Index

□□□

MODULE 1

CHAPTER 1 Introduction to NLP

Syllabus

1.1 Origin & History of NLP; Language, Knowledge and Grammar in language processing; Stages in NLP; Ambiguities and its types in English and Indian Regional Languages; Challenges of NLP; Applications of NLP.

1.2 Self-Learning topics: Variety types of tools for regional languages pre-processing and other functionalities.

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 ofwords.
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 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.

1.5 STAGES IN NLP

There are five phases of NLP (Refer Fig. 1.5.1)

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.

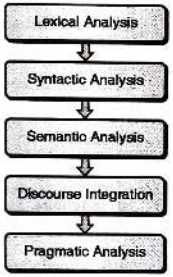


Fig. 1.5.1 : Basic steps of NLP

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 or 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.').

1.5.1 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.

1.6 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).

1.6.1 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.

Applications

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.

1.7 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.

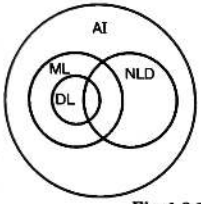
The solution here is to develop a NLP system that can recognise its own limitations, and use questions to clear up the ambiguity.

(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.

1.8 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.



AI = Artificial intelligence
ML = Machine learning
DL = Deep learning
NLP = Natural language processing

Fig. 1.8.1

- 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.
- 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. 1.8.2

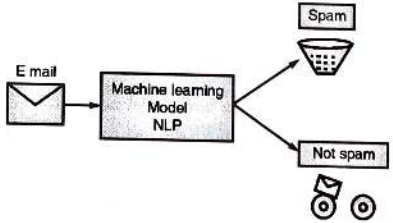


Fig. 1.8.2

(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.

1.9 ADVANTAGES AND DISADVANTAGES OF NLP

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

1.9.1 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.

1.9.2 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.

1.10 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.

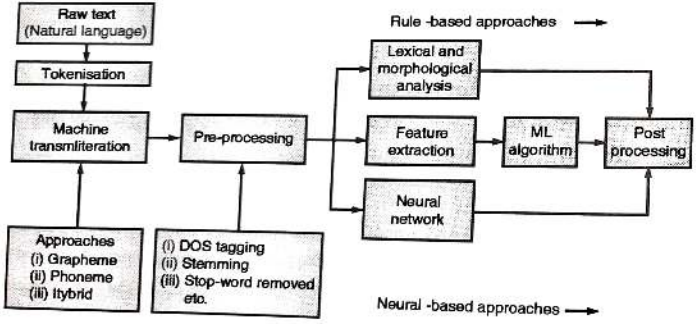


Fig. 1.10.1

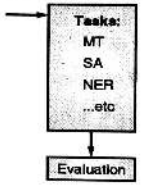


Fig. 1.10.2

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
/ v /	/ a /
/ v« /	/'a' or 'a' /

Chapter Ends...

□□□

MODULE 2

CHAPTER 2 Word Level Analysis

Syllabus

Basic Terms : Tokenization, Stemming, Lemmatization; Survey of English Morphology, Inflectional Morphology, Derivational Morphology; Regular expression with types;

Morphological Models: Dictionary lookup, finite state morphology; Morphological parsing with FST (Finite State Transducer);Lexicon free FST Porter Stemmer algorithm; Grams and its variation: Bigram, Trigram; Simple (Unsmoothed) N-grams;

N-gram Sensitivity to the Training Corpus; Unknown Words: Open versus closed vocabulary tasks; Evaluating N-grams: Perplexity; Smoothing: Laplace Smoothing, Good-Turing Discounting;

Self-Learning topics : Noisy channel models, various edit distance, Advance Issues in Language Modelling

2.1 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?

2.1.1 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.

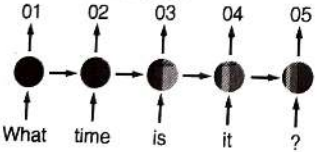


Fig. 2.1.1

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.

2.1.2 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.

2.1.3 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.

2.1.4 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.

2.1.5 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']

2.1.6 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.

2.1.7 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.

2.1.8 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.

2.2 STEMMING

GQ. What are the benefits of tokenization ? what is its importance ?

- 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.

2.2.1 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.

2.2.2 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.

2.2.3 Types of Stemmer in NLTK

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

GQ. Discuss different types of stemmer in NLTK

2.2.3.1 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
```


2.2.3.2 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
```

2.2.3.3 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
```

2.2.3.4 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] :

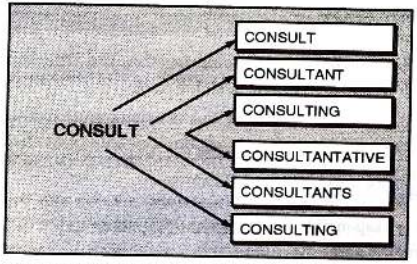
```
mass → mas
was → was
bee → bee
computer → computer
advisable → advis
```

2.2.4 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.

2.3 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.

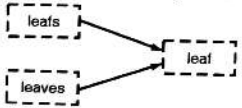


Fig. 2.3.1

- **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.

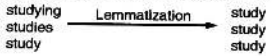


Fig. 2.3.2

2.3.1 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.

2.3.2 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.

Sr. No.	Stemming	Lemmatization
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

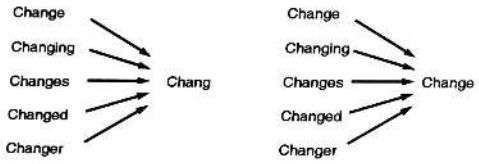


Fig. 2.3.3

2.3.3 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.

2.3.4 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.

2.3.5 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.

2.3.6 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.

2.4 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)

2.4.1 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.

2.4.2 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"

2.4.3 Inflectional Morphology

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

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

(ii) 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.

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

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

(v) 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,

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

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

(vi) 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.

(vii) 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

2.4.4 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

2.4.5 Derivational Morphology

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

(i) 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.

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

(iii) 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.

(iv) 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)

2.4.6 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,	Inflection typically adds grammatical information about number (singular, dual, plural), person (first, second,

	Derivation	Inflection
	employer, employee) or an adjective (employable), or taking a noun like union and making it a verb (unionise) or an adjective (unionish, unionesque).	third), tense (past, future), aspect (perfective, imperfective, habitual) and case (nominative, accusative), among other grammatical categories that languages might mark.
	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.

2.4.7 Regular Expression with Types

A regular expression (RE) is a language for specifying text search strings. RE helps up to match or find other strings or sets of strings, using a specialised syntax held in a pattern

Regular expressions are used to search texts in UNIX as well as in MS WORD in identical way.

We have various search engines using a number of RE features.

2.4.8 Properties of Regular Expressions

Followings are some of the important properties of RE :

(a) RE is a formula in a special language which can be used for specifying simple classes of strings, a sequence of symbols.

It implies that RE is an algebraic notation for characterising a set of strings.

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

Mathematically, a Regular Expression can be defined as follows :

(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

(a) X, Y

(b) X · Y (concatenation of X Y)

(c) X + Y (union of X and Y)

(d) X* , Y * (**Kleen** closure of X and Y) are also regular expressions.

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

Remark

An American Mathematician Stephen cole kleen formalised the Regular Expression language.

2.4.9 Examples of Regular Expressions

The table 2.4.1 shows a few examples of Regular Expressions

Table 2.4.1

Regular Expressions	Regular Set
(0 + 10*)	{0, 1, 10, 100, 1000, 10000, ...}
(0* 10*)	{1, 01, 10, 010, 0010, ...}
(0 + ϵ) (1 + ϵ)	{ϵ, 0, 1, 01}
(a + b)*	It is a set of strings of a's and b's of any length which also include the null string i.e. {ϵ, a, b, aa, ab, bb, ba, aaa....}
(a + b)* abb	It is a set of strings of a's and b's ending with the string abb i.e. {abb, aabb, babb, aaabb, ababb,....}

Regular Expressions	Regular Set
(aa)* (bb)* b	It is a set consisting of even number of a's followed by odd number of b's i.e. {b, aab, aabbb, aabbb, aaaab,....}
(11)*	It is a set consisting of even number of 1's which also includes an empty string, i.e. {∈, 11, 1111, 111111, ...}
(aa + ab + ba + bb)*	It is string of a's and b's of even length that can be obtained by concatenating any combination of strings aa, ab, ba and bb including null, i.e.{aa, ab, ba, bb, aaab, aaba,....}

2.4.10 Regular Sets and Their Properties

Regular set is defined as the set that represents the value of the regular expression and consists of specific properties

Properties of Regular Sets

(i) The union of two regular sets is also a regular set.

(ii) The intersection of two regular sets is also regular.

(iii) The complement of regular set is also regular

(iv) The difference of two regular sets is also regular

(v) The reversal of regular sets is also regular.

(vi) The closure of regular sets is also a regular set.

(vii) The concatenation of two regular sets, is also regular.

2.4.11 Finite State Automata

- Automata is the plural of automation and automation mean 'self-acting'.
- It (Finite state Automata) is defined as an abstract self-propelled computing device that follows a predetermined sequence of operations automatically.
- An automation having a finite number of states is called a finite Automation (FA) or Finite State Automata (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 ($q_0 \in Q$).

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

2.4.12 Relation between Finite Automata, Regular Grammars and Regular Expressions

W mention below the relationship between Finite Automata, Regular Grammars and Regular Expressions

(i) Finite state Automata are the theoretical foundation of computational work and regular expressions 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) Regular expression characterises a kind of language, called as regular language. Hence, 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 another way to characterise regular language.

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

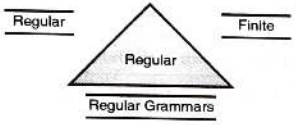


Fig. 2.4.1

2.4.13 Types of Finite State Automation (FAS)

There are two types of finite state automation :

(I) 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 a finite number of states, hence the machine is called as deterministic finite Automation (DFA).

Mathematically, a DFA 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 where

$$\delta : Q \times \Sigma \rightarrow Q.$$

(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 state diagrams where :

(a) The states are represented by **vertices**

(b) The transitions are shown by labelled **arcs**

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

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

Example of DFA :

Suppose a DFA be

(i) Q = {a, b, c}

(ii) Σ = {0, 1}

(iii) q_0 = {a},

(iv) f = {c}

(v) Transition function δ is as shown in the table.

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

The graphical representation of this DFA is :

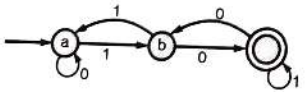


Fig. 2.4.2

(II) Non-deterministic Finite Automation (NDFA)

It is defined as the type of finite automation for every input symbol, the machine may move to any combination of the states. Again, it has finite number of starts.

Mathematically, NDFA 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 where

$$\delta : Q \times \Sigma \rightarrow Q$$

(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, ($F \subseteq Q$).

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

(a) The state are represented by **vertices**

(b) The transition are shown by labelled **arcs**

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

(d) The final state is represented by double circle.

Example of NDFA

Suppose a NDFA be

(i) Q = {a, b, c} (ii) Σ = {0, 1}

(iii) q_0 = {a} (iv) F = {c}

(v) Transition function δ is as shown :

Current state	Next state for input 0	Next state for input 1
A	a, b	B
B	C	a, c
C	b, c	C

We give below the graphical representation of NDFA

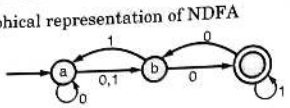


Fig. 2.4.3

2.4.14 Basic Regular Expression Pattern

A regular expression is also called as Rational Expression.

It is a sequence of characters that define a search pattern, for use in pattern matching with strings, or string matching.

Regular expressions are a generalised way to match patterns with sequences of characters. It is used in every programming language like C++ and Python.

GQ. How to write regular expression ?

(i) Repeaters : *, + and {}

These symbols act as repeaters and tell the computer that the preceding character is to be used for more than just one time.

(ii) The asterisk symbol (*)

It tells the computer to match the preceding character for 0 or more times.

Examples : The regular expression ab*c will give ac, abc, abbc, abbbc, ...so on .

(iii) The plus symbol (+)

It tells the computer to repeat the preceding character for at least one or more times.

Example : The regular expression like ab+c will give abc, abbc, ... and so on.

(iv) The curly braces {....}

It tells the computer to repeat the preceding character for as many times as the value inside this bracket.

Example : {3} means that the preceding character is to be repeated 3 times, {min,} means the preceding character matches min or more times,

{min, max} means that the preceding character is repeated at least min and at most max times.

(v) Wild Card – (.)

The dot symbol can take place of any other symbol, that is why it is called the wildcard character.

(vi) Optional Character : (?)

The symbol tells the computer that the preceding character may or may not be present in the string to be matched.

Example : We may write the format for document file as : The '?' tells the computer that x may or may not present in the name of file format.

(vii) The Caret (^) Symbol :

Setting position for match : tells the computer that the match must start at the beginning of the string or line.

Example : ^ \ d {3} will match patterns like "901" in "901 – 333.."

(viii) The Dollar ($) Symbol

It tells the computer that the match must occur at the end of the string or before at the end of the line or string.

Example : – \d {3} $ will match with patterns like "–333" in "901 – 333"

(ix) Character Classes

A character class matches any one of a set of characters. If is used to match the most basic element of a language like a letter, a digit, space, a symbol etc.

/S : matches any whitespace characters such as space and tab.

/S : matches any non-whitespace characters

/d : matches any digit character

/D : matches any non-digit characters.

/w : matches any word character (basic alpha-numeric)

/w : matches any non-word character.

/b : matches any word boundary (this includes spaces/ dashes/ commas/ senses- colons etc)

(x) [Set- of- characters] – Matches any single character in set-of-characters.

By default, the match is case-sensitive.

Example : [abc] will match characters a,b and c in any string.

(xi) [^ set of -characters] – Negation :

Matches any single character that is not in set-of-characters.

By default, the match is case sensitive

Example : [^ abc] will match any character except a, b, c.

(xii) [First-last]

Character range : Matches any single character in the range from first to last.

Example : [a – z A – z] will match any character from a to z or A to Z.

**(xiii) The Escape Symbol : **

To match for the actual '+', ' • ' etc. Characters, add a backslash (\ ) before that character.

This will tell the computer to treat the following character as a search character and consider it for matching pattern.

Example : \ d + [* – × *] \ d + will match patterns and "3 * 3" in "(2 + 2) * 3 * 9"

(xiv) Grouping Characters ( )

A set of different symbols of a regular expression can be grouped together to act as a single unit and behaves as a block. For this, one needs to wrap the regular expression in the parenthesis ( ).

Example : ([A – Z] \ w*) contains two different elements of the regular expression combined together. This expression will match any pattern containing uppercase letter followed by any character.

(xv) Vertical Bar (|)

Matches any one element separated by the vertical bar (|) character

Example : th (e | is | at) will match words the , this and that.

(xvi) \ number

Back reference : allows a previously matched sub-expression (expression enclosed within circular brackets) to be identified subsequently in the same regular expression. \ n means that group enclosed within the n-th bracket will be repeated at current position.

Example : ([a – z] \ ) will match "ee" in Geek because the character at second position is same as character at position 1 of the match.

(xvii) Comment : (? # comment)

Inline comment : The comment ends at the first closing parenthesis

Example : \bA (? # This is an inline comment) \ w* \ b

(xviii) # [to end of line] : X-mode comment

The comment starts at an unescaped # and continues to the end of line.

Example : (? ×) \ bA\ w* \ b # Matches words starting with A

2.4.15 Disjunction Grouping and Precedence

(I) Disjunction

Two atoms or groups separated by the meta character | (vertical bar) indicate the disjunction of the two expressions.

Example : The patten **lessly $ | lessness $** will match all words ending in lessly or lessness (e.g., aimlessly, carelessly, countlessness etc.)

The **precedence** of the | meta character is low so it is necessary to isolate with parenthesis the disjunctive part from the rest.

(II) Grouping

A sequence of atoms may be grouped if they are enclosed within parenthesis. For example, the pattern aimless(ly) ? will match both aimless and aimlessly.

The pattern :

^ [bcdfghjklmnpqrstvwxyz] a) + $ selects all words containing only an alternation of a lower case consonant and the letter a (banana, data, gala, lava, pa ta).

(III) Precedence

If one operator takes precedence over another, then we have to use parenthesis to specify what we mean.

This is what we name as the **operator precedence hierarchy, for regular** expressions.

2.5 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.

2.5.1 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 : 2⊙
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.

2.5.2 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].[5]
- 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].[6] 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. 1 where the zeros represent epsilon symbols.

Lexical side

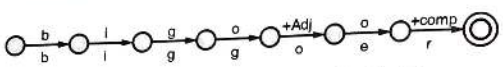


Fig. 2.5.1 : A path in a Transducer for English

- If the symbols are distinct, the pairs b : b are reduced to a single symbol.
- In standard notation, the path in Fig. 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.

2.6 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.

2.6.1 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.

2.6.2 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.

2.7 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.

2.7.1 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.

2.7.2 Difference between Stemming and Lemmatization

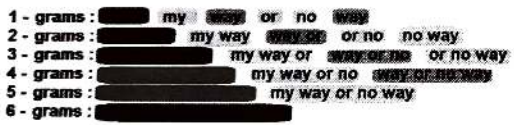
	Stemming	Lemmatization
(i)	Stemming is a process that stems or removes last few characters from a word, often leading to incorrect meanings and spellings	Lemmatization considers the context and converts the word to its meaningful base form, which is called Lemma.
(ii)	For example, stemming the word 'Caring' 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

2.8 GENERATING UNIGRAM, BIGRAM, TRIGRAM AND NGRAMS IN NLTK

- In this tutorial, we will understand the concept of ngrams in NLP and why it is used along with its variations like Unigram, Bigram, Trigram. Then we will see examples of ngrams in NLTK library of Python and also touch upon another useful function everygram. So let us begin.

GQ. What is n-gram Model

- In natural language processing n-gram is a contiguous sequence of n items generated from a given sample of text where the items can be characters or words and n can be any numbers like 1,2,3, etc.
- For example, let us consider a line – "Either my way or no way", so below is the possible n-gram models that we can generate –
- In natural language processing n-gram is a contiguous sequence of n items generated from a given sample of text where the items can be characters or words and n can be any numbers like 1,2,3, etc.
- For example, let us consider a line – "Either my way or no way", so below is the possible n-gram models that we can generate –



Use of n-grams in NLP

- N-Grams are useful to create features from text corpus for machine learning algorithms like SVM, Naive Bayes, etc.
- N-Grams are useful for creating capabilities like autocorrect, auto completion of sentences, text summarization, speech recognition, etc.

Generating ngrams in NLTK

- We can generate ngrams in NLTK quite easily with the help of ngrams function present in nltk.util module. Let us see different examples of this NLTK ngrams function below.

Unigrams or 1-grams

- To generate 1-grams we pass the value of n=1 in ngrams function of NLTK. But first, we split the sentence into tokens and then pass these tokens to ngrams function.
- As we can see we have got one word in each tuple for the Unigram model.

In [1]:

