data without compromising its security.

- Tokenisation tries to minimise the amount of data a business needs to have at hand. It has become popular for small and midsize businesses to bolster the security of **credit card** and **e-commerce** transactions while minimising cost and complexity of **compliance** with industry standards and government regulations.
- Tokenization technology can be used with sensitive data of all kinds, including bank transactions, medical records, criminal records, vehicle driver information, loan applications, stock trading and voter registration.
- Tokenisation is often used to protect credit card data, bank account information and other sensitive data handled by a payment processor.
- Payment processing cases that use tokenize sensitive credit information include :
  (i) mobile wallets like android pay and apple pay.
  (ii) e-commerce sites; and
  (iii) businesses that keep a customer's card on file.

1.14.2 Token

GQ. What is Token ? How it is Created ?

- Tokenization substitutes sensitive information with equivalent non sensitive information. The nonsensitive, replacement information is called a **token**.
- Tokens can be created in various ways :
  (i) Using a mathematically reversible cryptographic function with a key,
  (ii) Using a non reversible function such as a hash function,
  (iii) Using an index function or randomly generated number.
- In short, the token becomes the exposed information, and the sensitive information that the token stands in for is stored safely in a centralised server known as a token **vault**. The original information can only be traced back to its corresponding token from the token vault.
- Some tokenisation is vault less. Instead of storing the sensitive information in a secure database, vault less tokens are stored using an algorithm.
- If the token is reversible, then the original sensitive information is generally not stored in a vault.
- We mention a real world example of how tokenization with a token vault works :
  (i) A customer provides their payment details at a point-of-sale (POS) system or online checkout form.
  (ii) The data are stored with a randomly generated token, which is generated in most cases by the merchant's **payment gateway**.
  (iii) The tokenized information is then cent to a payment processor.
  The original sensitive payment information is stored in a token vault in the merchant's payment gateway.

(iv) The tokenised information is sent again by the payment processor before being sent for final verification.

1.14.3 Word Tokenization

- Word Tokenization is the most commonly used tokenization algorithm. It splits a piece of text into individual words based on a certain delimiter.
- Depending upon delimiters, different word-level tokens are formed. Pretrained Word Embeddings such as Word2Vec and GloVe comes under word tokenization.
- But, there are few drawbacks to this.

Drawbacks of Word Tokenization

- One of the major issues with word tokens is dealing with Out Of Vocabulary (OOV) words. OOV words refer to the new words which are encountered at testing. These new words do not exist in the vocabulary. Hence, these methods fail in handling OOV words.
- But wait – don't jump to any conclusions yet!
- A small trick can rescue word tokenizers from OOV words. The trick is to form the vocabulary with the Top K Frequent Words and replace the rare words in training data with unknown tokens (UNK). This helps the model to learn the representation of OOV words in terms of UNK tokens
- So, during test time, any word that is not present in the vocabulary will be mapped to a UNK token. This is how we can tackle the problem of OOV in word tokenizers.
- The problem with this approach is that the entire information of the word is lost as we are mapping OOV to UNK tokens. The structure of the word might be helpful in representing the word accurately. And another issue is that every OOV word gets the same representation
- Another issue with word tokens is connected to the size of the vocabulary. Generally, pre-trained models are trained on a large volume of the text corpus. So, just imagine building the vocabulary with all the unique words in such a large corpus. This explodes the vocabulary!
- This opens the door to Character Tokenization.

1.14.4 Character Tokenization

- Character Tokenization splits apiece of text into a set of characters. It overcomes the drawbacks we saw above about Word Tokenization.
- Character Tokenizers handles OOV words coherently by preserving the information of the word. It breaks down the OOV word into characters and represents the word in terms of these characters
- It also limits the size of the vocabulary. Want to talk a guess on the size of the vocabulary? 26 since the vocabulary contains a unique set of characters

Drawbacks of Character Tokenization

- Character tokens solve the OOV problem but the length of the input and output sentences increases rapidly as we are representing a sentence as a sequence of characters. As a result, it becomes challenging to learn the relationship between the characters to form meaningful words.
- This brings us to another tokenization known as Subword Tokenization which is in between a Word and Character tokenization.

1.14.5 Need Of Tokenization

GQ. Why do we need tokenization ?

- Tokenization is the first step in any NLP pipeline. It has an important effect on the rest of your pipeline. A tokenizer breaks unstructured data and natural language text into chunks of information that can be considered as discrete elements. The token occurrences in a document can be used directly as a vector representing that document.
- This immediately turns an unstructured string (text document) into a numerical data structure suitable for machine learning. They can also be used directly by a computer to trigger useful actions and responses. Or they might be used in a machine learning pipeline as features that trigger more complex decisions or behavior.
- Tokenization can separate sentences, words, characters, or subwords. When we split the text into sentences, we call it sentence tokenization. For words, we call it word tokenization.

Example of sentence tokenization

```
Sent_totenize ('Life is a matter of choices, and every choice you make makes you.')
```

Example of word tokenization

```
Word_totenize ("The sole meaning of life is to serve humanity")
['The', 'sole', 'meaning', 'of', 'life', 'is', 'to, 'serve', 'humanity']
```

1.14.6 Benefits of Tokenization

GQ. What are the benefits of tokenization ?

(i) Tokenization makes it more difficult for **hackers** to gain access to cardholder data.

In older systems, credit card numbers were stored in **databases** and exchanged freely over **networks**.

(ii) It is more compatible with legacy systems than encryption.

(iii) It is a less resource-intensive process than encryption.

(iv) The risk of the fallout in a data breach is reduced.

(v) The payment industry is made more convenient by allowing new technologies like mobile wallets, one-click payment and crypto currency. This improves customer trust because it improves both the security and convenience of a merchant's service.

(vi) It reduces the steps involved in complying regulations for merchants.

1.14.7 Tokenization Challenges in NLP

- While breaking down sentences seems simple, after all we build sentences from words all the time, it can be a bit more complex for machines.
- A large challenge is being able to segment words when spaces or punctuation marks don't define the boundaries of the word. This is especially common for symbol-based languages like Chinese, Japanese, Korean, and Thai.
- Another challenge is symbols that change the meaning of the word significantly. We intuitively understand that a '$' sign with a number attached to it ($100) means something different than the number itself (100). Punction, especially in less common situations, can cause an issue for machines trying to isolate their meaning as a part of a data string.
- Contractions such as 'you're' and 'I'm' also need to be properly broken down into their respective parts. Failing to properly tokenize every part of the sentence can lead to misunderstandings later in the NLP process.
- Tokenization is the start of the NLP process, converting sentences into understandable bits of data that a program can work with. Without a strong foundation built through tokenization, the NLP process can quickly devolve into a messy telephone game.

Sub Word Tokenization

- Sub word tokenization is similar to word tokenization, but it breaks individual words down a little bit further using specific linguistic rules. One of the main tools they utilize is breaking off affixes. Because prefixes, suffixes, and infixes change the inherent meaning of words, they can also help programs understand a word's function. This can be especially valuable for out of vocabulary words, as identifying an affix can give a program additional insight into how unknown words function.
- The sub word model will search for these sub words and break down words that include them into distinct parts. For example, the query "What is the tallest building?" would be broken down into 'what' 'is' 'the' 'tall' 'est' 'build' 'ing'
- How does this method help the issue of OOV words? Let's look at an example:
- Perhaps a machine receives a more complicated word, like 'machinating' (the present tense of verb 'machinate' which means to scheme or engage in plots). It's unlikely that machinating is a word included in many basic vocabularies.

- If the NLP model was using word tokenization, this word would just be converted into just an unknown token. However, if the NLP model was using sub word tokenization, it would be able to separate the word into an 'unknown' token and an 'ing' token. From there it can make valuable inferences about how the word functions in the sentence.
- But what information can a machine gather from a single suffix? The common 'ing' suffix, for example, functions in a few easily defined ways. It can form a verb into a noun, like the verb 'build' turned into the noun 'building'. It can also form a verb into its present participle, like the verb 'run' becoming 'running.'
- If an NLP model is given this information about the 'ing' suffix, it can make several valuable inferences about any word that uses the sub word 'ing.' If 'ing' is being used in a word, it knows that it is either functioning as a verb turned into a noun, or as a present verb. This dramatically narrows down how the unknown word, 'machinating,' may be used in a sentence.
- There are multiple ways that text or speech can be tokenized, although each method's success relies heavily on the strength of the programming integrated in other parts of the NLP process. Tokenization serves as the first step, taking a complicated data input and transforming it into useful building blocks for the natural language processing program to work with.
- As natural language processing continues to evolve using deep learning models, humans and machines are able to communicate more efficiently. This is just one of many ways that tokenization is providing a foundation for revolutionary technological leaps.

1.14.8 Discuss Types of Tokens

There are three main types of tokens as defined by securities and exchange commission.

(i) **Asset/security token :** These are tokens that promise a positive return on an investment. These are analogous to bands and equities.

(ii) **Utility token :** These act as something other than a means of payment. For example, a utility token may give direct access to a product, or as a discount on future goods and services. It adds value to the functioning of a product.

(iii) **Currency / Payment token :** These are created totally as a means of payment for goods and services external to the platform they exist on.

1.15 STEMMING

- Stemming is a natural language processing technique. It lowers inflection in words to their root forms; it aids in the preprocessing of text, words and documents for text normalisation.
- Inflection is the process by which a word is modified to communicate many grammatical categories, including tense, gender and **mood**.

- We employ stemming to reduce words to their basic form or stem, which may or may not be a legitimate word in the language.
- For example, the stem of these three words, connections, connected connects, is "connect".
- On the other hand, the root of trouble, troubled, and troubles is "troubl", which is not a recognised word.
- English language has several variants of a single term. The presence of these variances in a text-corpus results in data redundancy when developing NLP or machine learning models. Such models may become ineffective.
- To build a robust model, it is essential to normalise test by removing repetition and transforming words to their base form through stemming.
- Stemming is a rule-based approach that produces variants of a root/base word. In simple words, it reduces a base word to its stem word. This heuristic process is the simpler of the two as the process involves indiscriminate cutting of the ends of the words. Stemming helps to shorten the look-up and normalise the sentences for a better understanding.

1.15.1 Challenges in Stemming

The process has two main challenges :

- **Over stemming :** The inflected word is cut off so much that the resultant stem is nonsensical. Over stemming can also result in different words with different meanings having the same stem. For example, "universal", "university" and "universe" is reduced to "univers". Here, even though these three words are etymologically related, their modern meanings are widely different. Treating them as synonyms in a search engine will lead to inferior search results.
- **Understemming :** Here, various inflected words have the same stem despite different meanings. The issue crops up when we have several words that actually are forms of one another. An example of understemming in the Porter stemmer is "alumnus" → "alumnu", "alumni" → "alumni", "alumna"/"alumnae" → "alumna". The English word has Latin morphology, and so these near-synonyms are not combined.

1.15.2 Application of Stemming

In information retrieval, text minimise SEOs, web search results, indexing, tagging systems, and word analysis, stemming is employed. For instance, a Goolge search for prediction and predicted returns comparable results.

1.15.3 Types of Stemmer in NLTK

GQ. Discuss different types of stemmer in NLTK.

There are different kinds of stemming algorithms, and all of them are included in python NLTK, we discuss them.

1.15.3.1 Porter Stemmer

- Here five steps of word reduction are used in this method. Each step has its own mapping rules.
- Frequently, the resultant stem is a shorter word with the same root meaning
- porter stemmer is the renowned stemmer for its ease of use and rapidity.
- Porter stemmer ( ) is a module in NLTK that implements the porter stemming technique. **We consider an example :**
- We construct an instance of porter stemmer ( ) and use the porter algorithm to stem the list of words.

```
From nltk.setm import porter stemmer
Porter = porter stemmer ( )
Words =['connects', 'connecting', 'connections' 'Connected', 'connection',
'connecting',
        'connect']
for word in words :
Print (word, "→" porter, stem (word))
```

Output

```
Connects → connect
Connecting → connect
Connections → connect
Connected → connect
Connection → connect
Connecting → connect
Connects → connect
```

1.15.3.2 Snowball Stemmer

Snowball Stemmer ()

- The method used in this instance is more precise and is referred to as "English stemmer" or "porter 2 stemmer".
- It is rather faster and more logical than the original porter stemmer.
- Snowball stemmer ( ) is a module in IV LTK that implements the snowball stemming technique.

Example of Snowball stemmer ( )

We first construct an instance of snowball stemmer ( ) to us the snowball algorithm to stem the list of words.

```
From nltk.stem import snowball stemmer
Snowball = snowball stemmer (language = 'english')
Words = ['generous', 'generate', 'generously', 'generation' ]
For word in words :
Print (word, "→", snowball stem (word))
```

[Out] :

```
generous → generous
generate → generat
generously → generous
generation → generat
```

1.15.3.3 Lancaster Stemmer

Lancaster stemmer ( )

Lancaster stemmer is straightforward, although it often produces results with excessive stemming. Over-stemming renders stems non-linguistic or meaningless.

Example of Lancaster stemmer ( )

We construct an instance of Lancaster stemmer ( ) and then use Lancaster algorithm to stem the list of words.

```
From nltk. Stem import Lancaster stemmer
Lancaster = Lancaster stemmer ( )
Words = ['eating', 'eats', 'eaten', 'puts', 'putting']
For word in words :
Print (word, "→", Lancaster : stem (word))
```

[Out] :

```
eating → eat
eats → eat
eaten → eat
puts → put
putting → put
```

1.15.3.4 Regexp stemmer-regexp stemmer ( )

- Regexp stemmer identifies morphological affixes using regular expressions. Substrings matching the regular expressions will be discarded.
- Regexpstemmer ( ) is a module in NLTK that implements the regex stemming technique.

Example of Regexstemmer ( )

Here, we first construct an object of Regexp stemmer ( ) and then use the regex stemming method to stem the list of words.

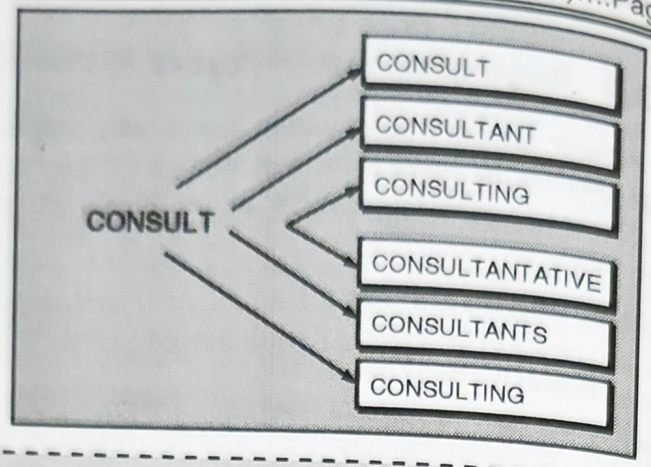
```
From nltk.stem import regexpstemmer
Regexp = regexpstemmer ('ing $|s $| e$|able $', min = 4)
Words = ['mass', 'was', 'bee', 'computer', 'advisable']
For word in words :
Print (word, "→", regexp.stem (word))
```

[Out]

```
mass → mas
was → was
bee → bee
computer → computer
advisable → advis
```

1.15.4 Text Stemming

- As already mentioned, stemming is the process of reducing inflexion in words to their "root" forms, such as mapping a group of words to the same stem. Stem words mean the suffix and prefix that have been added to the root word.
- In computer science, we need this process to produce grammatical variants of root words. A stemming is provided by the NLP algorithms that are stemming algorithms or stemmers. The stemming algorithm removes the stem from the word. For example, 'walking', 'walks', 'walked 'are made from the root word 'Walk'. So here, the stemmer removes ing, s, ed from the above words to take out the meaning that the sentence is about walking in somewhere or on something. The words are nothing but different tenses forms of verbs.
- Below is an example of stem 'Consult.' see how addition of different suffixes generated longer form of the same stem.
- This is the general idea to reduce the different forms of the word to their root word.
- Words that are derived from one another can be mapped to a base word or symbol, especially if they have the same meaning.



GQ. What are the most common types of error associated with text stemming in text mining or NLP?

- We can not be sure that it will give us a 100% result, so we have two types of error in stemming : over stemming and under stemming.

GQ. What is Over stemming error?

- This kind of error occurs when there are too many words cut out. It may be possible that the segmentation of the long-form word may give birth to two such stems that are identical but may actually differ in contextual meaning. These could be known as nonsensical items, where the meaning of the word has lost, or it can not be able to distinguish between two stems or resolve the same stem where they should differ from each other.
- For example, take out the four words university, universities, universal, and universe. A stemmer that resolves these four stems to "Univers" is over-stemming. It should be the universe stemmer that stemmed together, and university, universities stemmed together they all four are not fit for the single stem.

GQ. What is Under stemming error ?

- Under-stemming is the opposite of stemming. It comes from when we have different words that actually are forms of one another. It would be nice for them to all resolve to the same stem, but unfortunately, they do not.
- This can be seen if we have a stemming algorithm that stems from the words data and datum to "dat" and "datu." And you might be thinking, well, just resolve these both to "dat." However, then what do we do with the date? And is there a good general rule? So there under stemming occurs.

1.16 LEMMATIZATION

- Lemmatization is the process of grouping together the different inflected forms of a word so that they can be analysed as a single item.

- Lemmatization is similar to stemming but it brings context to the words, so it links words with similar meaning to one word. Some people treat these two as same. But, lemmatization is preferred over stemming because lemmatization does morphological analysis of words.
- Lemmatization is responsible for grouping different inflected forms of words into the root form, having the same meaning.
- Tagging systems, indexing, SEOs, information retrieval, and web search all use lemmatization to a vast extent.
- Lemmatization involves using a vocabulary and morphological analysis of words, removing inflectional endings, and returning the dictionary form of a word (the lemma).
- The process of lemmatization seeks to get rid of inflectional suffixes and prefixes for the purpose of bringing out the word's dictionary form.

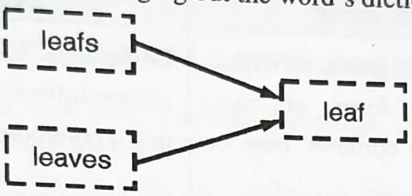


Fig. 1.16.1

- **Lemmatization** entails reducing a word to its canonical or dictionary form. The root word is called a 'lemma'. The method entails assembling the inflected parts of a word in a way that can be recognised as a single element. The process is similar to stemming but the root words have meaning.
- Lemmatization has applications in :
  1. Biomedicine: Using lemmatization to parse biomedicine literature may increase the efficiency of data retrieval tasks.
  2. Search engines
  3. Compact indexing: Lemmatization is an efficient method for storing data in the form of index values.
- For example, NLTK provides Word Net Lemmatizer class– a slim cover wrapped around the word net Corpus. This class makes use of a function called Morphy() to the Word Net Corpus Reader class to find a root word/lemma.

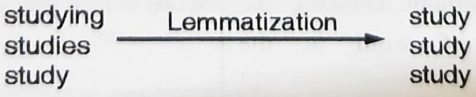


Fig. 1.16.2

1.16.1 Uses of Lemmatization

GQ. What are the uses of Lemmatization.

- Lemmatization helps chat bots to understand customer's queries to a better extent.

- Since this involves a morphological analysis of the words, the chat bots can understand the contextual form of the words in the text. And it can gain a better understanding of the overall meaning of the sentence that is being lemmatized.
- Lemmatization is also used to enable robots to speak and converse. This makes lemmatization a rather important part of natural language processing.

1.16.2 Difference Between Stemming And Lemmatization

GQ. State the difference between stemming and lemmatization.

Sr. No.	Stemming	Lemmatization
1.	Stemming attempts to reduce inflectional form of each word into a common base or root	Lemmatization also attempts to reduce inflectional form of each word into a common base or root.
2.	In stemming the end or beginning of a word is cut off, keeping common prefixes and suffixes.	Lemmatization uses dictionaries to conduct a morphological analysis of the word and link it to the lemma.
3.	One stem can be common for inflectional forms of many lemma can be linked to forms with different stems.	Lemmatization involves greater complexity. It is because the process needs the words to be classified by a part of speech and the inflected form. This is quite difficult task in any language.
4.	Stemming tends to be faster process because it chops words without knowing the context of word in the sentence.	Lemmatization is a slow process, it is because it knows the context of the word before processing.
5.	Stemming is a rule-based approach.	Lemmatization is a dictionary based approach.
6.	The process of stemming has a lower degree of accuracy.	The process of lemmatization has comparatively a higher degree of accuracy.

Stemming vs Lemmatization

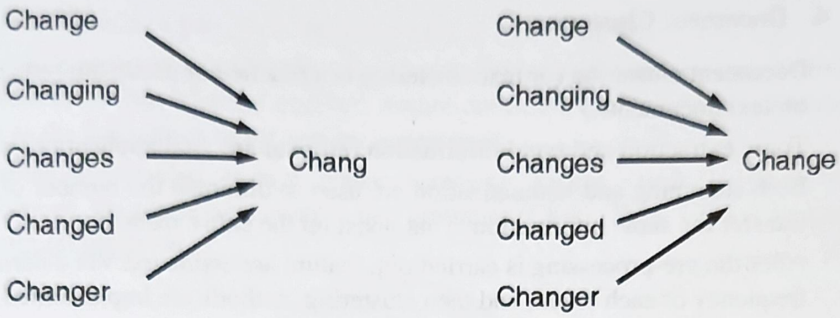


Fig. 1.16.3

1.16.3 Importance of Lemmatization

(i) Lemmatization is a vital part of natural Language Understanding (NLU) and Natural Language Processing (NLP).

(ii) It plays critical roles both in Artificial Intelligence and big data analysis.

(iii) Lemmatization is extremely important because it is for more accurate than stemming. This brings great value when working with a chatbot where it is crucial to understand the meaning of a user's message.

The major disadvantage to lemmatization algorithm is that they are much slower than stemming algorithms.

1.16.4 Applications of Lemmatization

The process of lemmatizastion is used extensively in test mining. The test mining process enables computers to extract relevant information from a particular set of text.

Some of the other areas where lemmatization can be used are as follows :

1. Sentiment analysis

- Sentiment analysis refers to an analysis of people's messages, reviews or comments to understand how they feel about something before the text is analysed, it is lemmatized

2. Information retrieval environments

- Lemmatizing is used for the purpose of mapping documents to common topics and displaying search results. To do so, indexes when documents are increasing to large numbers.

3. Biomedicine

- Lemmatization can be used while morphologically analysing biomedical literature. The Biolemmatizer tool has been for this purpose only.

- It pulls lemmas based on the use of a word lexicon. But if the word is not found in the lexicon, it defines the rules which turn the word into a lemma.

4. Document Clustering

- Document clustering (or text clustering is a practice of group analysis conducted on text documents.)
- Topic extraction and rapid information retrieval are vital applications of it.
- Both stemming and lemmatization are used to diminish the number of tokens to transfer the same information. That boost up the entire method.
- After the pre-processing is carried out, feature are estimated Via determining the frequency of each token, and then clustering methods are implemented.

5. Search engines

- Search engines like Google make use of lemmatization so that they can provide better, more relevant results to their users.
- Lemmatization even allows search engines to display relevant results and even expand them to include other information that reader may find useful.

1.16.5 Advantages and Disadvantages of Lemmatization

Advantages

(i) Lemmatization is more accurate

(ii) It is useful to get root words from the dictionary, unlike just cutting the words like stemming.

(iii) Lemmatization gives more context to chatbot conversations as it recognises words based on their exact and contextual meaning.

Disadvantages

(i) Lemmatization is a time-consuming and slow process.

(ii) As it extracts the root words and meaning of the words from the dictionary, so most lemmatization algorithms are slower compared to their stemming counter parts.

1.16.6 Example of Lemmatization

Running → Run
Runs → Run
Run → Run
} common root

Creating → Create
Creates → Create
Created → Create
} common root

The boy's cakes are of different sizes. The boy cake be of differ size.

1.17 SYNTAX ANALYSIS

- Syntax analysis or parsing is the second phase of a compiler.
- A lexical analyser can identify tokens with the help of regular expressions and pattern rules. But a lexical analyser cannot check the syntax of a given sentence due to the limitations of the regular expressions.
- Regular expressions cannot check balancing tokens, such as parenthesis. Therefore, the phase uses context free grammar (CFG). Thus, CFG is a superset of regular grammar.

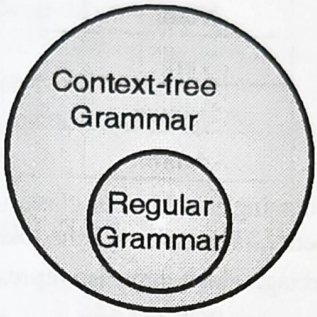


Fig. 1.17.1

- The diagram implies that every regular grammar is also context-free CFG is an important tool which describes the syntax of programming language.

1.17.1 Context-Free Grammar

A context-free grammar has four components :

(i) A set of **non-terminals** (V) : Non-terminals are syntactic variables that denote sets of strings.

The non-terminals define sets of strings that help define the language generated by the grammar

(ii) A set of tokens, known as **terminal symbols** ($\sum$) : Terminals are basic symbols from which strings are formed.

(iii) A set of **productions** (P) : The productions of a grammar specify the manner in which the terminals and non-terminals can be combined to form strings.

Each production consists of a **non-terminal** and is called as left side of the production, an arrow, and a sequence of tokens and/or **on-terminals**, called the right side of the production.

(iv) One of the non-terminals is designed as the start symbol (S), from where the production begins.

The strings are derived from the start symbol by replacing repeatedly a non-terminal (initially the start symbol) by the right side of a production, for that non-terminal.

1.17.2 Part-Of-Speech Tagging (POS)

GQ. Explain POS tagging.

- Part-Of-Speech (POS) Tagging is a process of converting a sentence to forms list of words, list of tuples (where each tuple is having a form (word, tag)).
- The tag is a part-of-speech tag and it signifies whether the word is a noun, adjective, verb and so on.

Part of Speech	Tag
Noun	n
Verb	v
Adjective	a
Adverb	r

- We can also say that tagging is a kind of classification that may be defined as the automatic assignment of description to the tokens.
- The descriptor is called tag, which may also represent semantic information.
- In simple words, we say that POS tagging is a task of labeling each word in a sentence with its appropriate part of speech.
- We have mentioned that parts of speech include nouns, verb, adverb, pronouns, adjectives, conjunction and so on.
- Most of the POS tagging falls under Rule-Base POS tagging, stochastic POS tagging and transformation based tagging.

1.17.3 Rule-based POS Tagging

GQ. Explain Rule-based POS tagging.

- Rule-based taggers use dictionary or lexicon for obtaining 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.
- Rule-based tagging can handle any disambiguity by analyzing the linguistic features of a word. And that is done by its preceding as well as following words.
- For example, if the preceding word of a word is article or adjective, then the word must be a noun.
- All such kind of information in rule-based POS tagging is coded in the form of rules.
- These rules may be either :
  - (i) Context-pattern rules,
  - (ii) Regular expression compiled into finite state automata, and is intersected with lexically ambiguous sentence representation.

Rule-based POS tagging can be visualised by its two-stage architecture :

(i) **First stage :** Here dictionary is used to assign each word a list of potential parts-of-speech.

(ii) **Second stage :** Here, the method uses large list of hand-written disambiguation rules to sort down the list to a single part-of-speech for each word.

1.17.3.1 Properties of Rule-based POS Tagging

We mention below the properties of Rule-based POS taggers :

(i) These taggers are knowledge-driven taggers.
(ii) The rules in Rule-based POS tagging are done manually.
(iii) There are around 1000 number of rules.
(iv) Smoothing and language modelling are defined in rule-based taggers and is done explicitly.

1.17.4 Stochastic POS Tagging

GQ. Explain Stochastic POS Tagging.

- Stochastic model is the model that includes frequency or probability (statistics).
- Different approaches to the problem of a model that includes probability to the problem of part-of-speech tagging is referred to as stochastic tagger.
- The simplest stochastic tagger uses the following approaches for POS-tagging :

(i) Word-Frequency Approach

- Here the stochastic taggers disambiguate the words; i.e. the words based on the probability that a word occurs with a particular tag.
- The tag that is encountered most frequently with the word in the training set is assigned to an ambiguous instance of that word.
- The main problem with this approach is that it may yield inadmissible sequence of tags.

(ii) Tag sequence probabilities

- This is a different approach of stochastic tagging. Here the tagger calculates the probability of a given sequence of tags occurring.
- The best tag for a given word is determined by the probability at which it occurs with previous n tags. Hence it is also called as n-gram approach.

1.17.4.1 Properties of Stochastic POS Tagging

We mention below its properties :

(i) The POS tagging is based on the probability of tag occurring.
(ii) Training corpus is required here.

(iii) If the words do not exist in the corpus, then there is no probability.
(iv) Different testing corpus, other than training corpus, are used.
(v) It is the simplest POS tagging because it chooses most frequent tags associated with a word in training corpus.

1.17.5 Transformation-based Lagging (TBL)

- Transformation based tagging is also called Brill tagging.
- It is the instance of the transformation-based learning. It is a rule-based algorithm for automatic tagging of POS to the given text.
- TBL allows to have linguistic knowledge in a readable form. It transforms one state to another state by using transformation rules.
- TBL can be though of as the mixture of both the above-mentioned taggers-rule-based and stochastic.
- Like rule-based tagging, it is also based on the rules that specify which tags need to be assigned to which words.
- Also we can see similarity between stochastic and transformation tagger. Similar to stochastic, it is machine learning technique-in which rules are automatically induced from data.

1.17.5.1 Working of Transformation Based Learning (TBL)

- To understand the concept governing transformation-based taggers, we have to understand the working of transformation-based learning.
- We mention below the steps of the working of TBL :
  (i) **Begin with the solution :** The TBL usually starts with some solution to the problem and works in cycles.
  (ii) **Choosing most beneficial transformation :** In each cycle, TBL will choose the most beneficial transformation.
  (iii) **Applying to the problem :** The transformation that is chosen in the last step will be applied to the problem.
- The algorithm comes to an end when the selected transformation in step (ii) will not require any further transformation to be selected.

1.17.5.2 Advantages of Transformation-based Learning (TBL)

We mention below the advantages :
(i) We have to learn small set of simple rules and these rules are enough for tagging.
(ii) Since the learned rules are very easy to understand, hence development and debugging is very easy in TBL.

(iii) As in TBL there is interlacing of machine learned and human-generated rules, its complexity is reduced.