```
from nltk.util import ngrams
n = 1
sentence = 'You will face many defeats in life, but never let yourself be defeated.'
unigrams = ngrams(sentence.split(), n)

for item in unigrams:
  print(item)
```

[Out] :

```
('You',)
('will',)
('face',)
('many',)
('defeats',)
('in',)
('life,',)
('but',)
('never',)
('let',)
('yourself',)
('be',)
('defeated.',)
```

Bigrams or 2-grams

- For generating 2-grams we pass the value of n=2 in ngrams function of NLTK. But first, we split the sentence into tokens and then pass these tokens to ngrams function.
- As we can see we have got two adjacent words in each tuple in our Bigrams model.

In [2]:

```
from nltk.util import ngrams

n = 2
sentence = 'The purpose of our life is to happy'
unigrams = ngrams(sentence.split(), n)

for item in unigrams:
  print(item)
```

[Out] :

```
('The', 'purpose')
('purpose', 'of')
('of', 'our')
('our', 'life')
('life', 'is')
('is', 'to')
('to', 'happy')
```

☞ **Trigrams or 3-grams**

- In case of 3-grams, we pass the value of n=3 in ngrams function of NLTK. But first we split the sentence into tokens and then pass these tokens to ngrams function.
- As we can see we have got three words in each tuple for the Trigram model.

In [3]:

```
from nltk.util import ngrams

n = 3
sentence = 'Whoever is happy will make others happy too'
unigrams = ngrams(sentence.split(), n)

for item in unigrams:
  print(item)
```


[Out] :

```
('Whoever', 'is', 'happy')
('is', 'happy', 'will')
('happy', 'will', 'make')
('will', 'make', 'others')
('make', 'others', 'happy')
('others', 'happy', 'too')
```

☞ **Generic Example of ngram in NLTK**

- In the example below, we have defined a generic function ngram_convertor that takes in a sentence and n as an argument and converts it into ngrams.

In [4]:

```
from nltk.util import ngrams
def ngram_convertor(sentence,n=3):

  ngram_sentence = ngrams(sentence.split(), n)
  for item in ngram_sentence:
    print(item)
In [5]:
sentence = "Life is either a daring adventure or nothing at all"
ngram_convertor(sentence,3)
```

[Out] :

```
('Life', 'is', 'either')
('is', 'either', 'a')
('either', 'a', 'daring')
('a', 'daring', 'adventure')
('daring', 'adventure', 'or')
('adventure', 'or', 'nothing')
('or', 'nothing', 'at')
('nothing', 'at', 'all')
```

☞ **NLTK Everygrams**

- NTK provides another function everygrams that converts a sentence into unigram, bigram, trigram, and so on till the ngrams, where n is the length of the sentence. In short, this function generates ngrams for all possible values of n.

- Let us understand everygrams with a simple example below. We have not provided the value of n, but it has generated every ngram from 1-grams to 5-grams where 5 is the length of the sentence, hence the name everygram.

In [6]:

```
from nltk.util import everygrams

message = "who let the dogs out"
msg_split = message.split()
list(everygrams(msg_split))
```

[Out] :

```
[('who',),
 ('let',),
 ('the',),
 ('dogs',),
 ('out',),
 ('who', 'let'),
 ('let', 'the'),
 ('the', 'dogs'),
 ('dogs', 'out'),
 ('who', 'let', 'the'),
 ('let', 'the', 'dogs'),
 ('the', 'dogs', 'out'),
 ('who', 'let', 'the', 'dogs'),
 ('let', 'the', 'dogs', 'out'),
 ('who', 'let', 'the', 'dogs', 'out')]
```

Converting data frames into Trigrams

- In this example, we will show you how you can convert a data frames of text into Trigrams using the NLTK ngrams function.

In [8]:

```
import pandas as pd
df=pd.read_csv('file.csv')
df.head()
```


[Out] :

	headline_text
0	Southern European bond yields hit multi-week lows
1	BRIEF-LGselss its entire stake in unit LGLIF....
2.	Brief-Golden Wheel-Tiandisays unit confirms s....
3.	BRIEF-Sunshine 100 China Holdings Dec contract....
4.	Euro zone stockes start 2017 with new one year....

In [15]:

```
from nltk.util import ngrams
def ngramconvert(df,n=3):
  for item in df.columns:
    df['new'+item]=df[item].apply(lambda sentence: list(ngrams(sentence.split(), n)))
  return df
```

In [23] :