(iv) Transformation-based tagger is much faster than Markov-model tagger.

1.17.5.3 Disadvantages of Transformation-based Learning (TBL)

The disadvantages are as follows :

(i) Transformation-based learning (TBL) does not provide tag probabilities.

(ii) If corpora is large enough, then training time in TBL is very long.

1.18 ADVANTAGES AND DISADVANTAGES OF NLP

The use of natural language processing comes with advantages as well as disadvantages.

1.18.1 Advantages of NLP

(i) Once implemented, NLP is less expensive and more time efficient than employing a person.

(ii) NLP can also help businesses. It offers faster customer service response times. Customers can receive immediate answers to their questions.

(iii) Pre-trained learning models are available for developers to facilitate different applications of NLP ; It makes them easy to implement.

(iv) Natural Language Processing is the practice of teaching machines to understand and interpret conversational inputs from humans.

(v) NLP can be used to establish communication channels between humans and machines.

(vi) The different implantations of NLP can help businesses and individuals save time, improve efficiency and increase customer satisfaction.

1.18.2 Disadvantages of NLP

(i) Training can be time-consuming. If a new model needs to be developed without the use model needs to be developed without the use of a pre-trained model, it can take weeks before achieving a high level of performance.

(ii) There is always a possibility of errors in predictions and results that need to be taken into account.

(iii) NLP may not show context.

(iv) NLP may require more keystrokes.

(v) NLP is unable to the new domain, and it has a limited function. That is why NLP is built for a single and specific takes only.

1.19 SELF LEARNING TOPICS

Types of for regional language

Various types of tools in Indians regional language are:

(i) Using the phonetic keyboard
(ii) Fonts Download
(iii) Padma Plugin

(i) Using the Phonetic Keyboard

- Using Indian languages on computer are very attractive for a layman.
- Qulllpad and lipikaar is a free online typing tool in Indian languages. It supports transit eration technologies according to pre-defined rules.
- A transit eration technology is one that allows user to type words as, they would usually do (like 'rashtrabhasha' instead 'RASHTRASHA) such as case sensitive typing rules.
- Transit eration tools expect users to type English words phonetically. This allows users to communicate in their own regional language of their choice.

(ii) Fonts Download

- Technology development for Indian language (TDIL) programme initiated by the department of electronic and IT (DEIT), govt. of Indian has the objective to develop information processing tools to facilitate human machine interaction in Indian language and to develop technologies to access multilingual knowledge resources.
- The fonts are being made available free for public through language CDS and web downloads for the benefit of massen.

(iii) Padma Plugin

- Padma is a technology for transforming Indic text between public and proprietory formate. The technology currently supports Telusu, Malyalam Tamil, Devenagri (including Marathi), Gujarathi, Bengali and Gurmukhi.
- Padma's goal is to bridge the gap between closed and open standard until the day Unicode support is widely available on all platforms.
- Padma transforms Indic text encoded in proprietary formats automatically Unicode.

Regional languages pre-processing and other functions

- If there is a nation where old and morphologically rich varieties of regional languages exist then it is India.

- It is comparatively easy for computers to process the data represented in English language through standard ASCII codes than other natural languages. But building the machine capability of understudying other natural languages is arduous and is carried out using various techniques.
- Nowadays the mternet is no more monolingual contents of the other regional languages are growing rapidly. According to 2001 census there the are approximately 1000 documented languages and dialects in India.
- Much research is being carried out to facilitate users to work and interact with computers in their own regional natural languages.
- Google offers 13 languages and provide the data of translation in Indian regional languages (IRL) like Kannada, Hindi, Bengali, Tamil, Telugu, Malyalm, Marathi, Punjabi, and Gujarathi.
- The major concentrated tasks on IRL are machine translation (MT), sentiment analysis (SA), Parts – Of – Speech (POST) Tagging and Named Entity Recognition (NEER).
- Machine translation is inter-lingual communication where machine translate source language to the target language by preserving its meaning.
- Sentiment analysis is identification of opinions expressed and orientation of thoughts in a piece of text.
- POS tagging is a process in which each word in a sentence is labelled with a tag indicating its appropriate part of speech.
- Named Entity Recognition identifies the proper names in the structured or unstructured documents and then classifies the names into sets of categories of interest.
- Machine-learning algorithms and natural language processing techniques are widely and deeply investigated for English.
- But not much work has been reported for IRL due to the richness in morphology and complexity in structure.

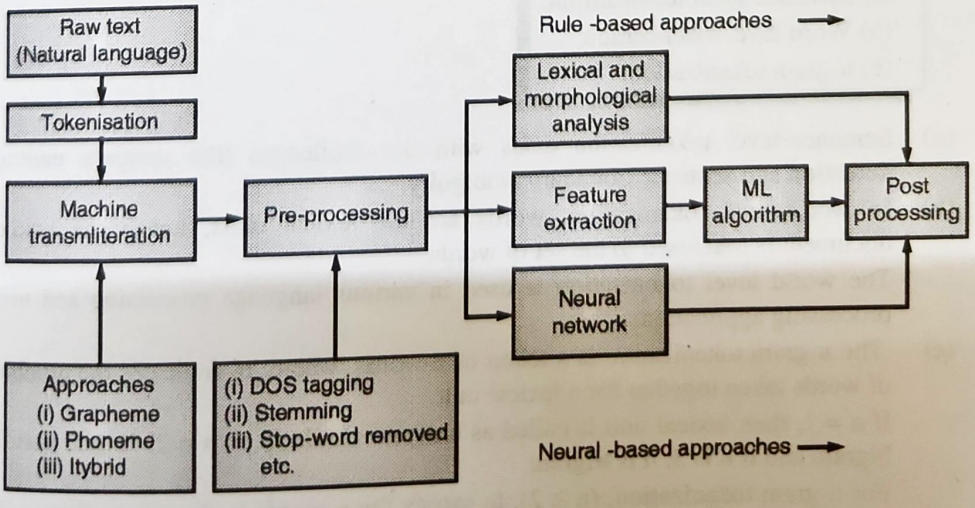


Fig. 1.19.1

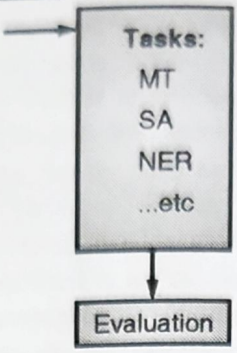


Fig. 1.19.2

☞ Generic Model for Language Processing

- The generic model for language processing consists of various stages VIZ ; machine transliteration, pre-processing, lexical and morphological analysis, POS tagging, feature extraction and evaluation.
- The contributions of techniques for success of the language processing tasks are as follows :

☞ Tokenization

- In natural language processing applications, the raw text initially undergoes a process called tokenization.
- In this process, the given text is tokenized into the lexical units, and these are basic units.
- After tokenization, each lexical unit is termed as token. Tokenization can be at sentence level or word level, depending upon the category of the problem.
- Hence, there are 3 kinds of tokenization :

(a) Sentence level tokenization.
(b) Word level tokenization,
(c) n-gram tokenization.

(a) Sentence-level tokenization deals with the challenges like sentence ending detection and sentence boundary ambiguity.

(b) In world-level tokenization, words are the lexical units, hence the whole document is tokenised to the set of words.

The world level tokenization is used in various language processing and text processing applications.

(c) The n-gram tokenization is a token of n-words, where 'n' indicates the number of words taken together for a lexical unit.

If n = 1, then lexical unit is called as unigram similarly if 'n = 2' lexical unit is bigram and if n is 3, it is trigram.

For n-gram tokenization, ($n \geq 2$), to satisfy the n-words in the tokens there will be overlapping of terms in the token.

☞ Machine transliteration

- In natural language processing, machine transliteration plays a vital role in applications like cross-language machine translation, named entity recognition, information retrieval etc.
- Transliteration is a process of converting a word or character from the source languages, alphabetical system to the target languages, alphabetical system without losing the phonetics of the source languages word or character.
- Before transliteration, words are divided into syllabic units using Unicode and character encoding standards. Then each of the syllabic units of a word gets converted to target language.

☞ For example

Hindi	English
/ ∨ /	/ a /
/ ∨« /	/'a' or 'a' /

Chapter Ends...

❑❑❑

UNIT II

CHAPTER 2

Language Syntax & Semantics

Syllabus

Morphological Analysis : What is Morphology? Types of Morphemes, Inflectional morphology & Derivational morphology, Morphological parsing with Finite State Transducers (FST)

Syntactic Analysis : Syntactic Representations of Natural Language, Parsing Algorithms, Probabilistic context-free grammars, and Statistical parsing

Semantic Analysis : Lexical Semantic, Relations among lexemes & their senses –Homonymy, Polysemy, Synonymy, Hyponymy, WordNet, Word Sense Disambiguation (WSD), Dictionary based approach, Latent Semantic Analysis.

2.1 ENGLISH MORPHOLOGY

For each **token** in the text, the Natural Language Provides information about its **internal structure** (**morphology**) and its **role** in the **sentence** (**syntax**).

GQ. Explain English Morphology.

- Morphology it the study of the internal structure of words. Morphology focuses on how the components within a word (stems, root words, prefixes, suffixes etc.) are arranged or modified to create different meanings.
- English often adds 's' or "es" to the end of count nouns to indicate pluralities and 'd or 'ed' to a verb to indicate past tense. The suffix "– ly" is added to adjectives to create adverbs (for example, "happy" (adjective) and "happily" (adverb)).
- The natural Language API uses morphological analysis to inter grammatical information about words.
- Morphology varies greatly between languages. Language such as English lacks affixes indicating case, rely more on the word order in a sentence to indicate the respective roles of words.

- Hence morphological analysis depends heavily on the source language, and an understanding of what is supported within that language.
- In English there are numerous examples, such as "replacement", which is composed of re-"place", and -ment, and "walked", from the elements "walk" and –ed.
- English morphology supports language elements (grammar, vocabulary) and language skills (reading, writing, speaking)

2.1.1 Survey of English Morphology

1. Morphology is the study of the way the words are built up from smaller meaning-bearing units called morphemes. Morphemes are defined as the minimum meaning-bearing units in a language. cats' conversion rule?
2. Morphological parsing is required for such a task.◊Previously we learned how to accommodate both 'cat' and its plural form 'cats' in a regexp. But, how can we represent words such as 'geese', 'foxes' etc. which are also plural forms but do not follow the 'cat
3. The words such as 'foxes' are broken down into a stem and an affix. The stem is the root word while the affix is the extension added to the stem to represent either a different form of the same class or a whole new class.
4. In English, morphology can be broadly classified into two types :
   (a) Inflectional morphology : It is the combination of a word stem with a grammatical morpheme which results in a word of the same class as the original stem. In English infliction is simple, only nouns, verbs, and sometimes adjectives can be inflicted. Eg. cat → cats, mous → mice, walk → walking, etc.
   (b) Derivational morphology : It is the combination of a word stem with a grammatical morpheme which results in a word of a different class. In english it is very hard to predict the meaning of the stem from the derived structure. Eg. appoint → appointee, clue → clueless, kill → killer etc.

2.2 MORPHEME

- A 'morpheme' is the smallest 'lexical item' in a language. The field of linguistic study dedicated to morphemes is called morphology.
- In English, morphemes are often but 'not necessarily' words. Morphemes that stand alone are considered as 'roots' (such as the morpheme cat); other morphemes, called affixes, are found only in combination with other morphemes.
- This distinction is not universal and does not apply to, for example, 'Latin', in which many roots cannot stand alone. For example, the Latin root 'reg' ("king") must always be suffixed with a case marker : rex (reg-s), reg-is, reg-i, etc.
- For a language like Latin, a root can be defined as the main lexical morpheme of a word.

2.2.1 Classification : (Free and Bound Morphemes)

Every morpheme can be classified as free or bound.

(1) 'Free morphemes' can function independently as words (e.g. town, dog) and can appear within lexemes (e.g. townhall, doghouse).

(2) 'Bound morphemes' appear only as parts of words, always in conjunction with a 'root' and sometimes with other bound morphemes. For example, 'un–' appears only when accompanied by other morphemes to form a word.

2.2.2 Kinds a Morphology

1. **Inflectional :** Regular, applies to every noun, verb, whatever or at least the majority of them. E.G. all count nouns have singular/plural distinction, all verbs have tense distinctions, etc. Tend to be very **Productive**, i.e. are found throughout the language; every (count) noun can be pluralized, every verb has a past tense, etc.
2. **Derivational :** Morphemes usually change "form class" ("part of speech"), e.g. makes a verb out of a noun, or an adjective out of a verb, etc. Not always very regular, *not very productive* But useful in various ways, especially in the formation of *abstract* nouns, esp. in development of scientific *registers*.
   - Example : Photograph (n.) → photograph-y (another kind of N.)
   - clear (adj.) + ance, +ity, +ness : clearance, clarity, clearness: 3 different kinds of N's
   - -ness, -hood, -ize, -dom, -ling. Likeness, likelihood (but not *like hood, *likeliness); kingdom, princeling (but not *king ling, princedom). -ize is very productive: can be added to many form classes to make verbs: potentialize, manhattanize, losangelize, maximize, miniaturize, etc.
   - nation (n.) + al (adj.) → `national' + ize→ (makes a verb) `nationalize' + ation → `nationalization' (back to a noun) ``process of making s.t. belong to the nation") + de- → `denationalization' ``reversing the process of making s.t. belong to the nation"

2.2.3 Inflectional Morphology

It is one of the ways to combine morphemes with stems.

(1) Inflectional morphology conveys grammatical information, such as number, tense, agreement or case.

(2) Due to inflectional morphological process, the meaning and categories of the new inflected words usually do not change.

That is a noun can be inflected to a noun while adding affixes, a verb can be inflected to a verb in different tense in English.

(3) One can say that the root word (stem) is **inflected** to form other words of same meaning and category.

(4) Inflection creates different forms of the same word.

(5) In English, only nouns and verbs can be inflected (sometimes adjectives also). Inflectional morphemes are very less when compared with some other languages.

Example

Category	Stem	Affixes	Inflected Word
Noun	Word box	– s	Words
		– es	Boxes
Verb	Treat	– s	Treats
		–ing	Treating
		– ed	Treated

In the above example,

(i) The Inflectional morpheme 's' is combined with the noun stem 'word' to create plural noun 'words'.

(ii) The inflectional morpheme '–ing' is combined with the verb stem 'treat' to create a gerund 'treating' .

(6) Inflectional morphemes do not change the essential meaning or the grammatical category of a word.

Adjectives stay adjectives, nouns remain nouns, and verbs remain verbs.

For example, if we add an '– s' to the noun carrot to show plurality, carrot remains a noun.

(7) Examples of Inflectional Morphemes

Inflectional morphemes are suffixes that get added to a word, thus, adding a grammatical value to it. It can assign a tense, a number, a comparison, or a possession. Here are some examples of inflectional morphemes.