```
new_df = ngramconvert(df,3)
new_df.head()
```

Out[23]:

	headline_text	Newheadling_text
0	Southern European bond yields hit multi-week lows	[(Southern, European, bond), European, bond,...]
1	BRIEF-LG sells its entire stake in unit LG Lif....	[(Brief-LG, sells, its), sells, its, entire).....
2	BRIEF-Golden Wheel Tiandi says unti confirms s....	[BRIEF-Golden, Wheel, Tiandi), (Wheel, Tiandi....
3.	BRIEF-Sunshine 100 China Holdings Dec contract	[(BRIEF-Sunshine, 100, China), (100, China, Ho....
4.	Euro zone stocks start 2017 with new one-year	[(Euro, zone stocks), (zone, stocks, start), ...

2.8.1 Language Models: N-Gram

- A step into statistical language modelling
- One can think of an N-gram as the sequence of N words, by that notion, a 2-gram (or bigram) is a two-word sequence of words like "please turn", "turn your", or "your homework", and a 3-gram (or trigram) is a three-word sequence of words like "please turn your", or "turn your homework"

2.8.2 Intuitive Formulation

- Let's start with equation P(w|h), the probability of word w, given some history, h. For example,

P(the | its water is so transperant that)

Here,
w = The
h = its water is so transparent that

- And, one way to estimate the above probability function is through the relative frequency count approach, where you would take a substantially large corpus, count the number of times you see ***its water is so transparent that***, and then count the number of times it is followed by ***the***. In other words, you are answering the question:
- Out of the times you saw the history h, how many times did the word w follow it

  P(the | its water is so transparent that) = C (its water is so transparent that the) / C (its water is so transparent that)
- Now, you can imagine it is not feasible to perform this over an entire corpus; especially it is of a significant a size.
- This shortcoming and ways to decompose the probability function using the chain rule serves as the base intuition of the N-gram model. Here, you, instead of computing probability using the entire corpus, would approximate it by just a few historical words

2.8.3 The Bigram Model

- As the name suggests, the bigram model approximates the probability of a word given all the previous words by using only the conditional probability of one preceding word. In other words, you approximate it with the probability: P(the | that)

- And so, when you use a bigram model to predict the conditional probability of the next word, you are thus making the following approximation:

  $P(w_n \mid w_n^{n-1}) \approx P(w_n \mid w_{n-1})$
- This assumption that the probability of a word depends only on the previous word is also known as **Markov** assumption.
- Markov models are the class of probabilistic models that assume that we can predict the probability of some future unit without looking too far in the past.
- You can further generalize the bigram model to the **trigram model** which looks two words into the past and can thus be further generalized to the **N-gram model**

2.8.4 Probability Estimation

- Now, that we understand the underlying base for N-gram models, you'd think, how can we estimate the probability function. One of the most straightforward and intuitive ways to do so is **Maximum Likelihood Estimation (MLE)**
- For example, to compute a particular bigram probability of a word ***y*** given a previous word ***x***, you can determine the count of the bigram C(xy) and normalize it by the sum of all the bigrams that share the same first-word ***x***.

2.8.5 Challenges

- There are, of course, challenges, as with every modeling approach, and estimation method. Let's look at the key ones affecting the N-gram model, as well as the use of MLE

Sensitivity to the training corpus

- The N-gram model, like many statistical models, is significantly dependent on the training corpus. As a result, the probabilities often encode particular facts about a given training corpus. Besides, the performance of the N-gram model varies with the change in the value of N.
- Moreover, you may have a language task in which you know all the words that can occur, and hence we know the vocabulary size V in advance. The closed vocabulary assumption assumes there are no unknown words, which is unlikely in practical scenarios.

2.8.6 Smoothing

- A notable problem with the MLE approach is sparse data. Meaning, any N-gram that appeared a sufficient number of times might have a reasonable estimate for its probability. But because any corpus is limited, some perfectly acceptable English word sequences are bound to be missing from it.
- As a result of it, the N-gram matrix for any training corpus is bound to have a substantial number of cases of putative "zero probability N-grams"

2.9 N-GRAM SENSITIVITY TO TRAINING CORPUS

- In speech recognition, input may be noisy an this can lead to wrong speech-to text conversions. N-gram models can correct this based on their knowledge of the probabilities.
- Similarly, N-gram models are used in machine translation to produce more natural sentences in the target language.
- Sometimes dictionary lookups will not help while correcting spelling errors.
- For example, in the pharse "in about fifteen minutes", the word 'minutes' is a valid dictionary word but it's incorrect in this context. N-gram models can correct such errors.
- N-gram models are generally at word-level. It has also been used at character level to do stemming, i.e., separate the root word from the suffix.
- By looking at N-gram statistics, we can also classify languages or differentiate between US and UK spellings.
- In general, many NLP applications benefit from N-gram models including part-of-speech tagging, natural language generation, word similarity. Sentiment extraction and predictive text input.

2.9.1 Evaluation of an N-gram Model

- The best way to evaluate a model is to check how well it predicts in end-to-end application testing. This approach is called **extrinsic evaluation**, but it is time consuming and expensive.
- An alternate approach is to define a suitable metric and evaluate independently of the application. This is called **intrinsic evaluation**.

- This does not guarantee application performance but it is a quick first step to check algorithmic performance.
- Now, we discuss N-grams corpus :

2.9.2 N-gram corpus

- An item can refer to anything (letter, digit, syllable, token, word or phonemes, or base pairs according to the application).
- In the field of computational linguistics and probability, an n-gram (sometimes also called Q-gram) is a contiguous sequence of n-items from a given sample of text or speech.

2.9.3 Purpose of n-gram

N-grams of texts are extensively used in **text mining and natural language processing tasks**.

They are basically a set of co-occuring words within a given window and when computing the n-grams, one typically moves one word forward, one can also move X-words forward in more advanced scenarios.

N-gram data

- N-gram are continuous sequences of words or symbols or tokens in a document.
- In technical terms, they can be defined as the neighbouring sequences of items in a document.
- They come into play when we deal with text data in Natural Language Processing (NLP) tasks.
- N-gram viewer displays a graph showing how those phrases have occurred in a corpus of books (e.g. "British English", "English Friction", "French") over the selected years.

2.9.4 Objectives of n-gram models

Given a sequence of N-1 words, an N-gram model predicts the most probable word that might follow the sequence.

It is a probabilistic model that is trained on a corpus of text. Such a model is useful in many NLP applications, including speech recognition, machine translation and predictive text input.

2.9.5 Applications of n-grams

It can be used to implement efficiently and effectively, by using sets of n-grams including spelling error detection and correction, query expansion, information retrieval with serial, inverted and signature files, dictionary look-up, text compression, and language identification.

2.9.6 N-gram language model in AI

An N-gram language model predicts the probability of a given N-gram within any sequence of words in the language. A good N-gram model can predict the next word in the sentence, i.e. the value of p(w/h).

N-grams good for : N-gram models are widely used in probability, communication theory, computational linguistics (e.g. statistical natural language processing), computational biological and data compression.

2.10 UNKNOWN WORDS

Unknown words are a major problem that makes the natural language processing (NLP) impossible to correctly analyse the meaning of the sentence.

Identifying unknown words

Using the mentioned below strategies, we can tackle unfamiliar words :

1. Break it up into pieces,
2. Look for context clues,
3. Make connections to other words that are heard.
4. Make connections to our own life.
5. Make a guess even if it is wrong.
6. Look it up in a dictionary.

2.10.1 Out-of vocabulary (OOV) words

- Out of vocabulary (OOV) are terms that are not part of the normal lexicon found in a natural language processing environment. In speech recognition, it is the audio signal that contains these terms.
- Word vectors are the mathematical equivalent of word meaning.
- There are many techniques to handle out-of-vocabulary words.

- Typically a special out of vocabulary token is added to the language model.
- Often the first word in the document is treated as the out of vocabulary word, occurs somewhere in the training data and gets a positive probability.

2.10.2 To Identify a Word

There are six ways to identify words during the act of reading :

1. Context clues (semantics)
2. Word order and grammar (syntax)
3. Word parts or analysing words
4. Morphemic analysis (prefixes, suffixes, and root words)
5. Sight words
6. Phonics

2.10.3 Word Identification Strategies

There are generally four identification strategies, they are :

(1) decoding (2) analogising,

(3) predicting, (4) recognising whole words by sight

Word identification is the ability to accurately and automatically identify sight words and apply decoding strategies to read unfamiliar words.

Stages of word recognition

- A stage model for the early acquisition of reading is proposed with the following ordered sequence of steps
- Pseudo-reading, logographic-visual, alphabetic-phonemi, and orthographic-morphemic.

Five stages of literary development

The five stages of literacy development are :

1. Emergent literacy 2. Alphabetic fluency,
3. Words and patterns 4. Intermediate reading
5. Advanced reading

2.10.4 Word Recognition Model

The four-part processing model for word recognition is a simplified model that illustrates how the brain reads or recognises the words. It illustrates that there are four processes that are active in the reading :

(1) brain including phonological

(2) orthographic

(3) meaning and

(4) context-processing

2.10.5 Open Versus Closed Vocabulary Tasks Open Vocabulary

- Open vocabulary reveals more specific and concrete patterns across a broad range of content domains, better address ambiguous word senses, and are less prone to misinterpretation, suggesting that they are well-suited for capturing the nuances of everyday psychological processes.

Closed vocabulary

Closed-vocabulary approaches provide a way to study the fundamental patterns of how people think and feel.

To analyse vocabulary words : Analysing vocabulary in literature

1. Use prior knowledge, context clues and word structure to construct meaning.
2. Define connotation and denotation.
3. Determine the meaning of words using content.
4. To show how to understand words by their relationship.
5. Interpret literacy meaning and Fig. ** of speech in content.

2.10.6 Difference between Open and Closed Class Words

Sr. No.	Open word class	Closed word class
1.	An open class is one that commonly accepts the addition of new words. It includes nouns, verbs and adjectives.	A closed class is one in which new items are rarely added. It includes pronouns and conjunctions

Sr. No.	Open word class	Closed word class
2.	Open classes normally contain large number of words.	Closed classes are much smaller.
3.	The open-class distinction is related to the distinction between lexical and functional categories.	Some authors consider the distinction identical.
4.	Open classes are generally lexical categories in the strict sense, containing words with greater semantic content.	Closed classes are normally functional categories, consisting of words that perform grammatical functions.
5.	Words are added to open classes through such processes as compounding, derivation, coining and borrowing. When a new word is added through some such process, it can subsequently be used. The open or closed status of word classes varies between languages	A closed class may obtain new items through these same processes, but such changes are much rarer and take much more time. A closed class may obtain new items through these same processes, but such charges, are rarer and take much more time. A closed class is normally seen on part of the core language and is not expected to change.

2.10.7 Evaluating Language Models

If we are given a corpus of text and want to compare two different n-gram models, we divide the data into training and test. Sets, train the parameters of both models on the training set, and then compare how well the train models fit the test set.

The test set : Whichever model is assigned a **higher probability** to the test set meaning it more accurately predicts the test set – is a better model.

Given two probabilistic models, the better model is the one that has a tighter fit to the test data or that better predicts are details of the test data, and hence will assign a higher probability to the test data.

2.11 EVALUATING N-GRAMS : PERPLEXITY

- A common metric is to use **perplexity**, often written as **PP**. Given a test set

  $w = w_1 w_2 ..... w_n$,

$$PP(w) = p(w_1 w_2 ..... w_n)^{\frac{-1}{N}}$$

- Because of inverse relationship with probability, minimising perplexity implies maximising the test set probability.
- Perplexity can also be related to the concept of entropy in information theory.
- It is important in any N-gram model to include markers at start and end of sentences. This ensures that the total probability of the whole language sums to one.
- But all calculations should include the end markers but not the start markers in the count of word tokens.
- In natural language processing, **perplexity** is a way of evaluating language models.
- A language model is a probability distribution over entire sentences or texts.

Perplexity

- Given a language model M, we can use a held-out validation set to compute the perplexity of a sentence.
- The perplexity on a sentence (say) S is defined as :

$$\text{Perplexity (P)} = P(s)^{\frac{-1}{n}}$$

$$= \sqrt[n]{\prod_{k=1}^{h} \frac{1}{P(w_k \mid w_0 w_1 ....w_{k-1})}}$$

- We notice that this s the inverse of the geometric mean of the terms in the product's denominator.
- Since each word has its probability, i.e. conditional probability on the history computed once, we can interpret this as being a **per-word metric**.

This mean that, all else the same, the perplexity is not affected by sentence length.

2.11.1 Perplexity Per Word

- Using the definition of perplexity for a probability model, one finds that the average sentence x_i in the test sample could be coded in 190 bits (i.e., the test sentence had an average log-probability of – 190)
- This gives a vast model perplexity of 2^{190} per sentence.
- But it is common to normalize for sentence length and consider only the number of bits per word.
- If the test sample's sentence comprised a total of 1000 words, and could be coded using a total of 7.95 bits per word, one could report a model of $2^{7.95} = 247$ per word
- The lowest perplexity that is published is about 247 per word, corresponding to a cross-entropy of $\log_2 (247) = 7.95$ bits per word or 1.75 bits per letter using a **trigram** model.

2.11.2 Need of Perplexity in NLP

- In the content of Natural language processing, perplexity is one way to evaluate language models.
- A language model is a probability distribution over sentences. It is both able to generate plausible human-written sentences (if it is a good language model) and to evaluate the goodness of already written sentences.
- A lower perplexity score indicates better generalization performance.
- A lower perplexity indicates that the data are more likely.

Perplexity in Machine-Learning

- In machine learning, the term perplexity has three closely related meanings. Perplexity is a measure of how easy probability distribution is to predict.
- Perplexity is a measure of how variable prediction model is. And perplexity is a measure of prediction error.

Why to Use Perplexity ?

- Generally, perplexity is a state of confusion or a complicated and difficult situation or thing. Technically, perplexity is used for measuring the utility of a language model.
- The language model is to estimate the probability of a sentence or a sequence of words or an upcoming word.

☞ **Is high perplexity good for NLP ?**

- Since predictable results are preferred over randomness, people say that low perplexity is **good** and **high perplexity is bad.**
- Since the perplexity is the exponentiation of the entropy (also we can assume that the concept of perplexity is equivalent to entropy). A language model is a probability distribution over sentences.

2.12 SMOOTHING

- Smoothing is the process of flattening a probability distribution implied by a language model so that all reasonable word sequences can occur with some probability.
- This involves broadening the distribution by redistributing weight from high probability regions to zero probability regions.
- Smoothing not only prevents zero probabilities, it attempts to improve the accuracy of the model as a whole.

☞ **Need of smoothing**

- In a language model, we use parameter estimation (MLE) on training data. We cannot actually evaluate our MLE models on unseen test data because both are likely to contain words/n-grams that these models assign zero probability,
- Relative frequency estimation assigns all probability mass to events that don't occur (unseen events) in the training data.

Example :

Training data : The cow is an animal

Test data : The dog is an animal.

We use unigram model to train :

$$P(\text{the}) = \text{count (the)/(Total number of words in training set)} = \frac{1}{5}$$

$$\text{Likewise, } p(\text{cow}) = p(\text{is}) = p(\text{an}) = p(\text{animal}) = \frac{1}{5}$$

To evaluate (test) the **unigram model :**

$$P(\text{the cow is an animal}) = p(\text{the}) \cdot p(\text{cow}) \cdot p(\text{is}) \cdot p(\text{an}) \cdot p(\text{animal})$$
$$= 0.00032$$

Since, we use unigram model on the test data, it becomes zero because p (dog) = 0

- The term 'dog' never occurred in the training data. Hence, we use smoothing.

2.12.1 Laplace Smoothing

- We use maximum likelihood estimation (MLE) for training the parameters of an N-gram model.
- The problem with MLE is that it assigns zero probability to unknown (unseen) words. It is because, MLE uses a training corpus.
- If the word in the test set is not available in the training set, then the count of that particular word is zero and that leads to zero probability.
- To eliminate this zero probability, we do smoothing.
- Smoothing is about taking some probability mass from the events seen in training and assigns it to unseen events.
- **Add-1 smoothing** (is also called as **Laplace smoothing**) is a simple smoothing technique that Add1 to the count of all n-grams in the training set before normalizing into probabilities.

Example :

Recall that the unigram and bi-gram probabilities for a word w are calculated as follows :

$$P(w) = \frac{C(w)}{N}$$

$$\text{and} \quad p\left(\frac{w_n}{w_{n-1}}\right) = \frac{C(w_{n-1}\, w_n)}{C(w_{n-1})}$$

where, p(w) is the uniform probability, $p(w_{n-1}\, w_n)$ is the bigram probability, C(w) is the count of occurrence of w in the training set, $C(w_{n-1}\, w_n)$ is the count of bigram $(w_{n-1}\, w_n)$ in the training set, N is the total number of word token in the training set.

☞ **Add-1 smoothing for unigrams**

$$P_{\text{Laplace}}(w) = \frac{(C(W) + 1)}{N} + |V|$$

- Where N is the total number of tokens in the training set and |V| is the size of the vocabulary representing the unique set of words in the training set.
- As we have added 1 to the numerator, we have to normalise that by adding the count of unique words with the denominator in order to normalise.

☞ **Add-1 smoothing for bigrams**

$$P_{Laplace}\left(\frac{w_n}{w_{n-1}}\right) = \frac{(C(w_{n-1}w_n)+1)}{C(w_{n-1})+|V|}$$

☞ **Good Turing Smoothing**

- Good Turing smoothing technique uses the frequencies of the count of occurrence of N-Grams for calculating the maximum likelihood estimate.
- For example, consider calculating the probability of a bigram from a corpus. Suppose that bigram has never occurred in the corpus and hence, probability without smoothing would turn out to be zero.
- But as per the good-Turing smoothing, the probability will depend upon the following :
  1. In case, the bigram has never occurred in the corpus (say), the probability will depend upon the number of bigrams which occurred exactly one time and the total number of bigrams.
  2. In case, the bigram has occurred in the corpus (say), the probability will depend upon the number of bigrams which occurred more than one time as the current bigram and the total number of bigram.

The formula is as follows :

For the unknown N-grams, the following formula is used to calculate the probability.

\(p...{unknown}(\frac{W...{i}} · {w ... {i – 1}}) = \ frac{N – 1} {N}\)

In this formula \(N – 1) is count of N-grams which appeared one time and N is count of total number of N-grams.

For the known N-grams, the following formula is used to calculate the probability : \(p (\frac {W ...{i}} · {W ...{i – 1}})

= (\frac {c*} {N}\)

Where c* = \ ((c + 1)\times\frac{N.....{i + 1}} · {N ...{c}}\)

In the above formula, c represents the count of occurrence of n-gram\(N...{c + 1}\)represents count of n-grams which occurred for c + 1 times, \(N ...{c}) represents count of n-grams which occurred for c times and N represents total count for all n-grams.

2.13 SELF-LEARNING TOPICS

☞ **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

$10\ [\log_{10}(1000)] = [\log_{10}(10^3)]$

$= 10\ [3\ \log_{10}(10)]$

$= 10\ (3 \times 1) = 30$ 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.

The normalised edit distance is not defined in terms of edit operations but rather in terms of the edit path.

(6) Edit distance is usually defined as a parametrisable metric. It is calculated with a specific set of allowed edit operations, and each operation is assigned a cost.

(7) Edit distance in NLP

Edit Distance is a measurement of how many changes we must do to one string to transform it into the string we are comparing it to. For example, the difference between 'Frederic' and 'Fred' is four, as we can change 'frederic' into 'fred' wit the deletion of letters 'e', 'r', 'i', and 'c'.

Chapter Ends...

□□□

MODULE 3

CHAPTER 3 Syntax Analysis

Syllabus

3.1 Part-Of-Speech tagging (POS); Tag set for English (Upenn Treebank); Difficultbes /Challenges in POS tagging; Rule-based, Stochastic and Transformation-based tagging. Generative Model: Hidden Markov Model (HMM Viterbi) for POS tagging; Issues in HMM POS tagging; Discriminative Model: Maximum Entropy model, Conditional random Field (CRF);Parsers: Top down and Bottom up; Modelling constituency; Bottom Up Parser: CYK, PCFG (Probabilistic Context Free Grammar), Shift Reduce Parser; Top Down Parser: Early Parser, Predictive Parser

3.2 Self-Learning topics : Evaluating parsers, Parsers based language modelling, Regional languages POS tree banks

3.1 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.

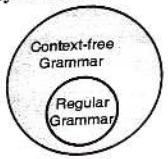


Fig. 3.1.1

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

3.1.1 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** (Σ) : 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.

3.1.2 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.

3.1.3 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.

3.1.3.1 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.

3.1.4 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.

3.1.4.1 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.

3.1.5 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.

3.1.5.1 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.

3.1.5.2 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.

3.1.5.3 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.

3.2 TAG SET FOR ENGLISH (UPENN TREEBANK)

- A tagset is a list of part-of-speech tags, i.e.; labels used to indicate the part of speech and often also other grammatical categories (case, tense, etc.) of each token in a text corpus.

- **The English Penn Treebank tagset** is used with English corpora annotated by the Tree Tagger tool. It is developed by Helmut Schmidt in the **TC project**. This version of the target contains modification developed by sketch engine.

3.2.1 English Penn Treebank Target

Table 3.2.1

POS tag	Description	Example
CC	Coordinating conjunction	and
CD	Cardinal number	1, third
DT	Determiner	the
EX	Existential there	there is
FW	foreign word	les
IN	Preposition, sub-ordinating conjunction	in, of, like
IN/that	that as subordinator	that
jj	adjective	green
jjR	adjective, comparative	greener
jjs	adjective, superlative	greenest
LS	list marker	1)
MD	Modal	could, will
NN	noun, singular or mass	table
NNS	noun plural	tables
NP	Proper noun	Ramesh
NPS	Proper noun plural	Vikings
PDT	predeterminer	both the boys
POS	Possessive ending	Friend's
PP	Personal pronoun	i, he, it
PPZ	Possessive pronoun	my, his
RB	adverb	However, usually, naturally, here, good
RBR	adverb, comparative	better

POS tag	Description	Example
RBS	adverb, superlative	best
RP	particle	give up
SENT	Sentence break punctuation	!?
SYM	Symbol	/ [= *
TO	infinitive 'to'	togo
UH	interjection	uhhuhhuhh
VB	Verb be, past tense	was, were
VBG	Verb be, gerund/present participle	being
VBN	Verb be, past participle	been
VBP	Verb be, sing-present non-3d	am, are
VBZ	Verb be, 3rd person sing. present	is
VH	Verb have, base form	have
VHD	Verb have, past tense	had
VHG	Verb have, gerund/present participle	having
VHN	Verb have, past participle	had
VHP	Verb have, sing. present, non-3d	have
VHZ	Verb have, 3rd person singular present	has
VV	Verb, base form	take
VVD	Verb, past tense	took
VVG	Verb, gerund/present participle	taking
VVN	Verb, past participle	taken
VVP	Verb, sing., present non-3d	take
WZ	Verb, 3rd person sing present	takes
WDT	Wh-determiner	Which
WP	Wh-pronoun	Who, what
WPS	Possessive wh-pronoun	Whose

POS tag	Description	Example
WRB	Wh-adverb	Where, when
#	#	#
$	$	$
"	Quotation marks	"
"	Opening quotion marks	"
(	Opening brackets	((
)	comma	,
:	Punctuation	= ; ; =

3.2.2 Difficulties/Challenges in POS Tagging

We mention below the difficulties and challenges in POS tagging

(i) The main problem with POS tagging is ambiguity.

(ii) In English, many common words have multiple meanings and hence multiple POS.

(iii) The job of a POS tagger is to resolve this ambiguity accurately based on the context of use. For example, the word 'shot' can be a noun or verb.

(iv) If a POS tagger gives poor accuracy, then this has an adverse effect on other tasks that follow. This is called as **downstream error propagation**.

To improve accuracy, POS tagging is combined with other processing.

For example, joint POS tagging and dependency parsing is an approach to improve accuracy compared to independent modelling.

(v) Sometimes a word on its own can give useful clues. For example, 'the' is a determiner. Prefix 'un' suggests an adjective, such as 'unfathomable'. Suffix '-ly' suggests adverb, such as 'importantly'. Capitalisation can suggest proper noun, such as 'Angeles'.

(vi) A word can be tagged, it depends upon the neighbouring words and the possible tags that those words can have.

Word probabilities also play a part in selecting the right tag to resolve ambiguity. For example, 'man' is rarely used as a verb and mostly used as a noun.

(vii) In a statistical approach, one can count tag frequencies of words in a tagged corpus and then assign the most probable tag. This is called **unigram tagging**.

- A much better approach is bigram tagging. This counts the tag frequency given a particular preceding tag.
- Hence, a tag is seen to have dependence on previous tag. We can generalize this to **n-gram tagging**.

3.3 GENERATIVE MODEL

- People throughout the world speak so many different languages, but a computer system or any other computerised machine only understands a single language i.e. binary language (i.e. 1's and 0s).
- This system or a process that converts human language to computer understandable language is known as **Natural Language processing (NLP)**.
- Various diversified models have been suggested so far but a generative predictive model which can optimize depending upon the nature of the problem is still not found. It is still an area of research under work.
- The generative model is a single platform for diversified area of NLP that can address specific problems relating to read text, hear speech, interpret it, measure sentiment and determine which parts are important. This is achieved by process of elimination once the relevant components are identified.
- Single platform provides same model generating and reproducing optimized solutions and addressing different issues.

Remarks

(i) A generative model could generate new photos of animals that look like real animals.

(ii) Given a set of data instances X and a set of labels Y, **Generative** models capture the joint probability P (X, Y) or just P (X), if there are no labels.

(iii) A generative model includes the distribution of the data itself, and tells how likely a given example is.

For example, models that predict the next word in a sequence are typically generative models because they can assign a probability to a sequence of words.

3.3.1 Hidden Markov Model (HMM Viterbi) for POS Tagging

- Hidden Markov Models (HMMS) are a type of statistical modeling which are applied in different fields such as medicine, computer science and data science.

- The Hidden Markov Model (HMM) is the foundation of many modern-day data science algorithms.
- It is used in data science to make efficient use of observations for successful predictions or decision-making processes.

3.3.2 Markov Models

- Markov models are used to predict the future state based on the current hidden or observed states.
- Markov model is a finite-state machine where each state has an associated probability of in any other state after one step. They can be used to model real-world problems where hidden and observable states are involved.
- Markov models can be classified into hidden and observable based on type of information available to use for making predictions or decisions.
- Hidden Markov models deal with hidden variables that cannot be directly observed but only inferred from other observations.
- But in observable model which are also termed as Markov chain, hidden variables are not involved.
- To make the concept of Markov models clear, we consider an example.
- Suppose a bag contains four marbles, two red marbles and two blue marbles.
- We randomly select a marble from the bag, note its colour, and then put it back in the bag. After repeating this pattern several times, we begin to notice a pattern : The probability of selecting a red marble is always two out of four, or 50 %. The reason is : the probability of selecting a particular color of marble is determined by the number of that colour in the bag.
- In other words, the past history (i.e., the contents of the bag) determines the future state (i.e., the probability of selecting a particular colour of marble).

3.3.3 Markov Chain

- Markov chains have the Markov property, and it states that the probability of moving to any particular state next depends only on the current state and not on the previous states.
- Markov chain consists of three main components :
  (i) Initial probability distribution
  (ii) One or more states
  (iii) Transition probability distribution.

- The Fig. 3.3.1 as shown, represents a Markov chain where there are three states and they represent the weather of the day (cloudy, rainy, sunny).

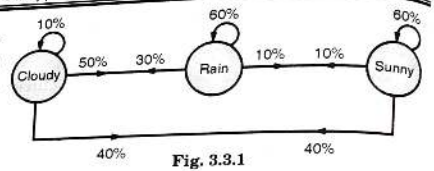


Fig. 3.3.1

- The transition probabilities representing the weather of the next day given the weather of the current day :
- There are three different states; cloudy, rain and sunny. We mention the transition probabilities based on the above diagram :

(i) If sunny today, then tomorrow :

(a) 50 % probability for sunny. (b) 10 % probability for rainy.

(c) 40 % probability for cloudy.

(ii) If rainy today, then tomorrow :

(a) 10 % probability for sunny. (b) 60 % probability for rainy.

(c) 30 % probability for cloudy.

(iii) If cloudy today, then tomorrow :

(a) 40 % probability for sunny. (b) 50 % probability for rainy.

(c) 10 % probability for cloudy.

- Using this Markov chain, we want to conclude : What is the probability that Wednesday will be cloudy if Monday is Sunny.
- We calculate the transitions :

(i) Sunny – Sunny (Tuesday) – Cloudy (Wednesday) :

The probability to a cloudy Wednesday is $(0.5 \times 0.4 = 0.2)$

(ii) Sunny – Rainy (Tuesday) – Cloudy (Wednesday) :

The probability of a cloudy Wednesday is $(0.1 \times 0.3) = 0.03$

(iii) Sunny – Cloudy (Tuesday) – Cloudy (Wednesday) :

The probability of a cloudy Wednesday is $0.4 \times 0.1 = 0.04$

$\therefore$ The total probability of a cloudy Wednesday = $0.2 + 0.03 + 0.04 = 0.27$

- So, here we have determined the probability of a particular state knowing the probabilities of previous states.

3.3.4 Hidden Markov Models (HMM)

GQ. What are Hidden Markov Models (HMM) ?

- The hidden Markov model (HMM) is another type of Markov model where there are a few states hidden.
- It is a hidden variable model which can give an observation of another hidden state using Markov assumption.
- The hidden state is the variable which cannot be directly observed but can be inferred by observing one or more states using Markov assumption.
- Markov assumption is the assumption that a hidden variable is dependent only on the previous hidden state.
- A Markov model is made up of two components : the state transition and hidden random variables that are conditioned on each other.
- A hidden Markov model consists of five important components :

(a) Initial probability distribution,

(b) One or more hidden states,

(c) Transition probability distribution :

The transition matrix is used to show the hidden state to hidden state transition probabilities.

(d) A sequence of observations.

(e) Emission probabilities : A sequence of observations likelihoods, also called as emission probabilities. Each observation expresses the probability of an observation, generated from a state.

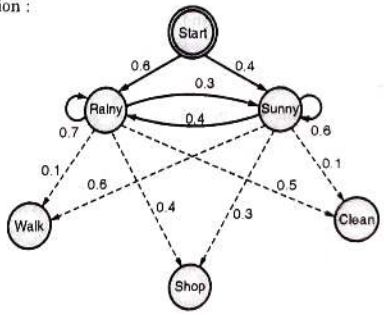


Fig. 3.3.2

- The emission probability is used to define the hidden variable in terms of its next hidden state.

- We mention Markov model representation pictorially :