- **Plural :** Bikes, Cars, Trucks, Lions, Monkeys, Buses, Matches, Classes
- **Possessive :** Boy's, Girl's, Man's, Mark's, Robert's, Samantha's, Teacher's, Officer's
- **Tense :** cooked, played, marked, waited, watched, roasted, grilled; sang, drank, drove
- **Comparison :** Faster, Slower, Quicker, Taller, Higher, Shorter, Smaller, Weaker, Stronger, Sharper, Bigger
- **Superlative :** Fastest, Slowest, Quickest, Tallest, Highest, Shortest, Smallest, Biggest, Weakest, Strongest, Sharpest

2.2.4 Types of Morphology

(1)	– S	3rd person singular present	She waits
(2)	– en	Past participle	She has eaten
(3)	– S	Plural	Three tables
(4)	's	Possessive	Holly's cat
(5)	– er	Comparative	You are taller

2.2.5 Derivational Morphology

Derivation is the process of creating new words from a steam/base form of a word.

(1) One of the most common ways to derive new words is to combine derivational affixes with root words (stems).

The new words formed through derivational morphology may be a stem for another affix.

(2) New words are **derived** from the root words in this type of morphology.

(3) English derivation is one of the complex derivations. It is because of one or more of the following reasons

(a) Less productive. That is, a morpheme added with a sets of verbs to make new meaningful words cannot always be added with all verbs.

For example, the base word 'summarise' can be added with the grammatical morpheme 'ation' results in a word 'summarisation', but this morpheme cannot be added with all the verbs to make similar effects.

(4) Complex meaning differences among nominalising suffixes.

For example, the words 'conformation' and 'conformity' both derived from the word stem 'conform' but meanings are completely different.

Derivation creates different words from the same lemma :

Example

Category	Stem	Affixes	Derived word	Target category
Noun	Vapour	–ize	Vaporize	Verb
Verb	Read	–er	reader	Noun
Adjective	Real	–ize	Realize	Verb
Noun	Mouth	–ful	Mouthful	Adjective

(1) Some more examples of words which are built up from smaller parts :

Black + bird combine to form black bird, dis + connect combine to form disconnect.

(2) Some more examples of English derivational patterns and their suffixes :

(a) adjective – to – noun, – ness (slow → slowness)

(b) adjective – to – verb, – en (weak – weaken)

(c) adjective – to – adjective –ly (personal - personally)

(d) nown – to – adjective – al (recreation- recreational)

2.2.6 Comparison between Derivation and Inflection

	Derivation	Inflection
(1)	Derivation may be effected by formal means like affixation, reduplication, internal modification of bases, and other morphological processes.	Inflection also may be effected by the formal means like affixation, ...etc.
(2)	Derivation serves to create new lexemes.	Inflection prototypically serves to modify lexemes to fit different grammatical contexts.
(3)	Derivation changes category, for example taking a verb like employ and making it a noun (employment, employer, employee) or an adjective (employable), or taking a noun like union and making it a verb (unionise) or an adjective (unionish, unionesque).	Inflection typically adds grammatical information about number (singular, dual, plural), person (first, second, third), tense (past, future), aspect (perfective, imperfective, habitual) and case (nominative, accusative), among other grammatical categories that languages might mark.
(4)	Also, we note that derivation need not change category. For example, the creation of abstract nouns from concrete ones in English (king-kingdom, child-childhood) is a matter of derivation Derivational prefixes in English tends not to change category, but it does add substantial new meaning, for example creating negatives (unhappy, inconsequential).	

Remark

There are instances that are difficult to categorise, or that seem to fall somewhere between derivation and inflection.

2.3 MORPHOLOGICAL PARSING WITH FST (FINITE STATE TRANSDUCER)

- Morphological parsing is the process of determining the **morphemes** from which a given word is constructed.
- If must be able to distinguish between orthographic rules and morphological rules. For example, the word 'foxes' can be decomposed into 'fox' (the stem), and 'es' (a suffix indicating plurality).

- The generally accepted approach to morphological parsing is through the use of a **finite state transducer (FST)**. It inputs words and outputs their stem and modifiers.
- The FST is actually created through algorithmic parsing of some word source, such as a dictionary, complete with modifier markups.
- Another approach is through the use of an **indexed lookup method**. It uses a constructed radix tree. This is not an often taken route because it breaks down for morphologically complex languages.
- With the advancement of neural networks in natural language processing, it is less common to use FST for morphological analysis. For languages for which there is a lot of available training data, FST is less is use.

2.3.1 Orthographic

- Orthographic rules are general rules. They are used when breaking a word into its stem and modifiers.
- Consider an example : singular English words ending with – y, when it is pluralised, it ends with – ies.
- Morphological rules which contain corner cases to these general rules.
- Both these rules are used to construct systems that can do morphological parsing.

2.3.2 Morphological

- Morphological rules are exceptions to the orthographic rules. It is when breaking a word into its stem and modifiers.
- Various models of natural morphological processing are proposed. Generally monolingual speakers process words as whole, while bilingual break words into their corresponding morphemes. It is because their lexical representations are not as specific, and also lexical processing in the second language may be less frequent than processing the mother tongue.
- Applications of morphological processing include :
  (a) Machine translation, (b) Spell checker, and (c) Information retrieval

Importance of morphology in NLD

- Morphological analysis is a field of linguistics that studies the structure of words.
- It identifies how a word is produced through the use of morphemes.
- A morpheme is a basic unit of the English language. The morpheme is the smallest element of a word that has grammatical functioning and meaning.
- In most of the applications related to the Natural Language processing, findings of the morphological Analysis and Morphological generation are very important.

☞ Application text to speech synthesis

- Various mediums such as computers, mobiles, are used for fulfilment of daily need. But disabled people and the people who are not much acquainted with such technical medias. They face log of difficulties.
- There arises the need of Text to speech synthesis. Morphological analysis is used to reduce the size of lexicon. It is because here one needs to remember the root word and various inflections need not be remembered.
- Morphological Analysis can be used to segregate a compound word into basic form.
- The applications are not limited to a particular language but can be included upto Hindi, Arabic, Marathi depending upon the particular field.

⏭ 2.4 LEXICON (FREE FST PORTER STEMMER ALGORITHM)

- Lexicon refers to the component of a NLP system that contains information (semantic, grammatical) about individual words or word strings.

☞ Example of lexicon

- An example of lexicon is YourDictionary.Com. An example of lexicon is a set of medical terms.

☞ Lexicon in language learning

- A lexicon is often used to describe the knowledge that a speaker has about the words of a language. This includes meanings, use, form, and relationships with other words.
- A lexicon can thus be thought of as a mental dictionary.

☞ Lexicon in NLTK

Lexicon is a vocabulary, a list of words, a dictionary. In NLTK, any lexicon is considered a corpus since a list of words is also a body of text.

☞ Importance of lexicon

Different language research suggest that the lexicon is representationally rich, that it is the source of much productive behaviour. Its lexically specific information plays a critical and early role in the interpretation of grammatical structure.

☞ Lexicon in Communication

- A lexicon is the collection of words :
- Or the internalised dictionary – that every speaker of a language has. It is also called lexis. Lexicon also refers to a stock of terms used in a particular profession subject or style.

Lexicon in AI

- In its simplest form, a lexicon is the vocabulary of a person, language, or branch of knowledge. It is the catalog of words used often in conjunction with grammar, the set of rules for the use of these words.

2.4.1 Porter Stemming Algorithm

- The porter stemming algorithm (or 'porter stemmer') is a process for removing the commoner morphological and inflexional endings from words in English.
- Its main use is as part of a term normalisation process that is generally done while setting up information Retrieval systems.

Porter stemmer with example

Words such as "Likes", "liked", "likely" and "liking" will be reduced to "like" after stemming.

In 1980, porter presented a simple algorithm for stemming English language words.

Porter stemmer has two major achievements :

(i) The rules associated with suffix removal are much less complex in case of porter stemmer.

(ii) The second difference is that the porter's stemmer uses a single unified approach to the handling of context.

Use of porter stemmer : (Applications) :

(i) The main applications of porter stemmer include data mining and information retrieval. But its applications are limited only to English words.

(ii) The group of stems is mapped on to the same stem and the output stem is not necessarily a meaningful word.

Implementation of porter stemmer

We follow the following steps :

- **Step (1) :** Import the NLTK library and from NLTK import porterstemmer, import nltk from nltk, stem import porterstemmer.
- **Step (2) :** Create a variable and store porterstemmer into it, (PS = Porter Stemmer)
- **Step (3) :** See how to use porter stemmer print (ps.stem ('bat')) print (ps.stem ('batting'))

Porter stemmer in NLP

- The porter stemming algorithm (or 'porter stemmer') is a process for removing the common morphological and inflexional endings from words in English. Its main use is as part of a term normalization process that is usually done when setting up information retrieval systems.

2.4.2 Difference between Stemming and Lemmatization

	Stemming	Lemmatization
(i)	Stemming is a process that stems or removes last few characters from a word, often leading to incorrect meanings and spellings	Lemmatization considers the context and converts the word to its meaningful base form, which is called Lemma.
(ii)	For example, stemming the word 'Caring' would return 'car'.	Lemmatizing the word 'caring' would return 'care'.
(iii)	Stemming is used in case of large dataset where performance is an issue.	Lemmatization is computationally expensive since it involves look-up tables

2.5 SYNTACTIC REPRESENTATION OF NLP

- Syntactic analysis is the third phase of Natural language processing (NLP). It is used to analyse syntax, sometimes called as syntax or parsing analysis.
- Syntax analysis compares the text to formal grammar rules to determine its meaning. The statement "heated ice-cream," for example, would be discarded by a semantic analyses.

2.5.1 The Parser Concept

- It is used to carry out the parsing process. It is a software component that takes input data (text) and converts it into a structural representation after verifying it for valid syntax using formal grammar.
- It creates a data structure, which can be a parse tree, an abstract syntax tree, or another hierarchical structure.

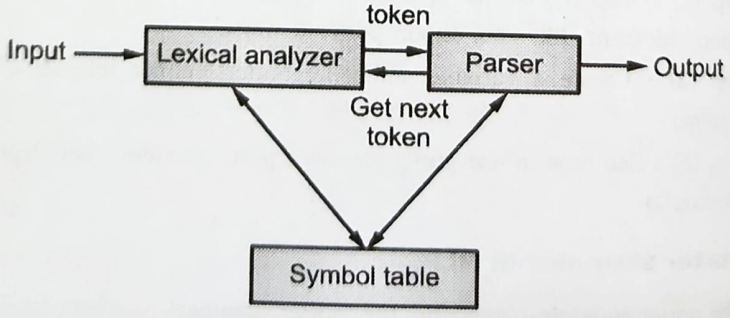


Fig. 2.5.1

The primary functions of parse include :

(i) To report any errors in syntax.

(ii) To recover from a frequently recurring error so that the rest of the program may be processed.
(iii) To make a parse tree.
(iv) To make a symbol table.
(v) Creating intermediate representations (IR).

2.6 PARSERS AND ITS RELEVANCE IN NLP

- The word 'Parsing' is used to draw exact meaning or dictionary meaning from the text. It is also called syntactic analysis or syntax analysis.
- Syntax analysis checks the text for meaningfulness. The sentence like "Give me hot ice-cream," will be rejected by the parser or syntactic analyser.
- In this sense, we can define parsing or syntactic analysis or syntax analysis as follows :
- It is defined as the process of analyzing the strings of symbols in natural language conforming to the rules of formal grammar.

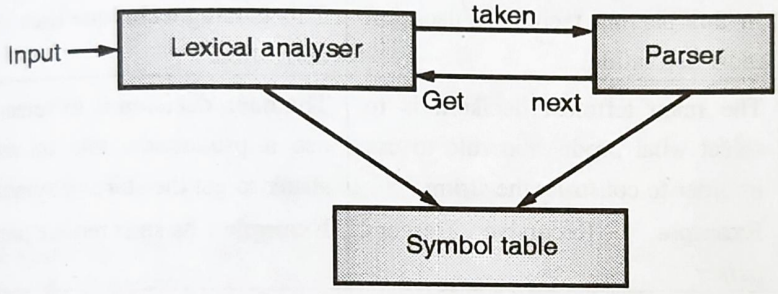


Fig. 2.6.1

- We can understand the relevance of parsing in NLP with the help of following points :
  (i) Parser is used to report any syntax error.
  (ii) It helps to recover from commonly occurring error so that the processing of the remainder of program can be continued.
  (iii) Parse-tree is created with the help of a parser.
  (iv) Parser is used to create symbol table, and that plays an important role in NLP.
  (v) Parser is also used to produce intermediate representations (IR).
- **Remark :** The word 'Parsing' whose origin is from Latin word 'pars' and it means '**part**'.

2.6.1 Parsing Top Down and Bottom Up

- There are 2 types of **parsing techniques**, the first one is **Top down parsing** and the second one is **Bottom up parsing**.

- **Top down parsing** is a parsing technique that first looks at the highest level of the parse tree and works down the parse tree by using the rules of grammar.
- And **Bottom-up parsing** is a parsing technique that first looks at the lowest level of the parse tree and works up the parse tree by using the rules of grammar.
- We mention below the differences between these two parsing techniques.

Sr. No.	Top-Down Parsing	Bottom-up Parsing
1.	It is a parsing strategy that first looks at the highest level of parse tree and works down the parse tree by using the rules of grammar.	It is a parsing strategy that first looks at the lowest level of the parse tree and works up the parse tree by using the rules of grammar.
2.	Top-down parsing attempts to find the left most derivations for an input string.	Bottom up parsing can be defined as an attempt to reduce the input string to the start symbol of a grammar.
3.	In this parsing technique uses left most derivation.	This parsing technique uses right most derivation.
4.	The main leftmost decision is to select what production rule to use in order to construct the string. **Example** : Recursive Descent parser.	The main decision is to select when to use a production rule to reduce the string to get the starting symbol. **Example :** Its shift reduce parser.

2.6.2 Modelling Constituency

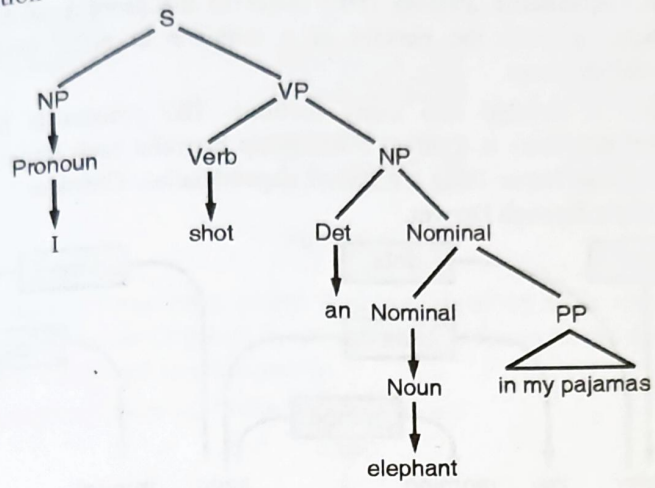
Knowledge of language is the doorway to wisdom. **- Roger Bacon**

- Roger Bacon gave the above quote in the 13th century, and it still holds.
- Today, the way of understanding languages has changed completely.
- Here, we shall be covering some basic concepts of modeling constituency or constituency parsing, in natural languages.

2.6.3 Constituency Parsing

- Constituency Parsing is the process of analyzing the sentences by breaking down it into sub-phrases also known as constituents.
- These sub-phrases belong to a specific category of grammar like NP (noun phrase) and VP (verb phrase).
- Constituency parsing is based on context-free grammars. Constituency context-free grammars are used to parse text.
- The parse tree includes sentences that have been broken down into sub-phrases, each of which belongs to a different grammar class.

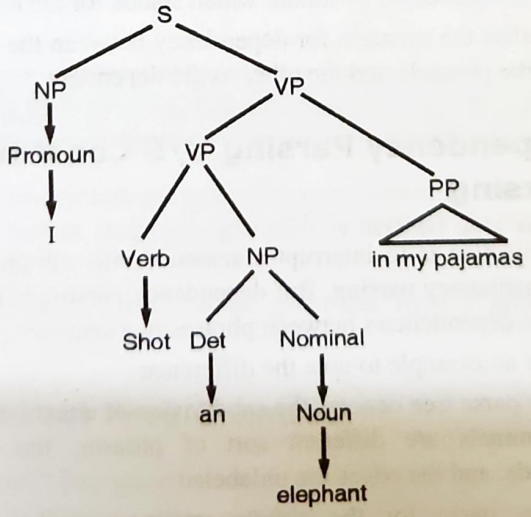
- A terminal node is a linguistic unit or phrase that has a mother or father node and a part-of-speech tag.
- As an example, "A cat" and "a box beneath the bed", are noun phrases, while "write a letter" and "drive a car" are verb phrases.
- We consider an example sentence : "I shot an element in my pajamas."
- We mention the constituency parse-tree graphically as :



(I)

Fig. 2.6.2(Contd...)

- The parse tree on the top (I) represents catching an elephant carrying pajamas, while the parse tree on the bottom (II) represents capturing an element in his pajamas.
- The entire sentence is broken down into sub-phrases till we have got terminal phrases remaining.



(II) Fig. 2.6.2

- VP stands for Verb-Phrases, and NP stands for Noun Phrases.

2.6.4 Dependency Parsing

- First we make the concept of dependency parsing clear, so that we can compare dependency parsing with constituency parsing.
- The term Dependency Parsing (DP) refers to the process of examining the dependences between the phrases of a sentence in order to determine its grammatical structure.
- A sentence is divided into many sections. The process is based on the assumption that there is a direct relationship between each linguistic unit in a sentence. These hyper links are called dependencies. Consider : "I prefer the morning flight through Denver."

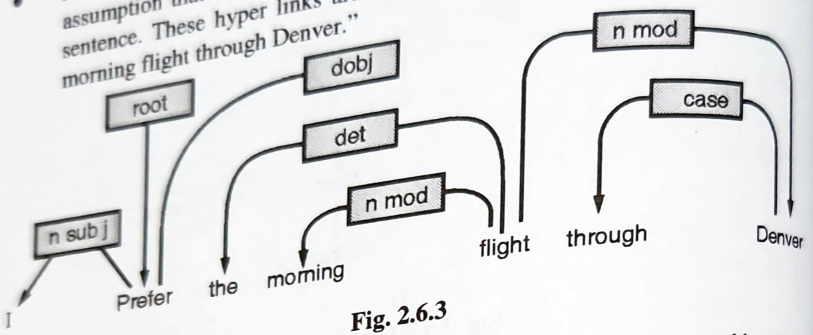


Fig. 2.6.3

- The relationships between each linguistic unit, or phrase are expressed by direct arcs. The root of the tree "prefer" varies the pinnacle of the preceding sentence.
- A dependence tag indicates the relationship between two phrases.
- For example, the word "flight" changes the meaning of the noun "Denver". : flight → Denver, where flight is the pinnacle and Denver is the kid or dependent. It is represented by nmod, which stands for the nominal modifier.
- This distinguishes the scenario for dependency between the two phrases, where one serves as the pinnacle and the other as the dependent.

2.6.5 Dependency Parsing V/S Constituency Parsing

- If the main objective is to interrupt a sentence into sub-phrases, it is ideal to implement constituency parsing. But dependency parsing is the best method for discovering the dependencies between phrases in a sentence :
- Let us consider an example to note the difference :
- A constituency parse tree denotes the subdivision of a text into sub-phrases. The tree's non-terminals are different sort of phrases, the terminals are the sentence's words, and the edges are unlabeled.
- A constituency parse for the simple statement "John sees Bill" would be !

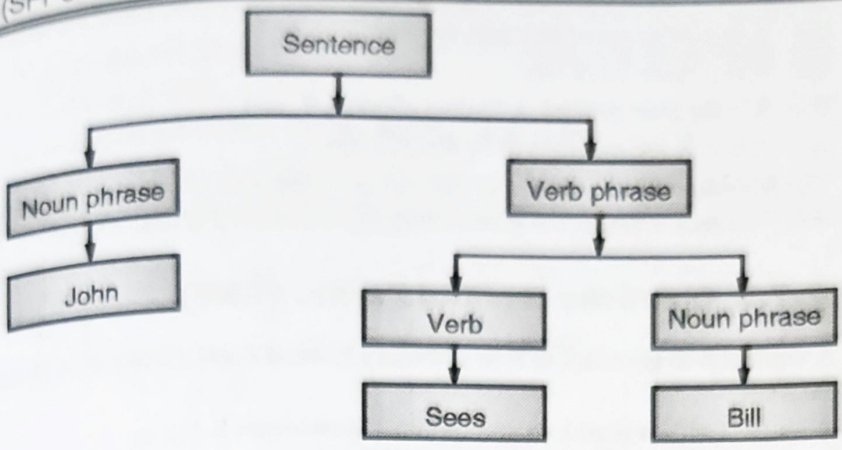


Fig. 2.6.4

- A dependency parse links words together based on their connections. Each vertex in the tree corresponds to a word, child nodes to words that are reliant on the parent, and edges to relationships.
- The dependency parse for "John sees Bill" is as :

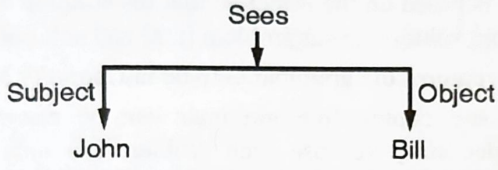


Fig. 2.6.5

- One should choose the parser type that is closely related to the objective.
- For sub-phrases inside a sentence, then constituency-parse is advisable.
- But for the connection between words, then dependency-parse is more convenient.

2.7 COCKE-YOUNGER-KASAMI (CYK) ALGORITHM

- Grammar implies syntactical rules for conversation in natural language. But in the theory of formal language, grammar is defined as a set of rules that can generate strings.
- The set of all strings that can be generated from a grammar is called the language of the grammar.

2.7.1 Context Free Grammar

- We have a context free grammar
  G = (V, X, R, S) and a string w, where :
  (i) V is a finite set of variables or non-terminal symbols.

(ii) X is a finite set of terminal symbols.

(iii) R is a finite set of rule.

(iv) S is the start symbol, a distinct element V, and

(v) V and X are assumed to be disjoint sets.

- The **Membership problem** is defined as : Grammar G generates a language L (G). To check whether the given string is a member of L (G).

2.7.2 Chomsky Normal Form : (CNF)

A context free grammar G is in Chomsky Normal Form (CNF), if each rule of G is of the form :

(i) $A \rightarrow BC$ [with at most two non-terminal symbols on R.H.S.]

(ii) $A \rightarrow a$, or [one terminal symbol on RHS]

(iii) $S \rightarrow$ nullstring [null string]

2.7.3 Cocke-Younger-Kasami Algorithm

- This solves the membership problem using a **dynamic programming** approach.
- The algorithm is based on the principle that the solution to problem [i, j] can be constructed from solution to subproblem [i, k] and solution to subproblem [k, j].
- The algorithm requires the grammar G to be in Chomsky Normal Form (CNF).
- Observe that any context-free grammar can be **converted to CNF.** This restriction is necessary because each problem can only be divided into two subproblems and not more-to bound the time complexity.

2.7.4 How CYK Algorithm Works ?

- For a string of length N, construct a table T of size $N \times N$. Each cell in the table T [i, j] is the set of all constituents that can produce the substring spanning from position i to j.
- The process involves filling the table with the solutions to be subproblems encountered in the **bottom-up** parsing process. Therefore, cells will be filled from left to right and bottom to top.

	1	2	3	4	5
1	[1, 1]	[1, 2]	[1, 3]	[1, 4]	[1, 5]
2		[2, 2]	[2, 3]	[2, 4]	[2, 5]
3			[3, 3]	[3, 4]	[3, 5]
4				[4, 4]	[4, 5]
5					[5, 5]

- In T [i, j], the row number i denotes the start index and the column number j denotes the end index.
- Let us consider the phrase, "a very heavy orange book"
- a (1) very (2) heavy (3) orange (4) book (5).
- We fill up the table from left to right and bottom to top, according to the rules as above :

	1 a	2 very	3 heavy	4 orange	5 book
1 a	Det	–	–	NP	NP
2 very		Adv	AP	Nom	Nom
3 heavy		A	AP	Nom	Nom
4 orange				Nom A, AP	Nom
5 book					Nom

2.8 PROBABILISTIC CONTEXT FREE GRAMMAR (PCFG)

- PCFGs extend **context-free grammars** similar to how hidden Markov models extend regular grammars. Each production is assigned a probability.
- The probability of a parse (derivation) is the product of the probabilities of the productions used in that derivation. These probabilities can be viewed as **parameters** of the model.

2.8.1 Some Important Definitions

(i) **Derivation :** The process of recursive generation of strings from a grammar.

(ii) **Parsing :** Finding a valid derivation using an automation.

(iii) **Parse tree :** The alignment of the grammar to a sequence.

- An example of a parser for PCFG grammars is the pushdown automation.
- The algorithm parses grammar nonterminals from left to right in a stack-like manner. This brute force is not very efficient.
- Another example of a PCFG parser is the standard statistical parser which is trained using Treebank.

2.8.2 Formal Definition of PCFG

A probabilistic context-free grammar G is defined by a quintuple :

$G = (M, T, R, S, P)$

Where

(i) M is the set of non-terminal symbols

(ii) T is the set of terminal symbols.

(iii) R is the set of production rules.

(iv) S is the start symbol,

(v) P is the set of probabilities on production rules.

2.8.3 Relation with hidden Markov Models

- PCFG model computes the total probability of all derivations that are consistent with a given sequence, based on some PCFG.
- This is equivalent to the probability of the PCFG generating the sequence. It is a measure of the consistency of the sequence with the given grammar.
- Dynamic programming variants of the CYK algorithm find the Viterbi parse of a RNA sequence for a PCFG model. The parse is the most likely derivation of the sequence by the given PCFG.

2.8.4 Viterbi PCFG Parsing

- Viterbi PCFG parser is a **bottom-up** that uses **dynamic programming** to find the **single most likely parse for a text**.
- It parses texts by iteratively filling in a most likely constituents table. This table records the most likely tree structure for each span and node value.

2.8.5 How a PCFG Differs from CFG ?

- A PCFG differs from a CFG by augmenting each rule with a conditional probability : $A \rightarrow B$ [P]. Here P expresses the probability that non-terminal A will be expanded to sequence β.
- Associate a probability with each grammar rule.

2.8.6 How PCFG Resolves Ambiguity ?

PCFG parsers resolve ambiguity by preferring constituents (and parse tree) with the highest probability.

2.8.7 How does PCFG is used ?

The PCFG is used to predict the prior probability distribution of the structure whereas posterior probabilities are estimated by the inside-outside algorithm and the most likely structure is found by the CYK algorithm.

2.8.8 What are Limitations of PCFG ?

- PCFGs do not take lexical information into account.
- It makes parse plausibility less than ideal.
- PCFGs have certain biases, i.e., the probability of a smaller tree is greater than a larger tree.

2.8.9 Are PCFGs Ambiguous ?

- Probabilities in a PCFG can be seen as a filtering mechanism.
- For an ambiguous sentence, the trees bearing maximum probability are singled out, while all others are discarded.
- The level of ambiguity is related to the size of the singled out set of trees.

2.9 SHIFT REDUCE PARSER

- Shift-Reduce parser attempts for the construction of parse in a similar manner as is done in bottom-up parsing, i.e. the parse tree is constructed from leaves (bottom) to the root (up).
- A more general form of the shift-reduce parser is the LR parser.
- This parser requires some data structures i.e.
  (i) An input buffer for storing the input string.
  (ii) A stack for storing and accessing the production rules.

2.9.1 Basic Operations

(i) **Shift :** This involves moving symbols from the input buffer onto the stack.
(ii) **Reduce :** If the handle appears on top of the stack then, its reduction by using appropriate production rule is done. It means that RHS of a production rule is popped out of a stack and LHS of a production rule is pushed onto the stack.
(iii) **Accept :** If only the start symbol is present in the stack and the input buffer is empty, then the parsing action is called accept.
When accepted action is obtained, it implies that successful parsing is done.
(iv) **Error :** This is the situation where the parser can
  (i) neither perform shift action
  (ii) nor reduce action and
  (iii) not even accept action.

2.9.2 Shift Reduce Parsing in Computer

- Shift reduce parser is a type of **Bottom-up parser**. It generates the parse Tree from leaves to the Root.
- In shift reduce parser, the input string will be reduced to the starting symbol.

- This reduction can be produced by handling the rightmost derivation in reverse, i.e. from starting symbol to the input string.

2.9.3 Why Bottom-up Parser is called Shift Reduce Parser ?

Bottom-up parsing is also called shift-and-reduce parsing where shift means read the next token, reduce means that a substring matching the right side of a production A.

2.9.4 What are the 2 Conflicts in Shift Reduce Parser ?

- In shift reduce parsing, there are two types of conflicts :
  (i) Shift-reduce (SR) conflict and
  (ii) Reduce-reduce conflict (RR)
- For example, if a programming language contains a terminal for the reserved word "while", only one entry for "while" can exist in the state.
- A shift-reduce action is caused when the system does not know if to 'shift' or 'reduce' for a given token.

2.9.5 Example

Ex. 2.9.1 : Consider the grammar

$E \rightarrow 2\,E2,\quad E \rightarrow 3\,E3,\quad E \rightarrow 4$

Perform shift-reduce parsing for input string "32423".

Soln. :

Stack	Input Buffer	Parsing Action
$	32423 $	shift
$ 3	2423 $	shift
$ 32	423 $	shift
$ 324	23 $	Reduce by $E \rightarrow 4$
$ 32 E	23 $	shift
$ 32 E2	3 $	Reduce by $E \rightarrow 2\,E2$
$ 3E	3 $	shift
$ 3 E3	$	Reduce by $E \rightarrow 3\,E3$
$ E	$	Accept

2.10 TOP DOWN PARSER : EARLY PARSER

- The Early Parser is an algorithm for parsing strings that belong to a given context-free language.
- Depending upon the variants, it may suffer problems with certain nullable grammars.
- The algorithm uses dynamic programming.
- It is mainly used for parsing in **computational linguistics**.

Earley Parser	
Class :	Passing grammars that are context-free
Data structure :	String
Worst-case performance :	$O(n^3)$
Best-case performance :	$\Omega(n)$ for all deterministic context-free grammars $\Omega(n^2)$ for unambiguous grammars
Average performance :	$\theta(n^3)$

2.10.1 Functioning of Earley Parser

- Early parsers are appealing because they can parse all context-free languages.
- The early parser executes in cubic time in the general case $O(n^3)$, where n is the length of the parsing string, quadratic time for unambiguous grammars $O(n^2)$, and linear time for all **deterministic context free grammars**.
- It performs well when the rule, are written left-recursively.