- The above hidden Markov model predicts whether someone will be walking, shopping or cleaning on a particular day. That will depend upon whether the day is rainy or sunny.
- From the Fig. 3.3.2, we note the following points :
  (i) There are two hidden states such as rainy and sunny. These are hidden states because the process output is whether the person is shopping, walking or cleaning.
  (ii) The sequence of operations is shop, walk and clean.
  (iii) An initial probability distribution is start probability.
  (iv) Transition probability is transition of one state (rainy or sunny) to another state given the current state.
  (v) Emission probability is the probability of observing the output.

3.3.5 Optimising HMM with Viterbi Algorithm

- The Viterbi algorithm is a dynamic programming algorithm for finding the most likely sequence of hidden states. It is called as viterbi path. It results in a sequence of observed events. And these events are in the context of Markov information sources and Hidden Markov Models (HMM).
- We consider an example and apply the Viterbi algorithm to it. Will can spot Mary.

Will as a model

Can as a verb

Spot as a noun

Mary as a noun.

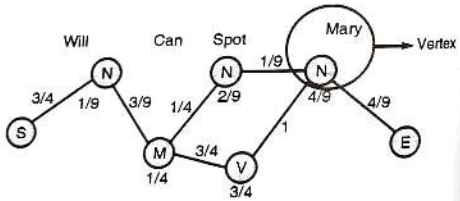


Fig. 3.3.3

Consider the vertex encircled in the above example. There are two paths leading to this vertex as shown below. We mention also the probabilities of the two paths.

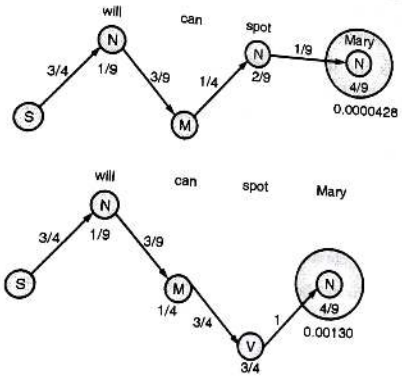


Fig. 3.3.4

- To obtain the optimal path, we begin from the end and trace backward, since each state has only one incoming edge. This gives as a path as shown below :

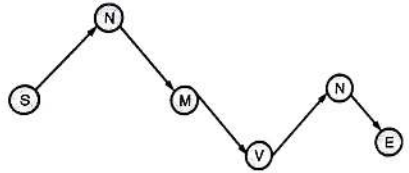


Fig. 3.3.5

- Thus the model can successfully tag the words with their appropriate POS tags.

3.4 ISSUES IN HMM POS TAGGING

(i) The main problem with HMM POS Tagging is **ambiguity.**

(ii) The POS tagging is based on the probability of tag occurring.

(iii) There is no probability for the words that do not exist in the corpus.

(iv) It uses different testing corpus, other than training corpus.

(v) It is the simplest POS tagging, since it chooses most frequent tags associated with a word in training corpus.

(vi) An HMM model is the doubly-embedded stochastic model, where the underlying stochastic process is hidden.

(vii) The hidden stochastic process can only be observed through another set of stochastic processes that produces the sequence of observations.

3.4.1 Discriminative Model

- Discriminative models are also called as **conditional models**. They are a class of logistic models used for classification or regression.
- They distinguish decision boundaries through observed data, such as pass/fail, win/lose, alive/dead or healthy/silk.

3.4.2 Maximum Entropy Model

- In many systems, there is a time or state dependency. These systems evolve in time through a sequence of states and current state is influenced by past states.
- For example, there is a high chance of rain today if it had rained yesterday. Other examples include stock prices, DNA sequencing, human speech or words in a sentence.
- It may happen that we have observations but not the states. For example, we have sequence of words but not the corresponding part-of-speech tags.
- In this case, we model the tags as states and use the observed words to predict the most probable sequence of tags. This is exactly what Maximum-Entropy Markov Model (MEMM) does.
- MEMM is a model that makes use of state-time dependencies. It uses predictions of the past and the current observation to make current prediction.

3.4.3 Statistical View of MEMM

- In statistics, a Maximum-Entropy Markov Model (MEMM), or Conditional Markov Model (CMM), is a graphical model for sequence labeling. It combines features of Hidden Markov Models (HMMS) and Maximum Entropy (Max Ent) models.
- An MEMM is a discriminative model that extends a standard **maximum entropy classifier** by assuming that the unknown values to be learnt are connected in a Markov chain instead of being conditionally independent of each other.

- MEMMS find applications in natural language-processing, specifically in part of speech tagging and information extraction.

3.4.4 Model

- Suppose we have a sequence of observations $O_1, O_2, \ldots, O_n$ that we seek to tag with the labels $S_1, S_2, \ldots, S_n$ that maximize the conditional probability $P(S_1, \ldots, S_n \mid O_1, \ldots, O_n)$.
- In MEMM, this probability is factored into Markov transition probabilities. The probability of transitioning to a particular label depends only on the observation at that position and previous position's label.

$$P(S_1, \ldots, S_n \mid O_1, \ldots, O_n) = \prod_{t=1}^{n} P(S_t \mid S_{t-1}, O_t)$$

- Each of these transition probabilities comes from the same general distribution $P(S \mid S', 0)$.
- For each possible label value of the previous label S', the probability of a certain label S is modeled in the same way as a **maximum entropy classifier.**

$$P_{S'}(S \mid 0) = \frac{1}{Z(0, S')} \exp\left(\sum_a \lambda_a f_a(0, S)\right)$$

- Here, the $f_a(0, S)$ are real-valued or categorical feature-functions, and $Z(0, S')$ is a normalization term. It ensures that the distribution sums to one.
- This form for the distribution corresponds to the **maximum entropy probability distribution** satisfying the constraint, that the expectation for the feature is equal to the expectation given the model :

$$E_e\left[f_a(0, S)\right] = E_p\left[f_a(0, S)\right], \text{ for all a.}$$

- The parameter λ_a can be estimated using generalized iterative scaling.
- The optimal state sequences $S_1, \ldots, S_n$ can be found using a very similar **Viterbi algorithm** to the one used for HMMS.
- The dynamic program uses the forward probability :

$$\alpha_{t+1}(S) = \sum_{S' \in S} \alpha_t(S') P_{S'}(S \mid 0_{t+1})$$

3.4.5 Advantages and Disadvantages of HEMMS

(I) Advantages : (Also comparison between HEMM and HMM)

(i) HMM is a **generative model** because words are modelled as observations generated from hidden states.

(ii) MEMM is a discriminative model.

It uses posterior probability P (T | W) directly.

It is a probability of a tag sequence given a word sequence. It discriminates among the possible tag sequences.

(iii) MEMM uses conditional probability, conditioned on previous tag and current word.

HMM uses joint probability for maximising the probability of the word sequence.

(iv) In HMM, for the tag sequence decoding problem, probabilities are obtained by training by training on a text corpus.

- In MEMM, a distribution is made by adding features which can be picked out by training.
- The idea is to select the maximum entropy distribution given the constraints specified by the features.
- MEMM is more flexible. It is because we can add features such as capitalisation, hyphens or word endings, but they are hard to consider in HMM. MEMM allows for diverse non-independent features.

(II) Shortcomings of MEMM

- Once we are in a state or label, the next observation will select one of many transitions leaving that state.
- But the model as a whole have many more transition. If a state has only one outgoing transition, the observation has no influence. In that case, transition scores are normalized on a per state basis.

3.5 CONDITIONAL RANDOM FIELD (CRF)

GQ. Explain CRF with applications.

- Conditional random fields (CRFS) are a class of statistical modeling methods. They are applied in **pattern recognition** and used for **structured prediction**.

- A classifier predicts a label for a single sample without considering 'neighbouring' samples, a CRF can take context into account.
- To achieve this, the predictions are modeled as a graphical model. And they represent the presence of dependencies between the predictions.
- Depending on the application the type of graph is used. In natural language processing, "linear chain" CRFs are popular, because each prediction is dependent only on its immediate neighbours.
- In image processing, the graph connects locations to nearby similar locations to enforce that they receive similar predictions.
- Other examples where CRFs are used : labeling or parsing
  - (i) of sequential data for natural language processing.
  - (ii) for biological sequences
  - (iii) for POS tagging
  - (iv) shallow parsing
  - (v) named entity recognition
  - (vi) gene finding.
  - (vii) peptide critical functional region finding,
  - (viii) object recognition,
  - (ix) image segmentation in computer vision.

3.5.1 Describe Undirected Probabilistic Graphical Method

- CRFs are a type of discriminative undirected probabilistic graphical model.
- We define a CRF on observations X and random variables Y as follows :

  Let G = (V, E) be a graph such that

  $Y = (Y_v)_{v \in V}$, so that Y is indexed by the vertices of G.

- Then (X, Y) is a **conditional random field**, when each random variable Y_v, conditioned on X, obeys the Markov property with respect to the graph; that is, its probability is dependent only on its neighbours in G_1 :

$$P(Y_v \mid X, \{Y_w : W \neq V\}) = P(Y_v \mid X, \{Y_w : W \sim V\}),$$

Where W ~ V mean that W and V are neighbours in G.

- It implies that a CRF is an undirected graphical model whose nodes can be divided into exactly two disjoint sets X and Y, the observed and output variables, respectively. Then the conditional distribution P (Y | X) can be modeled.

3.5.2 Inference for CRF

For general graphs, the problem of exact inference in CRFs is not possible. But, there exists special cases for which exact inference is feasible.

(i) If the graph is a chain or a tree, message passing algorithms yield exact solutions.

The algorithms that are used in these cases are similar to the **forward, backward** and **Viterbi algorithm** for the case of HMMS.

(ii) If the CRF only contains pair-wise potentials and the energy is **submodular**, combinatorial min cut/max flow algorithms yield exact solutions.

If exact inference is not possible, several algorithms can be used to obtain approximate solutions. These are :

(i) Loopy belief propagation,

(ii) Alpha expansion

(iii) Mean field inference

(iv) Linear programming relaxations.

3.5.3 Parameter Learning

- Learning the parameter θ is generally done by **maximum likelihood learning** for $P(Y_i \mid X_i ; \theta)$.
- If all nodes have exponential family distributions and all nodes are obtained during training, this **optimisation is convex**. It can be solved using gradient descent algorithms.
- But if some variables are unobserved, the inference problem has to be solved for these variables. Exact inference is not possible in general graphs hence approximations have to be used.

3.5.4 Examples on CRF

- In sequence modeling, a chin graph is used. An input sequence of observed variables X represents a sequence of observations and Y represents a hidden state variable.

- The Y_i is structured to form a chain, with an edge between each Y_{i-1} and Y_i. The layout admits efficient algorithms for :
  (i) model training, learning the conditional distributions between the Y_i and feature functions from some corpus of training data.
  (ii) decoding, determining the probability of a given label sequence Y given X.
  (iii) inference, determining the most likely label sequence Y given X.
- The conditional dependency of each Y_i on X is defined through a fixed set of feature functions of the form f (i, Y_{i-1}, Y_i, X), and is measurements on the input sequence that partially determine the likelihood of each possible valve for Y_i.
- The model assigns each feature a numerical weight and combines them to determine the probability of a certain value for Y.

3.6 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.

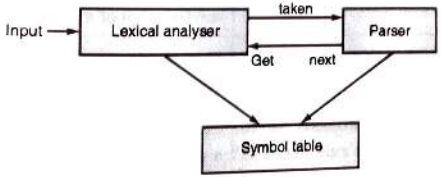


Fig. 3.6.1

- 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'**.

3.6.1 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.

3.6.2 Modelling Constituency

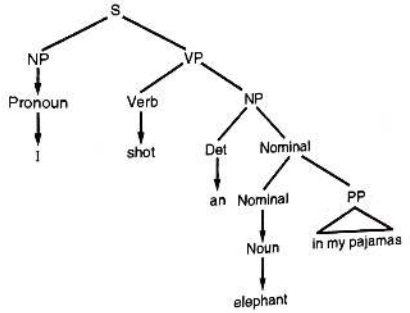
Knowledge of language is the doorway to wisdom.

- 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.

3.6.3 Constituency Parsing

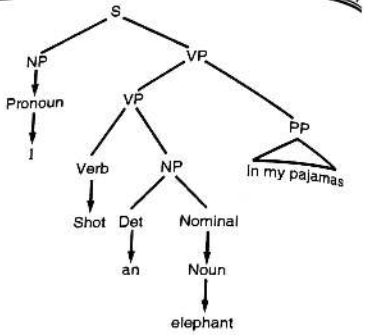
- Constituency Parsing is the process of analyzing the sentences by breaking down it into sub-phrases also known as constituents.
- These sub-phrases belong to a specific category of grammar like NP (noun phrase) and VP (verb phrase).
- Constituency parsing is based on context-free grammars. Constituency 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. 3.6.2(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.
- VP stands for Verb-Phrases, and NP stands for Noun Phrases.



(II)

Fig. 3.6.2

3.6.4 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."

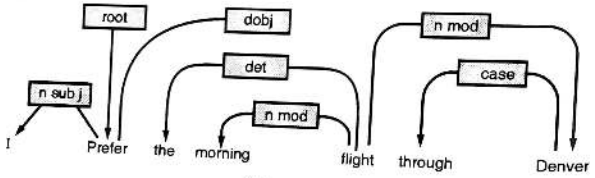


Fig. 3.6.3

- 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.

3.6.5 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 !

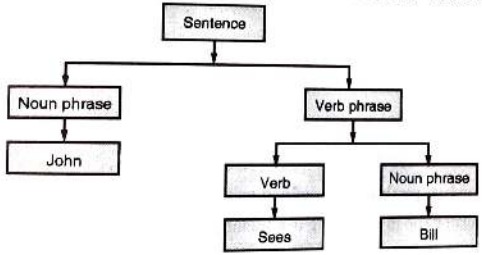


Fig. 3.6.4

- 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 :

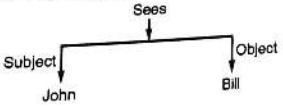


Fig. 3.6.5

- 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.

3.7 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.

3.7.1 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).

3.7.2 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]

3.7.3 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.

3.7.4 How CYK Algorithm Works ?

- For a string of length N, construct a table T of size N × 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

3.8 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.

3.8.1 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.

3.8.2 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.

3.8.3 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.

3.8.4 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.

3.8.5 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.

3.8.6 How PCFG Resolves Ambiguity ?

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

3.8.7 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.

3.8.8 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.

3.8.9 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.

3.9 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.

3.9.1 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.

3.9.2 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.

3.9.3 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.

3.9.4 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.

3.9.5 Example

Ex. 3.9.1 : Consider the grammar

$E \rightarrow 2\ E2$

$E \rightarrow 3\ E3$

$E \rightarrow 4$

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

3.10 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 :	$\theta(n^3)$

3.10.1 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.

3.10.2 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.

3.10.3 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.

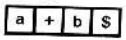
3.11 PREDICTIVE PARSER

- 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.

3.11.1 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 :



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 $.

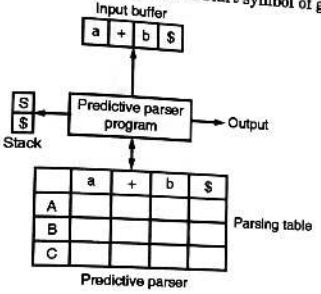


Fig. 3.11.1

(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.

3.11.2 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.

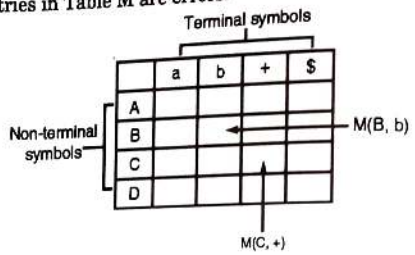


Fig. 3.11.2

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.

3.11.3 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.

3.11.4 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.

3.11.5 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.

3.11.6 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.

3.11.7 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.

3.12 SELF-LEARNING TOPICS

3.12.1 Evaluating Parsers

- Parsers vary in the types of mistakes they make, types of texts they are good at parsing, and speed-and they have all kinds of interesting features in each implementation.
- Here we describe approaches to parser evaluation and use them the analyse the common parser fail cases.

3.12.2 Parser Evaluation Metrics

- To evaluate quality of syntactic parsing properly, we need to choose an appropriate metric.
- To find a metric that will satisfy our requirements, we mention below the known approaches :

(a) Leaf-ancestor evaluation

- The Leaf-Ancestor Evaluation is conducted by assigning a score to every word in a sentence.
- The lineage of each word in the output tree (i.e. the sequence of nodes one has to pass to get to the word, starting from the root) is compared to the lineage of the same word in the gold tree, and the score is calculated as the minimum edit distance.

- Precision is obtained by dividing the sum of the scores by the number of nodes in the lineage of the output tree.

(b) Cross-bracketing

- This metric takes bracketed representations of trees, aligns them according to the leaf nodes, and counts the brackets that do not coincide. It shows the nodes that differ in span.
- The average number of crossed brackets per sentence is also calculated.

(c) Minimum tree edit distance

- The minimum edit distance between trees represents the number of transformations needed to turn the generated tree into the gold one.
- The average number of transformations per tree is also calculated. Transformations include adding, removing and renaming nodes.

3.12.3 Why to Evaluate Parsers ?

- A parser is a compiler or interpreter component that **breaks data into smaller elements** into easy translation into another language.
- A parser takes inputs in the form of a sequence of tokens, interactive commands, or program instructions and breaks them up into parts that can be used by other components in programming.

3.12.4 Parsing and its Techniques Learning