2.10.2 Earley Recogniser

- The following algorithm describes the Earley recognizer.
- The recognizer can be modified to create a parse tree as it recognizes, and in that way it can be turned into a parser.

2.10.3 The Algorithm

- Here; α, β and γ represent any string of terminals/nonterminals (including the empty string), X and Y represent single nonterminals, and a represents a terminal symbol.
- Earley's algorithm is a top-down dynamic programming algorithm.
- Here, we use Earley's dot notation : given a **production** $X \rightarrow \alpha\beta$ the notation $X \rightarrow \alpha \cdot \beta$ represents a condition in which α has already been parsed and β is expected.
- Input position O is the position prior to input. Input position n is the position after accepting the n^{th} token.

- For every input position, then parser generates a state set. Each state is a tuple $(X \rightarrow \alpha \cdot \beta)$, consisting of
  - (i) the production currently being matched $(X \rightarrow \alpha \beta)$
  - (ii) the current position in that production (represented by the dot)
  - (iii) the position i in the input at which the matching of the production began : the **origin position**.
- The state set at input position K is called S (K). The parser is seeded with S (0) consisting of only the top-level rule.
- The parser then repeatedly executes three operations : prediction, scanning and completion :
  - **(i) Prediction :** For every state in S (K) of the form $(X \rightarrow \alpha \cdot Y\beta, j)$, (where j is the origin position as above), add $(Y \rightarrow \cdot\ y, K)$, to S (K) for every production in the grammar with Y on the left-hand side $(Y \rightarrow y)$.
  - **(ii) Scanning :** If a is the next symbol in the input stream, for every state in S (K) of the form $(X \rightarrow \alpha \cdot a\beta, j)$, add $(X \rightarrow \alpha a \cdot \beta)$ to S (K + 1).
  - **(iii) Completion :** For every state in S (K) of the form $(Y \rightarrow y \cdot, j)$, find all states in S (j) of the form $(X \rightarrow \alpha \cdot Y\beta, i)$ and add $(X \rightarrow \alpha Y \cdot \beta, i)$ to S (K).
- The algorithm accepts if $(X \rightarrow Y\cdot, O)$ ends up in S (n), where $(X \rightarrow Y)$ is the top-level-rule and n is the input length, otherwise it rejects.

2.11 PREDICTIVE PARSER

- Predictive parser is another method that implements the technique of Top-down parsing without Backtracking.
- A predictive parser is an effective technique of executing recursive-descent parsing by managing the stack of activation records.

2.11.1 Predictive Parser Components

Predictive parsers has the following components :

(i) Input Buffer : The input buffer includes the string to be parsed following by an end marker $ to denote the end of the string :

a	+	b	$

Input string

Here a, +, b are terminal symbols.

(ii) Stack : It contains a combination of grammar symbols with $ on the bottom of the stack.

At the start of parsing, the stack contains the start symbol of grammar followed by $.

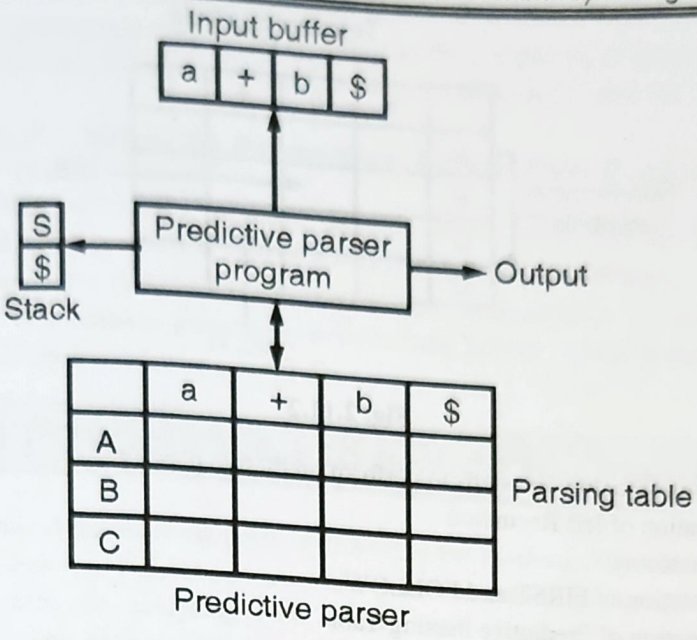


Fig. 2.11.1

(iii) **Parsing Table :** It is a two-dimensional array or Matrix M [A, a] where A is nonterminal and 'a' is a terminal symbol.

All the terminals are written column-wise, and all the non-terminals are written row-wise.

(iv) **Parsing program :** The parsing program performs some action by comparing the symbol on top of the stack and the current input symbol to be read on the input buffer.

(v) **Action :** Parsing program takes various actions depending upon the symbol on the top of the stack and the current input symbol.

2.11.2 Algorithm to Construct Predictive Parsing Table

Input : Context – free grammar G.

Output : Predictive parsing table M

Method : For the production $A \rightarrow \alpha$ of grammar G.

(i) For each terminal a in FIRST (α) add $A \rightarrow a$ to M [A, a]

(ii) If $\in$ is in FIRST (α), and b is in FOLLOW (A), then add $A \rightarrow \alpha$ to M [A, b].

(iii) If $\in$ is in FIRST (α), and $ is in FOLLOW (A), then add $A \rightarrow \alpha$ to M [A, $]

(iv) All remaining entries in Table M are errors.

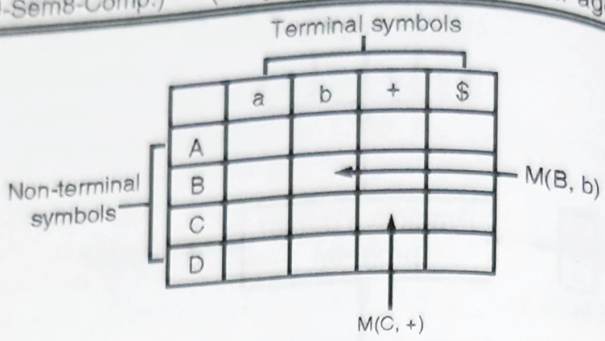


Fig. 2.11.2

We mention below the steps to perform predictive parsing :

(i) Elimination of left Recursion
(ii) Left Factoring
(iii) Computation of FIRST and FOLLOW.
(iv) Construction of Predictive Parsing Table
(v) Parse the Input string.

2.11.3 What are the Steps for Predictive Parsing ?

Preprocessing steps for predictive parsing are :

(i) Removing the left recursion from the grammar.
(ii) Performing left factoring on the resultant grammar.
(iii) Removing ambiguity from the grammar.

2.11.4 Is Predictive Parser Top-down ?

- A predictive parser is a recursive descent parser with no backtracking or backup.
- It is a top-down parser that does not require backtracking.
- At each step, the choice of the rule to be expanded is made upon the next terminal symbol.

2.11.5 The Difference between Predictive Parser and Recursive Descent Parser

The main difference between recursive descent parser and predictive parser is that recursive descent parsing may or may not require backtracking while predictive parsing does not require any backtracking.

2.11.6 Drawbacks of Predictive Parsing

Drawbacks or disadvantages of predictive parser are :

(i) It is inherently a recursive parser, so it consumes a lot of memory as the stack grows.

(ii) Doing optimisation may not be as simple as the complexity of grammar grows.
(iii) To remove this recursion, we use LL-parser, which uses a table for lookup.

2.11.7 What is Recursive Predictive Parsing ?

- Predictive parser is a recursive descent parser, which has the capability to predict which production is to be used to replace the input string.
- The predictive parser does not suffer from backtracking. To accomplish its tasks, the predictive parser uses a look-ahead pointer, which points to the next input symbols.

2.12 INTRODUCTION TO SEMANTIC ANALYSIS

- Semantic Analysis is the process of finding the meaning from text. It can direct computers to understand and interpret sentences, paragraphs, or whole documents, by analysing their grammatical structure, and identifying the relationships between individual words of the sentence in a particular context.
- Thus the aim of semantic analysis is to draw exact meaning or dictionary meaning from the text.
- The purpose of a semantic analyser is to check the text for meaningfulness.
- The most important task of semantic analysis is to get the proper meaning of the sentence. For example, analyse the sentence "Govind is great". In this sentence, the speaker is talking about Lord Govind or about a person whose name is Govind.

2.12.1 Use of Semantic Analysis

GQ. Where is semantic analysis used ?

- Semantic analysis is used in extracting important information from achieving human level accuracy from the computers.
- It is used in tools like machine translations, chatbots, search engines and text analysis.

2.12.2 Syntactic and Semantic Analysis

GQ. What is syntactic and semantic analysis ?

- Theoretically, syntactic analysis determines and checks whether the instance of the language is 'well formed' and analyses its grammatical structure.
- Semantic analysis analyses its meaning and finds out whether it 'makes sense'.
- Syntactic analysis depends on the types of words, but not on their meaning.

2.12.3 Semantic Analysis in Natural Language Processing

- Semantic analysis is a **subfield of NLP** and Machine Learning. It tries to clear the context of any text and makes one realise the emotions inherent in the sentence.
- This helps in extracting important information from achieving human level accuracy from the computers.

2.12.4 Steps to be Carried in Syntactic Analysis

(1) **Segmentation I :** Identify clause boundaries and word boundaries.
(2) **Classification I :** Determine parts of speech.
(3) **Segmentation II :** Identify constituents.
(4) **Classification II :** Determine the syntactic categories for the constituents.
(5) Determine the grammatical functions of the constituents.

2.13 MEANING REPRESENTATION

Representation of the meaning of a sentence is created by semantic analysis.

To understand the concept and approaches related to meaning representation, we first make the idea of 'building blocks' of semantic system.

2.13.1 Building Blocks of Semantic System

In representation of the meaning of words, the following blocks are used :

(i) **Entities :** It represents the individual, e.g. particular person, location, etc. For example, Haryana, Kejariwal, Pune are all entities.
(ii) **Concepts :** This represents the general category of the individuals such as a person, nation etc.
(iii) **Relations :** Here relation between entities and concept is represented.
For example, Lata Mangeshkar was a singer.
(iv) **Predicates :** It represents the verb structures.
For example, case grammar and semantic roles are the examples of predicates.

Now, it is clear, how the meaning representation combines together the building blocks of semantic systems.

It puts together entities, concepts, relation and predicates to describe a situation. It enables the reasoning about the semantic world.

2.13.2 Approaches to Meaning Representations

Approaches used by semantic analysis for the representation of meaning :

(i) First order predicate logic (FOPL)
(ii) Semantic Nets

(iii) Frames
(iv) Conceptual dependency (CD)
(v) Rule – based architecture
(vi) Case Grammar
(vii) Conceptual Graphs

2.13.3 Need of Meaning Representations

We mention below the reasons to show the need of meaning representation.

(i) Linking of linguistic elements to non-linguistic elements

Linking of linguistic elements to the non-linguistic elements can be done using meaning representation.

(ii) Representing Variety at Lexical Level

Using meaning representation, unambiguous canonical forms can be represented at the lexical level.

(iii) It can be used for Reasoning

Using meaning representation, one can reason out by verifying the 'truth' in the world and also infer the knowledge from the semantic representation.

2.14 LEXICAL SEMANTICS

Lexical semantics is a part of semantic analysis. It studies the meaning of individual words. That includes words, subwords, affixes (sub - units), compound words and phrases.

All the words, sub – words etc. are collectively called lexical items.

Thus lexical semantics is the relationship between lexical items, meaning of sentences and syntax of sentences.

The steps involved in lexical semantics are as follows :

(i) Classification of lexical items like words, sub-words, affixes etc. is preformed in lexical semantics.
(ii) Decomposition of lexical items like words, sub-words, affixes, etc. is preformed in lexical semantics.
(iii) Analyse the differences and similarities between various lexical semantic structures.

2.15 LEXICAL CHARACTERISTICS

Lexical characteristics, such as lexical approach. It is a way of analysing and teaching language based on the idea that it is made up of lexical units rather than grammatical structures. The units are words and fixed phrases.

2.15.1 Advantages of Lexical Approach

The great advantage of the lexical approach is that it is consciousness raising. It encourages the process of noticing of the lexical items. And this is the fundamental and preliminary step when dealing with new vocabulary.

2.15.2 Main Features of Lexical Unit

- The lexical unit can be (i) a single word, (ii) the na – bitual co – occurrence of two words.
- Second and third notion refers to the definition of a collocation or a multi – word unit.
- It is common to consider a single word as a lexical unit.

2.15.3 Limitation of the Lexical Approach

GQ. What is a limitation of the Lexical Approach ?

- While the lexical approach can be a quick way for students to pick up phrases, it does not produce much creativity.
- It can have the negative side effect of limiting people's responses to safe fixed phrases. Since they don't have to build responses, they don't need to learn the intricacies of language.

2.15.4 Principle of Lexical Approach

GQ. What is the principle of Lexical Approach ?

- The basic principle of lexical approach is "Language is grammaticalised lexis, not lexicalised grammar".
- In other words, lexis is central in creating meanings, grammar plays a subsidiary managerial role.

2.16 CORPUS STUDY

- Corpus study is corpus linguistics and is rapidly growing methodology that uses the statistical analysis of large collections of written or spoken data to investigate linguistic phenomena.
- Corpus linguistics is the language that is expressed in its text corpus, its body of "real world" text.
- Corpus study maintains that a reliable analysis of a language is more practicable with corpora, that is collected in the natural context of that language.
- The text – corpus method uses the body of texts written in any natural language. It derives the set of abstract rules which govern that language.

- These results can be used to find the relationships between the subject language and these other languages which have been undergone a similar analysis.
- Corpora have not only be used for linguistics research, they have also been used to form dictionaries (e.g. The American Heritage Dictionary of the English Language in 1969), and grammar guides, such as A Comprehensive Grammar of the English Language, Published in 1985.

2.16.1 Methods of Corpus Study

- Corpus study has generated a number of research methods, and they try to trace a path from data to Theory.
- Wallis and Nelson introduced the 3A perspective. They are: Annotation, Abstraction and Analysis.

(i) Annotation

- Annotation consists of the applications of a scheme to texts.
- Annotation may include structural make-up, **part-of-speech** tagging, parsing, and numerous other representation.

(ii) Abstraction

- It consists of translation of terms in the scheme to terms in a theoretically motivated model or dataset.
- Abstraction typically includes linguist-directed search but may include e.g., rule-learning for parsers.

(iii) Analysis

- Analysis consists of statistically probing, manipulating and generalising from the dataset. Analysis may include statistical evaluations, optimisation of the rule – bases or knowledge discovery methods. Most lexical corpora today are part - of – speech – tagged (POS - tagged).
- But even corpus linguists who work with 'unannotated plain text' also apply some method to isolate salient terms.
- In this situation, annotation and abstractions are combined in a lexical search.
- The main advantage of publishing an annotated corpus is that other users can perform experiments on the corpus.
- Linguists with differing perspectives and other interests than the originator's can exploit this work.
- By sharing data, corpus linguists are able to treat the corpus as a locus of linquistic debate and further study.

2.16.2 Corpus Approach

GQ. What is a corpus Approach ?

- The corpus Approach utilizes a large and principled collection of naturally occurring texts as the basis for analysis.
- The characteristic of the corpus approach refers to the corpus itself.
- One may work, with a written corpus, a spoken corpus, an academic spoken corpus, etc.

2.16.3 Corpus Linguistic Techniques

GQ. What are corpus linguistic techniques ?