- Parsing is known as **syntax Analysis**. It contains arranging the tokens as source code into grammatical phases that are used by the compiler to synthesis output.
- Generally grammatical phases of the source code are defined by parse tree. There are various types of parsing techniques such as Top-Down-Parser etc.

3.12.5 Parser-based Language Modelling

- Parser-based language modeling (PBLM) is the use of various probabilistic techniques to determine the probability of a given sequence of words occurring in a sentence.
- Language models (LM) analyse bodies of text data to provide a basis for their word predictions.

3.12.6 How Language Modelling Functions

- Language models determine word probability by analyzing text data.
- There are several different probabilistic approaches to modeling language, which vary depending on the purpose of the language model.
- From a technical perspective, the various types differ by the amount of text data they analyse and mathematics they use to analyse it.

Some common statistical language modeling types are :

(i) **N-gram :** N-grams are relatively simple approach to language models. They create a probability distribution for a sequence of n. The n can be any number, and defines the size of the gram, or a sequence of words being assigned a probability. Basically, n can be thought of as the amount of context the model is told to consider.

Some types of n-grams are unigrams, bigrams, trigrams and so on.

(ii) **Unigram :** The unigram is the simplest type of language model. It evaluates each term or word independently. Unigram models handle language processing tasks such as information retrieval.

The unigram is the foundation of a more specific model variant called as the query likelihood model, which uses information retrieval to examine a pool of documents and match the most relevant one to a specific query.

(iii) **Bidirectional :** Unlike n-gram models, which analyse text in one direction (backwards), bidirectional models analyse text in both directions, backwards and forwards.

- These models can predict any word in a sentence or body of text by using every other word in the text.
- Examining text bidirectionally increases result-accuracy. This type is often utilized in machine learning, and speech generation applications.
- For example, Google uses a bidirectional model to process search queries.

(iv) **Exponential :** It is also known as maximum entropy models. It is more complex than n-grams.

- The model evaluates text using an equation that combines feature functions and n-grams.
- Basically this type specifies features and parameters of the desired results.
- The model is based on the principle of entropy, which states that the probability distribution with the most entropy is the best choice.

- Exponential models are designed to maximize cross-entropy, which minimizes the amount of statistical assumptions that can be made.
- This enables the users to better trust the results they get from these models.

3.12.7 Regional Languages POS Tree Banks

- The new languages to be introduced in the POS terminals are Bengali, Gujarati, Kannada, Malyalam, Marathi, Punjabi, Tamil and Telugu.
- POS terminals are expected to go multilingual by the next quarter in a step aimed at localizing cashless payments.
- At present, POS machines are generally available in English or Hindi.
- Arvind Pani, cofounder and CEO of Reverie Language Technologies said,
- "We are working to integrate around 7-8 Indian languages in the POS machines- both stand-alone POS and mobile POS (mPOS)."
- He continued, "We are working on the technology and will tentatively be rolling it out by the next quarter. The time taken actually depends on the technology development and then getting multiple entities on board and conducting pilots."
- The recent success of the Bharat-Interface for Money, or BHIM app (available in seven regional languages for Android version) is an indicator that users are receptive to the idea of paying digitally in regional languages.
- The approaches launched by the Prime Minister has been downloaded a record 18 million times.
- He continued, "Implementing language through technology is complex in nature, therefore we have witnessed a huge gap in terms of awareness about local language experience.
- The interest of local language in the payment space has increased because it gets more users/people to ambit of cashless payments.
- Currently, there are two companies that we are talking to very actively but it is at a nascent stage now.
- One of the companies to adopt this technology is a merchant payment solutions firm, India Transact Services Ltd.
- The idea is to make the payment experience more user friendly. Receiving notifications about payments, answer to customer related queries, cash backs and other such related services in local languages will be helpful in attracting new first time card users."

Chapter Ends...

□□□

NLP Module 4 - Mumbai University revised schemes C 2019

Natural language Processing (University of Mumbai)

MODULE 4

CHAPTER 4

Semantic Analysis

Syllabus

4.1 Introduction, meaning representation; Lexical Semantics; Corpus study; Study of Various language dictionaries like WorldNet, Babelnet; Relations among lexemes & their senses - Homonymy, Polysemy, Synonymy, Hyponymy; Semantic Ambiguity;

Word Sense Disambiguation (WSD); Knowledge based approach(Lesk's Algorithm), Supervised (Naïve Bayes, Decision List),Introduction to Semi-supervised method (Yarowsky) Unsupervised (Hyperlex)

4.2 Self-Learning topics : Dictionaries for regional languages, Distributional Semantics, Topic Models

4.1 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.

4.1.1 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.

4.1.2 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.

4.1.3 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.

4.1.4 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.

4.2 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.

4.2.1 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.

4.2.2 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

4.2.3 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.

4.3 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.

4.4 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.

4.4.1 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.

4.4.2 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.

4.4.3 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.

4.4.4 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.

4.5 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.

4.5.1 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.

4.5.2 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.

4.5.3 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.

4.5.4 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'.

4.6 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.

4.6.1 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.

4.6.2 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).**

4.6.3 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.

4.6.4 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.

4.7 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.

4.7.1 BabelNet

Stable release : BabelNet 5.0 / February 2021

Operating system : Virtuoso Universal Server

Lucene

Type : Multilingual encyclopedic dictionary Linked data

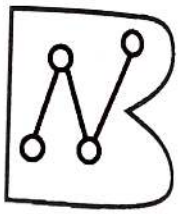


Fig. 4.7.1 : BabelNet

License : Attribution non-commercial share A like 3.0 unported

Website : babelnet.org

4.7.2 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.

4.7.3 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.

4.8 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.

4.8.1 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'.

4.8.2 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).

4.9 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.

4.9.1 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.

4.9.2 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.

4.9.3 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.

4.10 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.

4.10.1 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.

4.10.2 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.

4.10.3 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.

4.11 SUPERVISED NAIVE BAYES

- Naive Bayes Classifier Algorithm is a family of probabilistic algorithms. It is based on applying Bayes theorem with the assumption of conditional independence between every pair of a feature.
- Bayes theorem calculates probability $P\left(\frac{c}{x}\right)$ where c is the class of the possible outcomes and x is the given instance. And it represents some certain features.

$$P\left(\frac{c}{x}\right) = \frac{P\left(\frac{x}{c}\right) \cdot P(c)}{P(x)}$$

- Naive Bayes are mostly used in Natural Language Processing (NLP) problems.
- Naive Bayes predict the tag of a text. They calculate the probability of each tag for a given text and then output the tag with the highest one.

4.11.1 Method of Naive Bayes Working

- We consider an example, we classify the review whether it is positive or negative.

Training Dataset

Text	Reviews
"I liked the movie"	Positive
"It is a nice movie story is good"	Positive
"Nice songs But end is boring"	Negative
"Hero's acting is 50-50 but heroine looks pretty. Overall nice movie"	Positive
"sad, boring movie."	Negative

- Now, we classify whether the text "overall liked the movie" has positive or a negative review.

We calculate

- P(positive/overall liked the movie) - it is the probability that the tag of a sentence is positive given that the sentence is "overall liked the movie".
- P(negative/overall liked the movie) - it is the probability that the tag of a sentence is negative given that the sentence is "overall liked the movie".

- Now, first we apply Removing Stopwords and stemming in the text :

Removing Stopwards

- These are common words that don't really add anything to the classification, such as an, either, else, ever etc.

Stemming

- Stemming is to take out the root of the word.
- Applying these techniques, our text becomes :

Text

"ilikedthemovie"

"itsagoodmovienicestory"

"nicesongsbutsadlyboringend"

"herosactingissobutheroinelooksprettyovrerallnicestory"

"sadboringmovie"

4.11.2 Feature Engineering

- Finding feature from the date to make machine learning algorithms work, is the important part. Here, we have text and we need to convert this text into numbers. And then we carry on calculations.
- We use word frequencies. Every document is treated as a set of words. And the features will be the conuts of each of these words.
- We have to find which tag has a bigger probability, we need only find :

  P(positive) · P(overall liked the movie/positive)

  and P(negative) · P(overall liked the movie/negative)

  [**Note :** In both the cases, divisor is same].
- Since every word in a sentence is **independent** of the other ones, so we can look at individual words :

$$\therefore P(\text{overall liked the movie}) = P(\text{overall}) \cdot P(\text{liked}) \cdot P(\text{the}) \cdot P(\text{movie})$$

$$\therefore P(\text{overall liked the movie/positive}) = P(\text{overall/positive}) \cdot P(\text{liked/positive}) \cdot P(\text{the/positive}) \cdot P(\text{movie/positive})$$

- Now, we can calculate the probabilities :

(i) For a given sentence in our training data,

$$P(\text{positive}) = \frac{3}{5}$$

$$\therefore P(\text{negative}) = \frac{2}{5}$$

Calculating Probabilities

- Now, calculating p(overall/positive) means counting how many times the word overall appears in positive texts (1) divided by the total number of words in positive (17)

$$\therefore P(\text{overall/positive}) = \frac{1}{17} \; ; \qquad P(\text{liked/positive}) = \frac{1}{17}$$

$$P(\text{the/positive}) = \frac{2}{17} \qquad P(\text{movie/positive}) = \frac{3}{17}$$

- In our case, the total possible words count are 21.
- Applying smoothing, the results are :

Word	P(word/positive)	P(word/negative)
Overall :	$1 + \frac{1}{21}$	$0 + \frac{\frac{1}{7}}{21}$
Liked :	$1 + \frac{\frac{1}{17}}{21}$	$0 + \frac{\frac{1}{7}}{21}$
The :	$2 + \frac{\frac{1}{17}}{21}$	$0 + \frac{\frac{1}{7}}{21}$
Movie :	$3 + \frac{\frac{1}{17}}{21}$	$1 + \frac{\frac{1}{7}}{21}$

- Multiplying the probabilities, we get

P(overall liked the movie) = 0.0000138

P(overall not liked the movie) = 0.0000013

∴ **"Overall liked the movie" has positive tag.**

4.12 YAROWSKY ALGORITHM

- In computational linguistics the yarowsky algorithm is a semi-supervised or unsupervised learning algorithm, for word sense disambiguation.
- From observation, words tend to exhibit only one sense in most given discourse and in a given collocation.

4.12.1 Application

- The algorithm starts with a large, untagged corpus, There it identifies examples of the given polysemous word, and stores all the relevant sentences as lines.
- If there are two possible senses of the word, then the next step is to identify a small number of seed collocations representative of each sense. Then give each sense a label (say, sense A and B), then we assign the appropriate label to all training examples containing the seed collocations.
- The algorithm should initially choose seed collocations representative that will distinguish sense A and B accurately and productively. The can be done by selecting seed words from a dictionary's entry for that sense. The collection tend to have stronger effect if they are adjacent to the target word and the effect weakens with distance.
- Seed words that appear in the most reliable collocational relationships with target word will be selected.
- The effect is much stronger for words in a predicate argument relationship than for arbitrary associations at the come distance to the target word. And it is stronger for collocations with content words than with function words.
- A collocation word can have several collocational relationships with the target word throughout the corpus.
- This gives the word different rankings or even different classifications. It can also be carried out by identifying a single defining collocate for each class.
- A wordnet can be used as an automatic source for such defining terms. Also words that occur near the target word in great frequency can be selected as seed collocations representative. This process is not automatic, one must decide which word will be selected for each target word sense, the output will be reliable indicators of the senses.

- The training algorithm calculates the probability P[sense| collocation], and the decision list is ranked by the log-likelihood ratio :

$$\log\left[\frac{P(\text{sense A} \mid \text{collocation i})}{P(\text{sense B} \mid \text{collocation i})}\right]$$

- The decision-list algorithm resolves many problem in a large set of non-independent evidence source by using only the most reliable piece of evidence rather than the whole matching collocation set.
- The resulting classifier will be applied to the whole sample set.
- As more newly-learned collocations are added to the seed sets, the sense A or sense B set will grow, and the original residual will shrink.
- At the end of each iteration, the mistagged collocations are prevented and the purity of the seed sets is improved.
- In order to avoid strong collocates becoming indicators for the wrong class, the class-inclusion threshold needs to be randomly altered. The algorithm will continue to iterate till no more reliable collocations are found.
- According to Yarowsky, for any sense to be clearly dominate, the occurrences of the target word should not be less than 4.
- When the algorithm converges on a stable residual set, a final decision list of the target word is obtained.
- The most reliable collocation are at the top of the new list. The final decision list is applied to the new data, the collocation with the highest rank in the list is used to classify the data.

4.13 UNSUPERVISED (HYPERLEX)

- Hyperlexia is a syndrome characterised by a child's significantly higher ability to read. It is the ability to read words without prior training in learning to read, typically before the age of five. The children with hyperlexia have a higher word-decoding ability than their reading comprehension levels. Children with hyperlexia also have an intense fascination for written material at a very early age.
- Hyperlexia can be viewed as a superability in which word recognition ability goes far above expected levels of skill.
- Some hyperlexics have trouble in understanding speech. Some expects are of opinion that most children with hyperlexia, lie on the **autism spectrum.**

- Hyperlexic children are often amazed by letters or numbers. They are very good at decoding language and often become very early readers.
- Typical special interests of hyperlexic children include numbers, letters, fonts, foreign alphabets, languages, periodic table, the solar system, logos, geography, anatomy, countries, flags, capital etc.
- The word hyperlexia is derived from Greek terms. 'hyper' means above, beyond and 'lexia' means 'word'.

4.13.1 Development

- Hyperlexic children follow a different developmental path relative to other typical individuals.
- In spite of hyperlexic children's amazing reading ability, they may struggle to **communicate.** They learn to speak by heavy repetition. They may have also difficulty learning the **rules of grammar of the language.**
- Sometimes this results in social problems. Their language may develop using repeating words and sentence.
- Often the child has a large **vocabulary** and can identify many objects and pictures, but cannot put their language skills to good use.