- In corpus linguistics, common analytical techniques are dispersion, frequency, clusters, keywords, concordance and collocation.
- This part mentions how these techniques can contribute to uncovering discourse practices.

2.16.4 Corpus Example

GQ. What is a corpus example ?

- An example of a general corpus is the British National Corpus. Some corpora contain texts that are chosen from a particular variety of a language.
- For example, from a particular dialector from a particular subject area.
- These corpora are sometimes called 'sublanguage corpora'.

2.17 LANGUAGE DICTIONARY LIKE WORLDNET

- A dictionary is a listing of lexemes from the lexicon of one or more specific languages. They are arranged **alphabetically.** For ideographic languages by **radical and stroke.**
- They include information on definitions, usage, etymologies, pronunciations, translation etc.
- It is a lexicographical reference that shows interrelationships among the data.
- A clear distinction is made between general and **specialized dictionaries.** Specialized dictionaries include words in specialized fields, rather than a complete range of words in the language.
- Lexical items that describe concepts in specific fields are usually called terms instead of words.
- In theory, general dictionaries are supposed to be semasiological, mapping word to definition, while specialized dictionaries are supposed to be onomasiological.

- They first identify **concepts** and then establishing the terms used to designate them. In practice, the two approaches are used for both types.
- There are other types of dictionaries that do not fit into the above distinction. For example, **bilingual (translation) dictionaries,** dictionaries of synonyms (the saurs) and rhyming dictionaries.
- The word dictionary is usually meant to refer to a general purpose **monolingual dictionary.**
- There is also a difference between **Prescriptive** and **descriptive** dictionaries.
- The prescriptive dictionary reflects what is seen as correct use of the language.
- The descriptive reflects recorded actual use.
- Stylistic indications (e.g. 'informal' or 'vulgar') in many modern dictionaries are also considered by some to be less than objectively descriptive.

2.17.1 Types of Dictionaries

In a general dictionary, each word may have multiple meanings. Some dictionaries include each separate meaning in the order of most common usage while others list definitions.

In many languages, words can appear in many different forms, but only the **undeclined or unconjugated** form appears as the **headword** in most dictionaries.

Dictionaries are commonly found in the form of a book. But some dictionaries like New Oxford American Dictionary are dictionary software running on computers. Many online dictionaries are also available via internet.

2.17.2 Specialised Dictionaries

- According to Manual of specialised Lexicographies, **a Specialised dictionary,** is also referred to as a technical dictionary.
- It focuses on a specific subject field.
- Lexicographers categorise specialised dictionaries into three types :
  (i) A multi – field dictionary : It covers several subject fields, e.g. a business dictionary.
  (ii) A single - field dictionary covers one particular subject (e.g. law) and
  (iii) **A sub-field dictionary**. It covers a more specialised field (e.g. constitutional law).
- The **23-language Inter-Active Terminology for Europe** is a multi-field dictionary. The American National Biography is a single field.
- The African American National Biography project is a sub-field dictionary.
- Another variant is the **glossary,** an alphabetical list of defined terms in a specialised field, such as medicine **(medical dictionary).**

2.17.3 Defining Dictionaries

- A defining dictionary, provides a **core glossary** of the simplest meanings of the simplest concepts.

- From these, other concepts can be explained and defined.
- In English, the commercial defining dictionaries include only one or two meanings of under 2000 words. With these, the 4000 most common English idioms and metaphors can be defined.

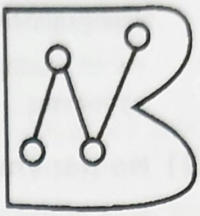
2.17.4 Historical Dictionaries

- A historical dictionary is a specific kind of descriptive dictionary. It describes the development of words and senses over time, using original source material to support its conclusions.
- Dictionaries for Natural Language Processing Dictionaries for Natural Language Processing (NLP) are Built to be used by Computer Programs.
- The direct user is a program, even though the final user is a human being. Such a dictionary does not need to be printed on paper.
- The structure of the content is not linear, ordered entry by entry. It has a complex network form.
- Since most of these dictionaries control **machine translations or cross-lingual information retrieval (CLIR)**, the content is usually multilingual and usually of huge size.
- To allow formalized exchange and merging of dictionaries, an ISO standard called **Lexical Makeup Framework** (LMF) has been defined and used among the industrial and academic community.

2.18 BABELNET DICTIONARY

- **Babelnet** is a multilingual lexicalised semantic network.
- Babelnet was automatically created by linking most popular computational lexicon of the English language, world net.
- The integration is done using an automatic mapping and by filling in lexical gaps in **resource-poor languages** by using **statistical machine translation.**
- The result is an encyclopaedic dictionary. It provides **concepts** and **named entities** that are **lexicalised** in many languages and connected with large amounts of semantic relations.
- Additional lexicalisations and definitions are added by linking to free – license word nets. Similar to wordnet, babelnet group – works in different languages are set into sets of synonyms, called babel **synsets.**
- Babelnet provides short definitions (called glosses) in many languages taken from wordnet.

2.18.1 BabelNet



Stable release : BabelNet 5.0 / February 2021
Operating system : Virtuoso Universal Server Lucene
Type : Multilingual encyclopedic dictionary Linked data
License : Attribution non-commercial share A like 3.0 unported
Website : babelnet.org

Fig. 2.18.1 : BabelNet

2.18.2 Statistics of BabelNet

- BabelNet (Version 5.0) covers 500 languages. It contains almost 20 million synsets and around 1.4. billion word senses.
- Each Babel Synset contains 2 synonyms per language, i.e. word senses, on average.
- Version 5.0 also associates around 51 million images with Babel Synsets and provides a Lemon RDF encoding of the resource, available via a SPARQL endpoint 2.67 million synsets are assigned domain labels.

2.18.3 Applications

- BabelNet has been shown to enable multilingual **Natural Language Processing** applications.
- The lexicalised **knowledge** available in Babelnet has been shown to obtain state-of-the-art results in :
  (i) Semantic relatedness
  (ii) Multilingual Word Sense Disambiguation.
  (iii) Multilingual Word Sense Disambiguation and Entity Linking with the Babelfy System
  (iv) Video games with a purpose.

2.19 RELATIONS ANONG LEXEMES AND THEIR SENSES

We have seen that semantic analysis can be divided into the following two parts :

(1) In the first part of the semantic analysis, the study of the meaning of individual words is performed. This part is called lexical semantics.
(2) In the second part, the individual words will be combined to provide meaning in sentences.

2.19.1 Important Elements of Semantic Analysis

We Mention below some important elements of Semantic Analysis,

(1) Hyponymy

- It is defined as the relationship between a generic term and instances of that generic term.

- The generic term is called hypernym and its instances are called hyponyms.
- As an example, the word colour is hypernym and the colour red, green etc are hyponyms.

(2) Homonymy

- It is 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 used to hit a ball or bat is a flying mammal also.

(3) Polysemy

- Polysemi means "many signs". It is a Greek word.
- It is a word or phrase with different but related sense.
- Polysemy has the same spelling but different and related meanings.
- For example, the word "bank" is a polysemy word with the following different meanings :
  (i) A financial institution
  (ii) The building in which such an institution is located.
  (iii) A synonym for "to rely on".

(4) Difference between Polysemy and Homonymy

Sr. No.	Polysemy	Homonymy
I	It has same spelling or syntax.	It has also same spelling or syntax.
II	The meanings of the word are related.	The meaning of the words are not related.
III	For example, the word "Bank" the meaning are related.	But for the word "Bank" we can write the meaning as a financial institution or a river bank. Here the meanings are not related, so it is an example of Homonymy

(5) Synonymy

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

(6) Antonymy

- It is a relation between two lexical items possessing symmetry between their semantic components relative to an axis.
- The scope of antonymy is a follows :
  (i) Application of property or not - Example is 'life/death', 'certitude/incertitude'.
  (ii) Application of scalable property : Example is 'rich/poor', 'hot/cold'.
  (iii) Application of a usage : Example is 'father/son', 'moon/sun'.

2.19.2 Ambiguity and Uncertainty in Language

- 'Ambiguity' refers to the meaning : 'Double Meaning'.
- Ambiguity in natural language processing refers to the ability of being understood in more than one way.
- We can say that ambiguity is the capability of being understood in more than one way obviously, Natural language is very ambiguous.
- We discuss various types of ambiguities in NLP :

(i) Lexical Ambiguity

The ambiguity of a single word is called lexical ambiguity. For example the word **walk** as a noun or a verb.

(ii) Syntactic Ambiguity

- When the sentence is parsed in different ways, this type of ambiguity occurs. For example, the sentence "The man saw the girl with camera."
- It is ambiguous, whether the man saw the girl with the camera or he saw the girl taking photos.

(iii) Semantic Ambiguity

- When the meaning of the words can be misinterpreted, such kind of ambiguity occurs.
- In short, semantic ambiguity occurs when a sentence contains an ambiguous word or phrase.
- For example, the sentence, "The bike hit the pole when it was moving" is having semantic ambiguity.
- The interpretation can be done as : "The bike, while moving, hit the pole and "The bike hit the pole while the pole was moving".

(iv) Anaphoric Ambiguity

- This type 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.

(v) Pragmatic Ambiguity

- When the context of a phrase gives multiple interpretation to the situation, then this kind of ambiguity arises.
- Thus when the statement is not specific, pragmatic ambiguity arises.
- For example, the sentence, "I like you too" can have multiple interpretations:
  (i) I like you (just as you like me),
  (ii) I like you (just like someone else dose).

2.20 WORD SENSE DISAMBIGUATION (WSD)

- To realise the various usage patterns in the language is important for various Natural Language Processing Applications.
- Word Sense Disambiguation is an important method of NLP, using it the meaning of a word can be determined. And that can be used in a particular context.
- The main problem of NLP systems is to identify words properly and to determine the specific usage of a word in a particular sentence.
- Word sense Disambiguation solves the ambiguity when that arises while determining the meaning of the same word, when it is used in different situations.

2.20.1 Word-Sense Disambiguation Applications

We mention below the various applications of WSD in various text processing and NLP fields.

(i) WSD can be used with Lexicography. Much of the modern Lexicography is corpus-based. WSD in Lexicography can provide significant textual indicators.

(ii) WSD can also be used in text mining and Information Extraction tasks.
It can be used for the correct labelling of words, because the main aim of WSD is to understand the meaning of a word accurately in a particular sentence.

(iii) From a security point of view, a text system should understand the difference between a **coal "mine"** and a **land "mine"**.

(iv) We note that the former serves industrial purposes, the latter is a security threat. Hence a text-mining application must be able to determine the difference between the two.

(v) WSD can be used for Information Retrieval purposes. Information Retrieval systems work through text data. And it is based on textual information. Hence knowing the relevance of using a word in any sentence helps.

2.20.2 Challenges in Word Sense Disambiguation

WSD faces lot many challenges and problems.

(i) The difference between various dictionaries or text-corpus is the most common problem.
Different dictionaries give different meanings for words. That makes the sense of the words to be perceived differently. A lot of text information is available and it is not possible to process everything properly.

(ii) Different algorithms are to be formed for different applications and that becomes a big challenge for WSD.

(iii) Words cannot be divided into discrete meaning they have related meanings. And this causes a lot of problems.

2.20.3 Relevance of WSD

- Word Sense Disambiguation is related to parts of speech tagging and is an important part of the whole Natural Language Processing process.
- The main problem that arises in WSD is the whole meaning of word sense. Word sense is not a numeric quantity that can be measured as a true or false, and can be denoted by 1 or 0.
- The meaning of a word is contextual and depends on its usage.
- Lexicography deals with generalising the corpus and explaining the full and extended meaning of a word. But sometimes these meanings fail to apply to the algorithms or data.
- But, WSD has immense applications and uses.
- If a computer algorithm can just read a text and come to know different uses of a text, it will indicate vast improvement in the field of text analytics.

2.21 KNOWLEDGE BASED APPROACH

A knowledge based system behaviour can be designed in following approaches :

(1) Declarative Approach

- In this approach, stating from an empty knowledge base, the agent can TELL sentences one after another.
- This is to be continued till the agent has knowledge of how to work with its environment. It stores required information in empty knowledge – based system.
- This is known as declarative approach.

(2) Procedural Approach

- In this method, it converts required behaviour directly into program code in empty knowledge – based system.
- Compared to declarative approach, it is a contrast approach. Here, coding system is designed.

2.21.1 Lesk Algorithm

The lesk algorithm is based on the assumption that words in a given 'neighbourhood' will tend to share a common topic.

In a simplified manner, the Lesk algorithm is to compare the dictionary definition of an ambiguous word with the terms contained in its neighbourhood.

Versions have been adapted to use wordnet. An implementation appears like this :

1. For every sense of the word being disambiguated one should count the number of words that are in both the neighbourhood of that word and in the dictionary definition of that sense.

2. The sense that is to be chosen is the sense that has the largest number of this count.

We consider an example illustrating this algorithm, for the context "pine cone".

Dictionary definitions are : PINE

1. Kind of evergreen tree with needle-shaped leaves.
2. Waste away through sorrow or illness.

CONE

1. Solid body which narrows to a point.
2. Something of this shape whether solid or hollow.
3. Fruit of certain evergreen trees.

We note that, the best intersection is pine #1∩cone#3 = 2.

2.21.2 Simplified Lesk Algorithm

- In simplified Lesk algorithm, the correct meaning of each word in a given context is determined: by noting down the sense that overlaps the most between its dictionary definition and the given context.
- Instead of collectively determining the meanings of all words in a given context, this approach takes into account each word individually, independent of the meaning of the other words occurring in the same context.
- A comparative evaluation performed has shown that simplified Lesk algorithm can outperform the original definition of the algorithm, both in terms of precision and efficiency.
- Evaluating the disambiguation algorithms on the Senseval-2 English all words data, they measure 58% precision using the simplified lesk algorithm compared to only 42% under the original algorithm.

2.21.3 Limitations of Lesk-Based Methods

(i) Lesk's approach is very sensitive to the exact wording of definitions.

(ii) The absence of a certain word can change the results considerably.

(iii) The algorithm determines overlaps only among the glosses of the senses being considered.

(iv) Dictionary glosses are fairly short and do not provide sufficient vocabulary to relate sense distinctions.

This is a significant limitation.

Different modifications of this algorithm have appeared. These works use other resources for analysis (the saur uses, synonyms dictionaries or morphological and syntactic models).

For example, it may use such information as synonyms, different derivatives, or words from definitions of words from definitions.

2.22 DICTIONARIES FOR REGIONAL LANGUAGES

(1) Hindi is the official language of India while English being the second official language. But there is no national language as per the constitution.

(2) **The oxford dictionary** is one of the most famous English language dictionaries in the world. It has many extra features that augment it as dictionary tool for language learners, like the ability to make notes on definitions and spellings, a flashcard learning system and a great the saurus.

(3) Hindi is the official language. In addition to the official language, the constitution recognises 22 regional languages, which does not include English, as scheduled languages.

(4) The **Sanskrit** language is the oldest language in India. Sanskrit language has been spoken since 5000 years before Christ. Sanskrit is still the official language of India. But, in the present time, Sanskrit has become a language of worship and ritual instead of the language of speech.