- Hyperlexic children often struggle with Who ? What ? Where ? Why ? and How ? questions.
- A child with hyperlexia often logs behind in social skills. Hyperlexic children have far less interest in playing with other children than their peers.

4.13.2 Types of Hyperlexia

In one paper, parold Treffert proposed three types of hyperlexia. They are :

Type (1) : Neurotypical children who are very early readers.

Type (2) : Autistic children who demonstrate very early reading as a splinter skill.

Type (3) : Very early readers who are not on the autism spectrum, though they exhibit some 'autistic - like' traits and behaviours which gradually fade as the child gets older.

A different paper by Rebecca Williamson brown, proposes only two types of hyperlexia.

Type (1) : Hyperlexia marked by an accompanying language decoder.

Type (2) : Hyperlexia marked by an accompanying visual-spatial learning disorder.

4.13.3 Features of Hyperlexia

Four **features** which consistently describe Hyperlexia :

(1) The presences of an accompanying neurodevelopmental disorder.

(2) Advanced reading skills, relative to comprehension skills, or general intelligence,

(3) An early acquisition of reading skills without explicit teaching,

(4) A strong orientation toward reading.

4.14 SELF-LEARNING TOPICS

4.14.1 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).

4.14.2 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.

4.14.3 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.

Chapter Ends...

❑❑❑

NLP Module 5 - sfghjk

Natural language Processing (University of Mumbai)

MODULE 5

CHAPTER

5

Pragmatic and Discourse Processing

Syllabus

5.1 Discourse : Reference Resolution, Reference Phenomena, Syntactic & Semantic constraint on coherence; Anaphora Resolution using Hobbs and Cantering Algorithm.

5.2 Self-Learning topics : Discourse segmentation, Conference resolution.

5.1 PRAGMATIC ANALYSIS

- A step in the information extraction from text is pragmatic analysis. In particular, it's the section on analysing a group of text structures to determine their true meaning or the intended meaning.
- The study of pragmatics, a branch of linguistics and semiotics, focuses on how context affects meaning.
- In contrast to semantics, which examines meaning that is "coded" or conventional in a particular language, pragmatics investigates how the transmission of meaning depends not only on the speaker's and listener's structural and linguistic knowledge (grammar, lexicon, etc.), but also on the context of the utterance, any prior knowledge about those involved, the speaker's implied intent, and other elements.
- For example, the semantic analysis meaning of the sentence "The Soldier fought like a lion" yields the meaning that the soldier fought using the paws or bit using teeth! Which not the intended meaning. The intended meaning here is to highlight the ferociousness of the soldier.
- Sarcasm identification or detection is one of the important application of pragmatic analysis.

5.2 DISCOURSE ANALYSIS

- Till now our attention has mostly been drawn to linguistic events that take place at the word or sentence level. Of course, language often consists of collocated, connected groupings of sentences rather than single, unrelated utterances. Such a collection of sentences is referred to as a discourse.
- Language in use is what the word discourse in linguistics refers to. Discourse analysis is the practise of analysing texts or languages that entails understanding social interactions and text interpretation. Dealing with morphemes, n-grams, tenses, linguistic features, page layouts, and other elements can be part of discourse analysis. One definition of discourse is a series of sentences.

5.3 REFERENCE RESOLUTION

- The discourse level is rich with occurrences in language. Think about the conversation in the example,

  John went to Bill's car dealership to check out an Acura Integra. He looked at it for about an hour **(5.1)**

- What do pronouns like "he" and "it" mean? The reader might not have issue deducing that he refers to John and not Bill, and that it refers to the Integra and not Bill's auto business, but for a computer?
- The mechanism through which speakers utilise terms like 'John' and 'he' in passage (5.1) to signify a person called 'John' is the subject of this section's consideration of the reference problem. Reference Resolution is a process by which when speakers use expressions like 'John' and 'he' in passage (5.1) to denote a person named 'John' this relation is identified.
- We must first clarify a few terms before we can continue further.

1. Referring Expression and Referent

- A natural language expression used to perform reference is called a referring expression, and the entity that is referred to is called the referent.
- **Example :** The name 'John' and 'he' in passage (5.1) are referring expressions, and 'John' the person is their referent.
- As a convenient shorthand, we will sometimes speak of a referring expression referring to a referent, e.g., we might say that 'he' refers to 'John' the person.

2. Corefer

- If the same entity has been referred by two referring expressions then it is called as Corefer.
- **Example :** The name 'John' and 'he' are corefer in passage (5.1) as both are referring to the person 'John'.

3. Antecedent

- There is also a name for a referring expression that grants permission for the use of another, similar to how mentioning the name 'John' permits 'John' the person to be afterwards referred to by the pronoun 'he'.
- We call the name 'John' the antecedent of 'he'.

4. Anaphora and Anaphoric

- Reference to an entity that has been already introduced into the discourse is called anaphora, and the referring expression used is said to be anaphoric.

Example : In passage (5.1), the pronouns 'he' and 'it' are anaphoric. And, the name 'John' is anaphora.

5. Discourse Model

- Speakers of natural languages have a wide range of methods to refer to entities. Say you wish to make a reference to your friend's Acura Integra car.
- Based on the context of the discourse, you might say 'it', 'this', 'that', 'this car', 'that car', 'the car', 'the Acura', 'the Integra', or 'my friend's car', among many other possibilities. However, you are not free to select any of these options in any situation.
- For instance, if the hearer is unfamiliar with your friend's automobile, has never heard of it before, and is not in the same near vicinity as the conversation participants, you cannot simply state, "It or the Acura" (i.e., the situational context of the discourse has a role to play when deciding upon the referring expression to be used).
- This is due to the fact that each form of referring expression conveys a unique signal about the position that the speaker thinks the referent occupies in the hearer's system of beliefs.
- The hearer's mental representation of the continuing conversation, which we refer to as a discourse model, is made up of a subset of these beliefs that have a particular status.
- The entities that have been mentioned in the conversation as well as the connections they are a part of are represented in the discourse model.
- As a system (or a human hearer) understands a text, it constructs a mental model called a discourse model that includes representations of the entities mentioned in the text as well as information about their characteristics and relationships.
- In the Fig 5.3.1 we claim that a representation for an 'evoked' referent gets accessible into the model when it is first referenced in a speech. This representation is 'accessed' from the model on future mention.

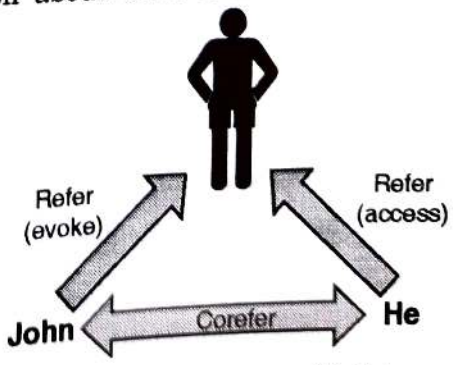


Fig. 5.3.1 : Discourse Model

- **Example :** When the name 'John' is first mentioned in the discourse the referent, the person, 'John' is evoked into the model. and, subsequently when the person in referred with the expression 'he' the representation is being accessed from the model.

5.4 REFERENCE PHENOMENA

- The range of referential phenomena offered by natural languages is quite extensive. Five types of referring expression have been covered here, indefinite noun phrases, definite noun phrases, pronouns, demonstratives, and names.
- Additionally it is important to note the three types of referents which complicate the reference resolution problem, inferrables, discontinuous sets and generics.

5.4.1 Type of Referring Expression

There five types of referring expressions, indefinite noun phrases, definite noun phrases, pronouns, demonstratives, and names.

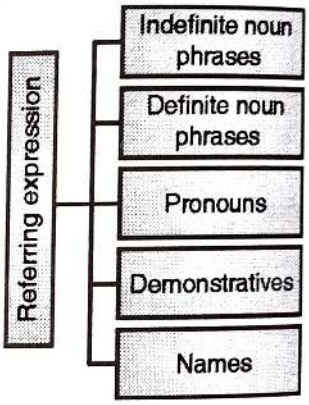


Fig 5.4.1 : Types of Referring Expressions

1. Indefinite Noun Phrases

- Indefinite references bring unfamiliar entities to the conversation environment.
- The determiner 'a' (or 'an') is used to indicate indefinite references most frequently, although it may also indicate by quantifiers like 'some' or even the determiner 'this'.

Example

- I saw 'an' Acura Integra today. **(5.2)**
- 'Some' Acura Integras were being unloaded at the local dealership today. **(5.3)**
- I saw 'this' awesome Acura Integra today. **(5.4)**
- I am going to the dealership to buy 'an' Acura Integra today. **(5.5)**
- These noun phrases elicit a representation for a new entity that fits the discourse model's definition.
- There may be a specific/non-specific ambiguity since the indefinite determiner 'a' doesn't say whether the speaker can identify the object. Example (5.2) only has the specific reading since the speaker is specifically thinking of the Integra that she saw.

Drawback : On the other hand, both interpretations are plausible in phrase (5.5).

- In other words, the speaker may already know which Integra she wants or may simply be preparing to choose one she likes (nonspecific).

- **Solution to the problem :** In some instances, a following referencing statement can be used to distinguish between the readings; if the expression is definite, the reading is particular (I hope they still have that option), and if it is indefinite, the reading is generic (I hope they have a car I like).
- There are several exceptions to this rule, such as when definite expressions are acceptable in particular modal circumstances (I'll park it in my garage), this is compatible with the nonspecific reading.

2. Definite Noun Phrases

An entity that is identifiable to the hearer is referred to as having a "definite reference" when,

(i) it has already been mentioned in the discourse context (and is therefore represented in the discourse model),

(ii) is a part of the hearer's worldview, or

(iii) the description itself implies the object's uniqueness.

Example

- I saw an Acura Integra today. 'The Integra' was white and needed to be washed. (5.6)
- 'The Indianapolis 500' is the most popular car race in the US. (5.7)
- *The fastest car in the Indianapolis 500* was an Integra. (5.8)
- (5.6) showcases an example where the referent may be determined from the conversation context.
- (5.7) showcases an example where the referent is identifiable from the hearer's set of beliefs.
- (5.8) showcases an example where the referent is inherently unique.
- Here it is important to note that the discourse model or the hearer's collection of worldviews must be accessible in order to refer to a definite noun in a sentence. In the second instance, it also evokes a discourse model representation of the referent.

3. Pronouns

- Pronominalization is another type of definite reference, as seen in example (5.9).
- I saw an Acura Integra today. It was white and needed to be washed. **(5.9)**
- Pronominal reference has more restrictions than complete definite noun phrases, necessitating a high level of activation or salience for the referent in the discourse model.

- While definite noun phrases can frequently refer further back, pronouns typically (but not always) refer to entities that were introduced no more than one or two sentences before to the current discourse. This is illustrated by the difference between sentences (5.10d) and (5.10d').

(5.10)

(a) John went to Bob's party, and parked next to a beautiful Acura Integra.

(b) He went inside and talked to Bob for more than an hour.

(c) Bob told him that he recently got engaged.

(d) He also said that he bought 'it' yesterday.

(d') He also said that he bought 'the Acura' yesterday.

- The Integra no longer has the level of salience necessary to permit pronominal reference by the time we get to the last sentence.
- **Cataphora** : In cataphora, pronouns are mentioned before their referents are, as in the following example (5.11). Here, it can be seen that the pronouns 'he' and 'it' both appear before the introduction of their respective referents.
- Before he bought it, John checked over the Integra very carefully. **(5.11)**
- **Bound :** Pronouns can be considered bound when they appear in quantified contexts, as in the following example (5.12). 'Her' operates as a variable bound to the quantified expression every woman when read in the appropriate way, rather than referring to any particular woman in the context.
- Every woman bought her Acura at the local dealership. **(5.12)**

4. Demonstratives

- Simple definite pronouns like 'it' function differently than demonstrative pronouns like 'this' and 'that'. They may show up both by themselves and as determiners, such as 'this Acura' and 'that Acura'.
- The decision between two demonstratives—this signalling proximity and that signalling distance—is typically connected to some idea of geographical proximity. Depending on the situational setting of the conversation participants, spatial distance might be calculated as in (5.13).
- [John shows Bob an Acura Integra and a Mazda Miata]
- Bob (pointing): I like 'this' better than 'that'. **(5.13)**

- As an alternative, the discourse model's conceptual relations might be used to symbolically understand distance. Consider the example (5.14).
- Here, 'this one' relates to the Acura that was purchased yesterday (closer temporal distance), and 'that one' refers to the Acura that was purchased five years ago (greater temporal distance).
- I bought an Integra yesterday. It's similar to the one I bought five years ago. That one was really nice, but I like this one even better. **(5.14)**

5. Names

Names are a prevalent type of referring expression and include names of persons, organisations, and places. In the discourse, names may be used to refer to both new and historical entities.

(5.15)

(a) 'Miss Woodhouse' certainly had not done him justice.

(b) 'International Business Machines' sought patent compensation from Amazon; 'IBM' had previously sued other companies

5.4.2 Types of Referents which Complicate the Reference Resolution

- After describing several referring expression types, we will now focus on a few intriguing referent kinds that make the reference resolution problem more challenging.

1. Inferrables

In certain instances, a referring expression refers to an entity that has been implied rather than one that has been expressly summoned in the text. Such referents are called as Inferrables.

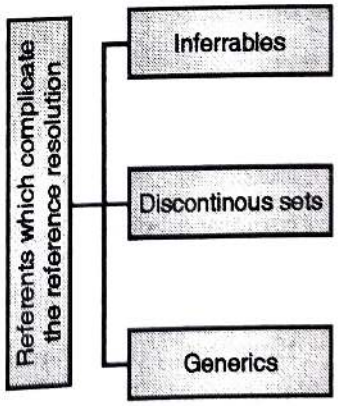


Fig 5.4.2 : Types of Referents which Complicate the Reference Resolution

Example

- I almost bought an Acura Integra today, but 'a door' had a dent and 'the engine' seemed noisy. **(5.16)**

- Now, consider the expressions 'a door' and 'the engine'. In a typical situation, the indefinite noun phrase 'a door' would introduce a new door into the discourse context, but in this instance, the hearer is to infer something additional: that it is not just any door, but rather one of the doors of the Integra.
- The definite noun phrase the engine is also typically used to denote 'the engine' that has already been invoked or is otherwise easily recognised. Although 'the engine' of the aforementioned Integra has not been stated clearly in this sentence, the hearer infers that it is the referent.
- The outcomes of processes that are described by utterances in a discourse can also be specified by inferrables. Take into account the recipe's possible follow-ups (a-c) to sentence (5.17).

(5.17)

Mix the flour, butter, and water.

(a) Kneed 'the dough' until smooth and shiny.

(b) Spread 'the paste' over the blueberries.

(c) Stir 'the batter' until all lumps are gone.

- The words 'the dough' (a solid), 'batter' (a liquid), and 'paste' (something in between) can all be used to refer to the outcome of the acts in the first line, but they all indicate distinct characteristics of this outcome.

2. Discontinuous Sets

- Generally, 'they' and 'them' are examples of a plural referring expression that is sometimes used to refer to groups of items that are invoked collectively. Other examples include another plural referring expression (their Acuras) and conjoined noun phrases (John and Mary), as shown in the following example,
- John and Mary love their Acuras. They drive them all the time. **(5.18)**
- However, plural references may also apply to groups of things that the text's discontinuous phrases have referred to, as in the following example,
- John has an Acura, and Mary has a Mazda. They drive them all the time. **(5.19)**
- Here, 'they' refer to John and Mary, and the Acura and the Mazda are referred to as 'them' in this sentence. Also, take note that the second line in this instance will often be read pairwise or correspondingly, meaning that John drives the Acura and Mary drives the Mazda rather than both of them driving both automobiles.

3. Generics

- The existence of generic reference adds another layer of complexity to the reference problem. Considering one (15.20).
- I saw no less than 6 Acura Integras today. 'They' are the coolest cars. **(5.20)**