(5) There are 22 official regional languages in India. They are: Assamese, Bengali, Bodo, Dogri, Gujarati, Hindi, Kannada, Kashmiri, Konkani, Maithili, Malyalam, Manipuri, Marathi, Nepali, Oriya, Punjabi, Sanskrit, Santhali, Sindhi Tamil, Telugu and Urdu.

(6) The youngest language in India is Malayalam.

It belongs to Dravidian language group and considered an the smallest and the youngest language of the Dravidian language group. Government of India declared this language an 'the classical language of India in 2013'.

(7) Currently six languages enjoy the 'classical status' Tamil (declared m 2004), Sanskrit (2005),Kannada (2008), Telugu (2008), Malyalam(2013) and odiya (2014).

2.23 DISTRIBUTIONAL SEMANTICS

(i) Distributional semantics is an important area of research in natural language processing. It aims to describe meaning of words and sentences with vectorial representation. These representations are called distributional representations.

(ii) Distributional semantics is a research area that develops and studies theories and methods for quantifying and categorising semantics similarities between linguistic items based on their distributational properties in large samples of language data.

(iii) The aim of distributional semantics is to learn the meanings of linguistic expressions from a corpus of text.

The core idea, known as the distributional hy-pothesis, is that the contexts in which an expression appears give us information about its meaning.

(iv) Distributional evidence in linguistics

The distributional typothesis in linguistics is derived from the semantic theory of language usage, i.e. words that are used and occur in the same contexts tends to produce similar meanings.

The underlying idea that 'a word is charactised by the company it keeps' was popularised also here.

(v) **Distributional structure**

The distribution of an element will be understood as the sum of all its environments. An environment of an element, A is an existing array of its co-occurrents, i.e. the other elements, each in a particular position, with which A occurs to yield an utterance.

(vi) **Distributional properties**

There are three basic properties of a distribution : location, spread and shape.

The location refers to the typical value of the distribution, such as the mean. The spread of the distribution, is the amount by which similar values differ from larger ones.

(vii) **Semantic criteria**

A verb's meaning has to do with events. Correspondingly, we can say that a noun denotes an entity, adverbs modify events and so on.

One can call the classification of words on the basis of their meaning : a semantic criteria.

2.24 TOPIC MODELS

- Topic modelling is recognising the words from the topics present in the document or the corpus of data.
- This is useful because extracting the words from a document takes more time and is much more complex than extracting from them topics present in the document.
- For examples, there are 1000 documents and 500 words in each document. So to process this it requires (500) (1000) = 500000 threads.
- But if we divide the documents containing certain topics, if there are 5 documents present in it, the processing is just 5(500) = 2500 threads.
- This appears simple than processing the entire document and this is how topic modelling has come up to solve the problem and also visualising things better.

2.25 LATENT SEMANTIC ANALYSIS (LSA)

- Latent semantic analysis is a natural language processing method that uses the statistical approach to identify the association among the words in a document.
- LSA deals with the following kind of issue.
- **Example :** mobile, phone, cell phone are all similar but if we say "The cell phone is ringing." Then the documents which have "cell phone" are only retrieved whereas the documents containing the mobile, phone, telephone are not retrieved.

Assumption of LSA

1. The words which are used in the same context are analogous to each other.
2. the hidden semantic structure of the data is unclear due to the ambiguity of the words chosen.

Singular Value Decomposition (SVD)

SVD is the statistical method that is used to find the latent (hidden) semantic structure of words spread across the document.

Let

C = collection of documents.

d = number of documents,

n = number of unique words in the

M = $d \times n$

The SVD decomposes the M matrix i.e. word i.e. word to document matrix into three matrices as follows :

$$M = u \Sigma V_T$$

Where u = distribution of words across different contents.

Σ = diagonal matrix of the association among the contents.

V_T = distribution of contexts across the different documents.

2.26 SELF-LEARNING TOPICS

2.26.1 Dictionary look-up

- Dictionary in NLP means a list of all the unique words occurring in the corpus.
- If some words are repeated in different documents, they are all written just once while creating the dictionary.
- A dictionary contains a list of single word terms or multi-word terms. These terms represent instances of a single concept.
- For example, there might be a list of countries to extract the concept country.
- In addition to the base form for each term, the dictionary can contain several variants of the base form. Each term includes a unique number that is called term ID.
- The type of annotations that are created for dictionary entries within text have the same name as the dictionary.

2.26.2 Detail Explanation of Dictionary Lookup

- Morphological parsing is a process by which word forms of a language are associated with corresponding linguistic descriptions. Morphological systems that specify these associations by merely enumerating them case by case do not offer any generalization means. Likewise for systems in which analyzing a word

form is reduced to looking it up verbatim in word lists, dictionaries, or databases, unless they are constructed by and kept in sync with more sophisticated models of the language.

- In this context, a dictionary is understood as a data structure that directly enables obtaining some precomputed results, in our case word analyses. The data structure can be optimized for efficient lookup, and the results can be shared. Lookup operations are relatively simple and usually quick. Dictionaries can be implemented, for instance, as lists, binary search trees, tries, hash tables, and so on.
- Because the set of associations between word forms and their desired descriptions is declared by plain enumeration, the coverage of the model is finite and the generative potential of the language is not exploited. Developing as well as verifying the association list is tedious, liable to errors, and likely inefficient and inaccurate unless the data are retrieved automatically from large and reliable linguistic resources.
- Despite all that, an enumerative model is often sufficient for the given purpose, deals easily with exceptions, and can implement even complex morphology. For instance, dictionary-based approaches to Korean [35] depend on a large dictionary of all possible combinations of allomorphs and morphological alternations. These approaches do not allow development of reusable morphological rules, though [36].
- The word list or dictionary-based approach has been used frequently in various ad hoc implementations for many languages. We could assume that with the availability of immense online data, extracting a high-coverage vocabulary of word forms is feasible these days [37]. The question remains how the associated annotations are constructed and how informative and accurate they are. References to the literature on the unsupervised learning and induction of morphology, which are methods resulting in structured and therefore non enumerative models, are provided later in this chapter.

☞ Example for dictionaries

The following table shows the contents of the dictionary countries

Base form	Variant
Sri Lanka	Ceylon
Germany	BRD
United states	US; USA

Actually Sri Lanka was known as Ceylon, the following annotations are created.

Annotation 1 :
Type : Countries
Base form : Sri Lanka
Id : 1
begin : ⊙
end : 9
covered Text : Sri Lanka

Annotation 2 :
Type : Countries
Baseform : Sri Lanka
id : 1
begin : 20
end : 26
covered Text : Ceylon

Annotations include the following features

Baseform
The base form of the recovered variants

id
The ID of the dictionary entry

begin
The offset that indicates the begin of the covered text

end
The offset that indicates the end of the covered text.
Covered text
The variant of the base form that is found in the text.

- One can also import dictionaries in the design studio dictionary – XML format or dictionaries that are compatible with language wave dictionary-resource format
- One can use dictionaries with the Dictionary Lookup operator. These dictionaries might contain more than one annotation type, and the features for these annotation types might vary from type to type.

2.26.3 Finite State Morphology

- By finite-state morphological models, we mean those in which the specifications written by human programmers are directly compiled into finite-state transducers. The two most popular tools supporting this approach, which have been cited in literature and for which example implementations for multiple languages are available online, include XFST (Xerox Finite-State Tool) [9] and LexTools [11].
- Finite-state transducers are computational devices extending the power of finite-state automata. They consist of a finite set of nodes connected by directed edges labeled with pairs of input and output symbols. In such a network or graph, nodes are also called states, while edges are called arcs.
- Traversing the network from the set of initial states to the set of final states along the arcs is equivalent to reading the sequences of encountered input symbols and writing the sequences of corresponding output symbols.

- The set of possible sequences accepted by the transducer defines the input language; the set of possible sequences emitted by the transducer defines the output language. For example, a finite-state transducer could translate the infinite regular language consisting of the words vnuk, pravnuk, prapravnuk, ... to the matching words in the infinite regular language defined by grandson, great-grandson, great-great-grandson, ...
- The role of finite-state transducers is to capture and compute **regular relations** on sets [38, 9, 11]. That is, transducers specify relations between the input and output languages. In fact, it is possible to invert the domain and the range of a relation, that is, exchange the input and the output.
- In finite-state computational morphology, it is common to refer to the input word forms as **surface strings** and to the output descriptions as lexical strings, if the transducer is used for morphological analysis, or vice versa, if it is used for morphological generation.
- Morphology is a domain of linguistics that studies the formation of words. It is traditional to distinguish between **surface forms** and their analyses, called as **lemmas**.
- The lemma for a surface form such as the English word **bigger** might be represented as **big+ Adj + comp**. To indicate that **bigger** is the comparative form of the adjective big.
- In modelling natural language morphology, we come across two challenges

1. Morphotactics

- Words are typically composed of smaller units of meaning, called morphemes.
- The morphemes that make up a word must be combined in a certain order : **piti-less-ness** is a word in English but **piti-ness-less** is not.
- Most languages build words by concatenation but some languages also exhibit non-concatenative process such as interdigitation and reduplication.

2. Morphological Alternations

- The shape of a morpheme often depends on the environment : **pity** is realised as **piti** in the context of **less**, **die** as **by** in **dying**.
- The basic claim of the finite-state approach to morphology is that the relation between the surface-forms of a language and their corresponding lemmas can be described as a **regular relation**.
- If the relation is regular, it can be defined using the metalanguage of regular expressions. Then with a suitable compiler, the regular expression source code can be compiled into a finite-state transducer and that implements the relation computationally.
- In the resulting transducer, each path (= sequence of states and arcs) from the initial state to a final state represents a mapping between a surface form and its lemma. This is known as the **lexical form**.
- For example, the comparative of the adjective **big** is **bigger** can be represented in English lexical transducer by the path in Fig. 2.26.1 where the zeros represent epsilon symbols.

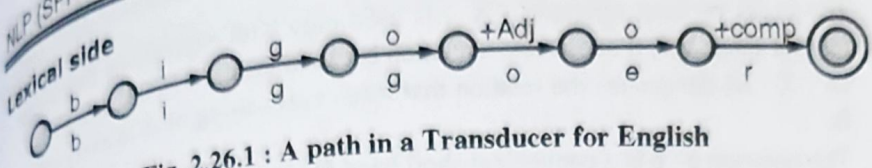


Fig. 2.26.1 : A path in a Transducer for English

- If the symbols are distinct, the pairs b : b are reduced to a single symbol.
- In standard notation, the path in Fig. 2.26.1 is labelled as
- bigo : g + Adj : 0 0 : e + comp : r
- Lexical transducers may contain hundreds of thousands, even millions, of states and arcs and an infinite number of paths in the case of languages such as German. German language allows noun compounds of any length.
- The regular expressions from which such complex networks are complied include high-level operators. These are developed to make it possible to describe constraints and alternations that are commonly found in natural languages. And these are present in a convenient and perspicuous way.

Basic Expression Calculus

- The notation used here comes from the Xerox finite-state calculus.
- We use uppercase letters A, B, etc; as variables over regular expressions. Lower case letters a, b etc. Stand for symbols.
- There are two special symbols : O, the Epsilon symbol, that stands for empty string and ?, the any symbol.
- Complex regular expressions can be built up from simpler ones by means of regular expression operators. Square brackets, [ ], are used for grouping expressions.
- Because both regular languages and regular relations are closed under concatenation and union, the following basic operators can be combined with any kind of regular expression :

  A | B union,

  A B concatenation

  (A) Optionality ; equivalent to [A | 0],

  A + Iteration; one or more concatenations of A,

  A* kleene star ; equivalent to (A+)

(I) Regular languages are closed under complementation, subtraction, and intersection, but regular relations are not.

Hence, the following operators can be combined only with a regular language :

~ A complement

\ A Term complement; all single symbols strings not in A.

A & B Intersection

A – B Subtraction (minus)

(II) Regular relations can be constructed by means of two basic operators :

A · X · B Cross product

A · O · B composition

The cross product operator, · X ·, is used only with expressions that denote a regular language; it constructs a relation between them.

[A · X · B] designates the relation that maps every string of A to every string of B.

The notation a : b is a convenient short hand for [a · x · b].

Remarks

(1) Replacement and marking expression in regular expressions have turned out to be very useful for morphology, tokenization and parsing.

(2) Descriptions consisting of regular expressions can be efficiently complied into finite-state-networks. And they can be determined, minimised in other ways to reduce the size of the network.

(3) Also they can be sequentialised, compressed, and optimised to increase the application speed.

(4) Regular expressions have semantics which are clean and declarative.

(5) They constitute a kind of high level programming language for manipulating strings, languages and relations.

(6) Regular languages and relations can be encoded as finite automata, they can be more easily manipulated than context-free and more complex languages.

(7) With regular-expression operators, new regular languages and relations can be derived directly without mentioning the new grammar rules.

This is a fundamental advantage over other higher-level formalism.

2.26.4 Noisy Channel Models

(1) The noisy channel model is a framework used in natural language processing (NLP) to identify the correct word in situations where it is unclear.

The framework helps detect intended words for spell checkers, virtual assistants, translation programs, question answering systems and speech to text software.

(2) Difference Between Noisy Channel and Noiseless Channel

- The capacity of a noiseless channel is numerically equal to the rate at which it communicates binary digits.
- The capacity of a noisy channel is less than this because it is limited by the amount of noise in the channel.
- Noiseless channel means that there will not any kind of disturbance in the path when data is carried forward from sender to receiver.

(3) Capacity of noisy and noiseless channel

- The channel capacity is directly proportional to the power of the signal as :

  SNR = (Power of signal) / (Power of noise)

A signal noise ratio of 1000 is commonly expressed as

$$10[\log_{10}(1000)] = [\log_{10}(10^3)]$$
$$= 10[3\log_{10}(10)]$$
$$= 10(3 \times 1) = 30 \text{ dB}$$

(4) The maximum data rate of noisy channel ?

- The amount of thermal noise is calculated as the ratio of signal power to noise power, SNR. Since SNR is the ratio of two powers that varies over a very large range, it is often expressed in decibles, called SNR_{db} and calculated as :

$$SNR_{db} = 10[\log_{10} SNR]$$

- With these characteristics, the channel can never transmit much more than 13 Mbps, no matter how many or how few signal levels are used and no matter how often or how infrequently samples are taken.

Examples : A telephone line normally has a bandwidth of 300 Hz (3000 to 3000 Hz) assigned for data communication.

(5) Noiseless Channel

An idealistic channel in which no frames are lost, corrupted or duplicated. The protocol does not implement error control in this category.

☞ Various edit distance

(1) In computational linguistics and computer science, edit distance is a way of quantifying how dissimilar two strings (e.g. words) are to one another by counting the minimum number of operations required to transform one string into the other.

(2) The maximum edit distance between any two strings (even two identical ones) is infinity, unless we add some restrictions on repetitions of edits. In spite of that there can be an arbitrarily large edit distance with any arbitrarily large set character set.

(3) The minimum edit distance between two strings is defined as the minimum number of editing operations (insertion, deletion, substitution) needed to transform one string into another.

(4) Operations in edit distance

Most commonly, the edit operations for this purpose are :

(i) insert a character into a string,

(ii) delete a character from a string, and

(iii) replace a character of a string by another character;

For these operations, edit distance is sometimes called as Levenshtein distance.

(5) The normalised edit distance is one of the distances derived from the edit distance. It is useful in some applications because it takes into account the length of the two strings compared.