- The most obvious reading here relates to the class of Integras as a whole rather than the specific six Integras stated in the first phrase.

5.5 SYNTACTIC AND SEMANTIC CONSTRAINTS ON COREFERENCE

- Any effective reference resolution method must include the step of filtering the collection of potential referents based on a few reasonably rigid requirements. Here, we go through a few of these restrictions.

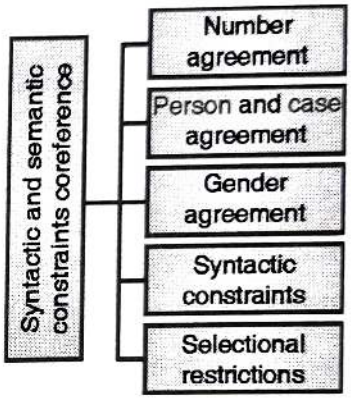


Fig. 5.5.1 : Syntactic and Semantic Constraints on Coreference

1. Number Agreement

- It is necessary for referring statements and their referents to have the same number; in English, this entails differentiating between references to the singular and plural.
- In Table No. 5.5.1, pronouns are categorised according to their number.

Table 5.5.1 : Pronouns according to Number Agreement

Singular	Plural	Unspecified
She, Her, He, Him, His, It	We, Us, They, Them	You

Example

- The following examples illustrate constraints on number agreement.
- John has a new Acura. 'It' is red. **(5.21)**
- John has three new Acuras. 'They' are red. **(5.22)**
- John has a new Acura. 'They' are red. **(5.23)**
- John has three new Acuras. 'It' is red. **(5.24)**
- Here, the examples (5.21) and (5.22) are correct whereas, (5.23) and (5.24) are incorrect statements as the term 'they' should be associated with plural and the term 'it' should be associated with singular. These to statements violate the constraints.

2. Person and Case Agreement

- Three different person tenses are recognised in English: first, second, and third. Also, due to case agreement, various pronoun forms may be necessary when used in the subject position (nominative case, e.g., he, she, or they), object position (accusative case, e.g., him, her, or they), and genitive position (genitive case, e.g., his Acura, her Acura, their Acura).
- Table No. 5.5.2 depicts a division of pronoun categories according to person and according to case agreement.

Table 5.5.2 : Division of Pronoun Categories according to Person and case agreement

	First	Second	Third
Nominative	I, We	You	He, She, They
Accusative	Me, Us	You	Him, Her, Them
Genitive	My, Our	Your	His, Her, Their

Example

- You and I have Acuras. 'We' love them. **(5.25)**
- John and Mary have Acuras. 'They' love them. **(5.26)**
- John and Mary have Acuras. 'We' love them. (Where, We = John and Mary) **(5.27)**
- You and I have Acuras. 'They' love them. (Where, They = You and I) **(5.28)**
- Here, the examples (5.25) and (5.26) follow the person and case agreement constraints whereas, (5.27) and (5.28) do not follow these constraints.

3. Gender Agreement

The gender implied by the referring statement must likewise be accepted by the referents. English third person pronouns differentiate between the genders of male, female, and nonpersonal, although unlike many other languages, the first two only apply to living objects. Table No. 5.5.3 illustrate the pronouns under these gender categories.

Table 5.5.3 : Pronouns under Gender Categories

Masculine	Feminine	Nonpersonal
He, Him, His	She, Her	It

Example

- John has an Acura. 'He' is attractive. (Where, he = John, not the Acura) **(5.29)**
- John has an Acura. 'It' is attractive. (Where, it = the Acura, not John) **(5.30)**

4. Syntactic Constraints

When two referential expressions are used in the same sentence, their syntactic interactions with potential antecedent noun phrases may potentially limit reference relations.

Example

- The pronouns in all of the following sentences are subject to the constraints indicated in brackets.
  - John bought 'himself' a new Acura. (himself = John) **(5.31)**
  - John bought 'him' a new Acura. (him ≠ John) **(5.32)**
  - John said that Bill bought 'him' a new Acura. (him ≠ Bill) **(5.33)**
  - John said that Bill bought 'himself' a new Acura. (himself = Bill) **(5.34)**
  - 'He' said that 'he' bought John a new Acura. (He ≠ John and he ≠ John) **(5.35)**
- Reflexive pronouns in English include 'himself', 'herself', and 'themselves'. A reflexive corefers with the subject of the most immediate sentence that contains it (5.31), but a nonreflexive cannot corefer with this subject, to greatly simplify the issue (5.32). Examples (5.33) and (5.34), in which the reverse reference pattern is evident between the pronoun and the subject of the higher phrase, demonstrate that this rule only applies to the subject of the most immediate clause. John, on the other hand, cannot corefer with either the subject of the most recent sentence or with a higher-level subject (5.35).
- These syntactic restrictions don't just apply to a referring expression and a certain potential antecedent noun phrase; they also forbid coreference between the two, regardless of whether there are any other antecedents that can be used to signify the same item.
- For example, a nonreflexive pronoun like 'him', generally, would be able to corefer with the subject of the preceding sentence as it does in example (5.36), but it is not possible in example (5.37) due to the presence of the coreferential pronoun 'he' in the second clause.
  - John wanted a new car. Bill bought 'him' a new Acura. (him = John) **(5.36)**
  - John wanted a new car. 'He' bought 'him' a new Acura. (he = John, him ≠ John) **(5.37)**

- The criteria discussed above simplify the problem too much in a number of ways, and they do not apply in many situations. In truth, the facts become extremely complex upon closer examination. In fact, it seems implausible that mere syntactic relations could adequately explain all of the evidence.
- For example, even though they appear in the same syntactic arrangements, the reflexive 'himself' and the nonreflexive 'him' in sentences (18.43) and (18.44), respectively, can both relate to the subject John.
  - John set the pamphlets about Acuras next to himself. (himself = John) **(5.38)**
  - John set the pamphlets about Acuras next to him. (him = John) **(5.39)**

5. Selectional Restrictions

- Referents may be dropped due to selectional restrictions that a verb imposes on its arguments, as in the following example (5.40).
- John parked his Acura in the garage. He had driven it around for hours. **(5.40)**
- Here, the term 'it' might be compared to either the Acura or the garage. The direct object of the verb 'drive', however, must be a vehicle that can be driven, such as a car, truck, or bus, and cannot be a garage. As a result, 'the Acura' is the only potential referent because the pronoun appears as the object of drive. It is feasible that a real-world NLP system might have a set of selectional constraints that were very extensive for the verbs in its lexicon.
- However, in the case of metaphors, restrictions on selection can be broken. For example, take a look at (5.41).
- John bought a new Acura. It drinks gasoline like you would not believe. **(5.41)**
- Although the verb 'drink' doesn't frequently relate to inanimate objects, in this context, it can be used to refer to a brand-new Acura.
- Also, more general semantic constraints might also be in force, but they are far more challenging to represent thoroughly. Take for example the passage (5.42).
- John parked his Acura in the garage. It is incredibly messy, with old bike and car parts lying around everywhere. **(5.42)**
- John parked his Acura in downtown Beverly Hills. It is incredibly messy, with old bike and car parts lying around everywhere. **(5.43)**
- In (5.42) as a car is probably too small to have bike and automobile parts lying around "everywhere," therefore the garage is almost surely the intended reference 'it' in this case.
- In order to resolve this reference, a system must be aware of the normal sizes of automobiles, garages, and the typical kinds of stuff one may find in each. On the other hand, one's familiarity with Beverly Hills might lead them to believe that the Acura is the passage's reference to that city (5.43).

5.6 ANAPHORA RESOLUTION

In this section two algorithms will be presented from anaphora resolution.

5.6.1 Hobbs Algorithm

- One method for pronoun resolving is the Hobbs algorithm. The syntactic parse tree of the sentences forms the foundation of the algorithm.
- Consider the (5.44) Jack and Jill example to better grasp the concept and how we as humans attempt to resolve the pronoun "his."
- Jack and Jill went up the hill, To fetch a pail of water. Jack fell down and broke 'his' crown, And Jill came tumbling after. **(5.44)**
- Here, Jack, Jill, hill, water, and crown are the potential resolution candidates for pronoun 'his'.
- Why, then, did we not consider the crown as a potential remedy? Perhaps because the word 'his' was followed by the noun 'crown'. The Hobbs algorithm's initial presumption states that the referent search must always be confined to the left of the target, eliminating the crown.
- If so, are Jill, water, or hill potential referents?
- But since Jill is a girl, we are aware that 'his' might not be referring to 'her'. The pronouns 'he', 'his', 'she', 'her', etc. are typically used to refer to male or female animate objects, while 'it' is used to refer to inanimate objects. This characteristic, referred as gender agreement, rules out the possibilities of Jill, hill, and water.
- Only a few sentences can be referenced with pronouns, and entities that are closest to the referring word are more significant than those that are farther away. It ultimately left us with Jack as the only option. Recency property is the name given to this property.
- Now, let's explore how we can embed intelligence (using the Hobbs algorithm) in robots that lack common sense to accomplish the task of pronoun resolution now that we have a better grasp of how humans digest text and resolve pronouns.

Algorithm

- It is important to note that, when resolving pronouns, the algorithm makes use of syntactic constraints as mentioned in the previous sections. The pronoun to be resolved and the syntactic parse of the sentences up to and including the current sentence serve as the input to the Hobbs algorithm.

- Fig 5.6.1 details the Hobbs algorithm.

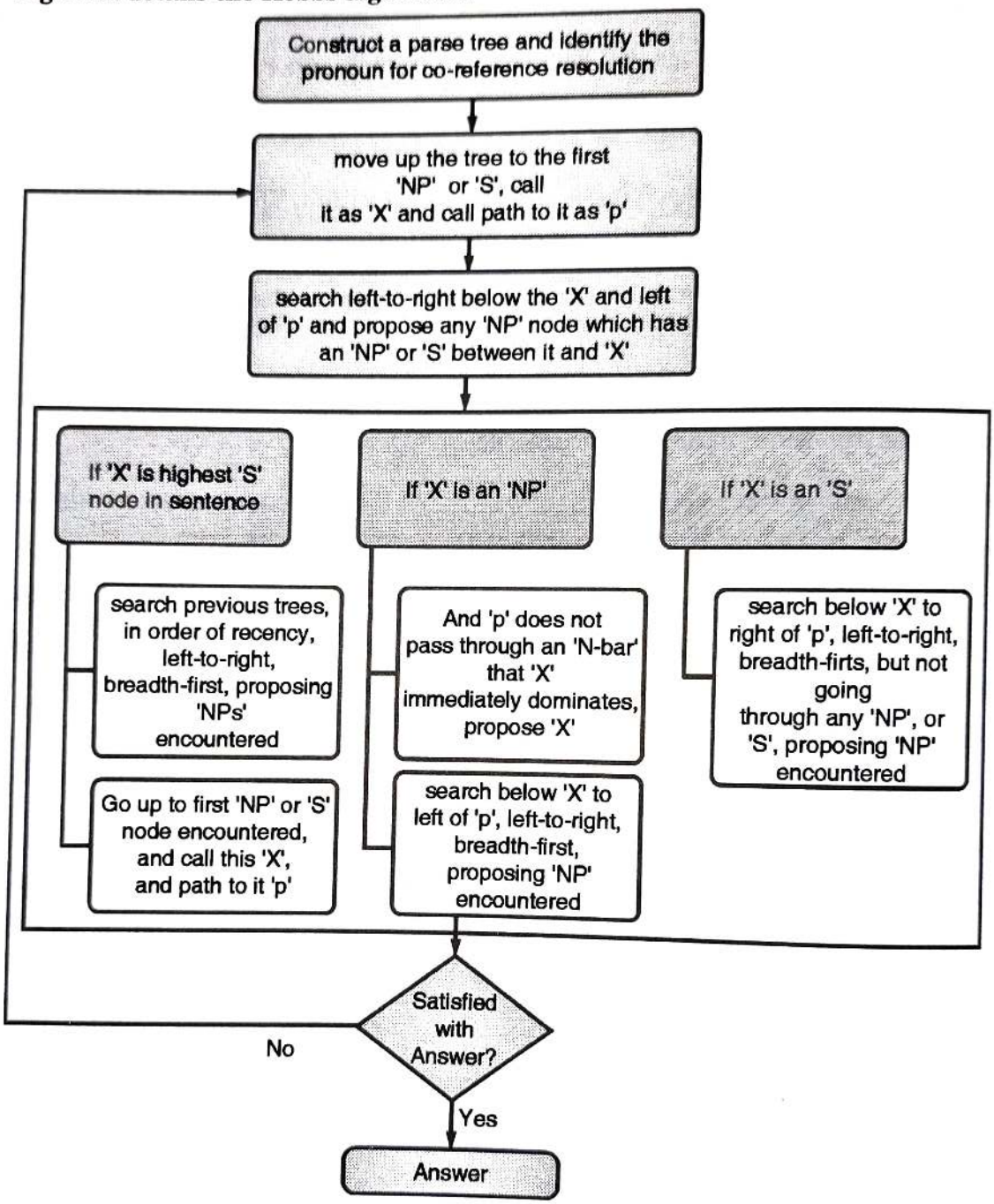


Fig 5.6.1 : Hobbs Algorithm

Example

Let's look at two sentences :

(5.45)

- **Sentence 1(S1) :** Jack is an engineer.
- **Sentence 2 (S2) :** Jill likes him.

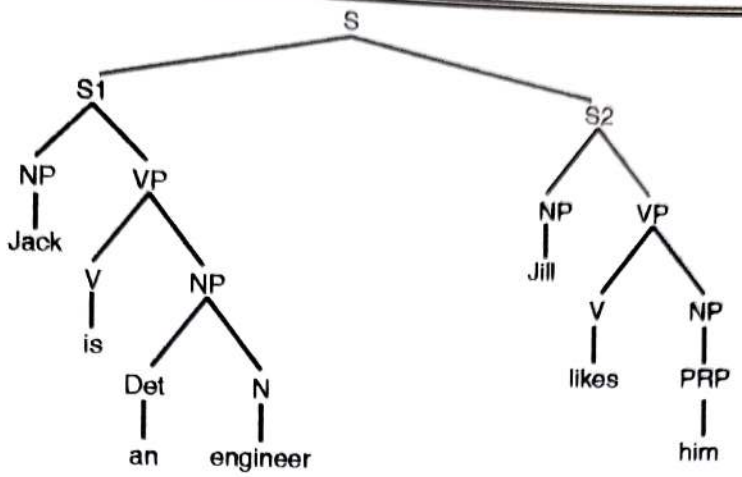


Fig. 5.6.2 : Parse Tree for the given sentences

- The problem identified is to resolve the pronoun 'him'.

☞ **Based on the algorithm**

1. Go back from 'him' to 'PRP' to 'NP' and this 'NP' will be called as 'X' and the path 'him' to 'PRP' to 'NP' as 'p'.
2. As there is no path to the left of 'p', we again go back from 'NP' to 'VP' to 'S2', which is our new 'X'.
3. Due to the syntactic restrictions imposed by Binding theory, it does not, however, study that branch. According to binding theory, a nonreflexive cannot refer to the subject of the most immediate clause in which it appears. However, a reflexive can. Reflexive words include such like himself, herself, themselves, etc. So, we again go backwards to 'S', becomes my new 'X'.
4. From 'S' we move to 'S1', which is our new 'X'.
5. Now we have encountered 'NP' when search the path. This 'NP' does not violate any constraints hence, this is out solution or answer. That is, the 'him' in the given sentences refers to 'Jack'.

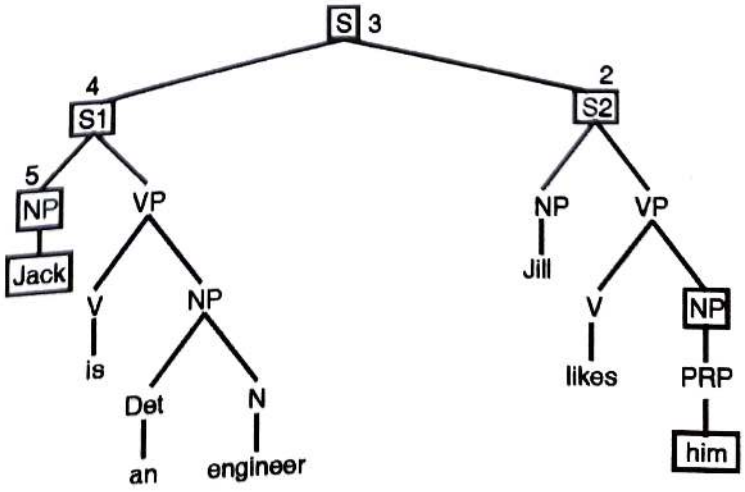


Fig. 5.6.3 : Resolution Path

Fig 5.6.3 shows the pronoun resolution path for the pronoun 'him'.

5.6.2 Centering Algorithm

- An explicit discourse model representation is not used by the Hobbs algorithm. Centering theory, in contrast, is a family of models that explicitly represents a discourse model and also makes the claim that at any given point in the discourse, only one entity is being "focused" and that this object should be distinguished from all other entities that have been evoked.
- There are two main representations tracked in the Centering theory discoursemodel. In what follows, take U_n and U_{n+1} to be two adjacent utterances. The Backward looking center of U_n, denoted as $C_n(U_n)$, represents the entity currently being focused on Forward looking in the discourse after U_n is interpreted. The forward looking centers of U_n, denoted as $C_f(U_n)$, form an ordered list containing the entities mentioned in U_n, all of which could serve as the C_b of the following utterance. In fact, $C_b(U_{n+1})$ is by definition the most highly ranked element of $C_f(U_n)$ mentioned in U_{n+1}. (The C_b of the first utterance in a discourse is undefined.) As for how the entities in the $C_f(U_n)$ are ordered, for simplicity's sake we can use the grammatical role hierarchy below.
- Subject > existential predicate nominal > object >indirect object or oblique > demarcated adverbial PP
- We call the highest rank forward looking center as Cp (preferred center).
- The preferred referents of pronouns are determined by the algorithm based on the relationships between the forward and backward looking centres in neighbouring phrases. According to the link between $C_b(U_{n+1})$, $C_b(U_n)$, and $C_p(U_{n+1})$, four intersentential interactions between a pair of utterances U_n and U_{n+1} are specified; these are depicted in Fig 5.6.4.

	$C_b = (U_{n+1}) = C_b(U_n)$ Or undefined $C_b(U_n)$	$C_b = (U_{n+1}) \neq C_b(U_n)$
$C_b = (U_{n+1}) = C_b(U_{n+1})$	Continue	Smooth-Shift
$C_b = (U_{n+1}) \neq C_b(U_{n+1})$	Retain	Rough-Shift

Fig. 5.6.4 : Transitions

Following rules are to be followed :

- **Rule 1 :** If any element of $C_f(U_n)$ is realized by a pronoun in utterance U_{n+1}, then $C_b(U_{n+1})$ must be realized as a pronoun also.

- **Rule 2 :** Transition states are ordered. Continue is preferred to Retain is preferred to Smooth-Shift is preferred to Rough-Shift.

The algorithm is defined as follows.

1. Create potential C_b-C_f pairings for every conceivable arrangement of reference assignments.
2. Use constraints, such as selectional limitations, centering rules, and syntactic coreference requirements, to narrow the results.
3. Order by transitional phrases

- If Rule 1 and other coreference requirements (gender, number, syntactic, and selectional restrictions) are not broken, the pronominal referents that are assigned are those that produce the most desired relation in Rule 2.Let us step through passage (21.66) to illustrate the algorithm.
- (5.46)
- John saw a beautiful 1961 Ford Falcon at the used car dealership. (U_1)
- He showed it to Bob. (U_2)
- He bought it. (U_3)
- Using the grammatical role hierarchy to order the Cf , for sentence U1 we get:

  C_f **(U_1) :** {John, Ford, dealership}

  C_p **(U_1) :** John

  C_b **(U_1) :** undefined

- The pronouns 'he', which is interchangeable with John, and 'it', which is interchangeable with the Ford or the dealership, are present in sentence U_2. John is C_b (U_2) by definition because 'he' is the highest-ranking C_f (U_1) member mentioned in U_2 (since 'he' is the only possible referent for 'he'). For each potential referent of 'it', the ensuing transitions are compared. The assignments would be as follows if we presume 'it' pertains to the Falcon:

  C_f **(U_2) :** {John, Ford, Bob}

  C_p **(U_2) :** John

  C_b **(U_2) :** John

Result : Continue ($C_p(U_2)= C_b(U_2)$; $C_b(U_1)$ undefined)

If we assume it refers to the dealership, the assignments would be:

$C_f(U_2)$: {John, dealership, Bob}

$C_P(U_2)$: John

$C_b(U_2)$: John

Result : Continue ($C_p(U_2)= C_b(U_2)$; $C_b(U_1)$ undefined)

- Since both options lead to a Continue transition, the algorithm is unable to determine which should be chosen. We will assume for the purposes of example that ties are broken in terms of the ranking on the prior C_f list. As a result, we will assume that 'it' refers to the Falcon rather than the dealership, maintaining the first choice above's representation of the current discourse model.
- While it is compatible with the Ford in sentence U_3, he is compatible with either John or Bob. If we take him to mean John, then John is $C_b(U_3)$, and the tasks would be:

$C_f(U_3)$: {John, Ford}

$C_p(U_3)$: John

$C_b(U_3)$: John

Result : Continue ($C_p(U_3)=C_b(U_3)=C_b(U_2)$)

- If we assume he refers to Bob, then Bob is Cb(U3) and the assignments would be:

$C_f(U_3)$: {Bob, Ford}

$C_P(U_3)$: Bob

$C_b(U_3)$: Bob

Result : Smooth-Shift ($C_p(U_3)=C_b(U_3)$; $C_b(U_3)6=C_b(U_2)$)

- Since a Continue is preferred to a Smooth-Shift per Rule 2, John is correctly taken to be the referent.
- The key salient factors that the centering algorithm implicitly takes into account are the preferences for repeated mentions, recency, and grammatical roles. Since the resulting transition type determines the final reference assignments, the grammatical role hierarchy only indirectly influences salience.
- In particular, if the former results in a more highly ranked transition, a referent in a low-ranking grammatical role will be preferred to one in a more highly ranked position.

- Consequently, the centering algorithm can mistakenly resolve a pronoun to a referent with low salience. For illustration's sake (5.47),
- Bob opened up a new dealership last week. John took a look at the Fords in his lot. He ended up buying one. **(5.47)**
- The third sentence's subject pronoun, "he," will be assigned to Bob by the centering method because Bob is $C_b(U_2)$, whereas John will be assigned if John is $C_b(U_2)$, resulting in a Smooth-Shift relation.
- On the other hand, John will be accurately designated as the referent by the Hobbs algorithm. The centering algorithm, like the Hobbs algorithm, needs both morphological gender detectors and a complete syntactic parse.
- As a model of entity coherence, centering theory also has implications for other discourse applications, such as summarization.

Chapter Ends...

❑❑❑