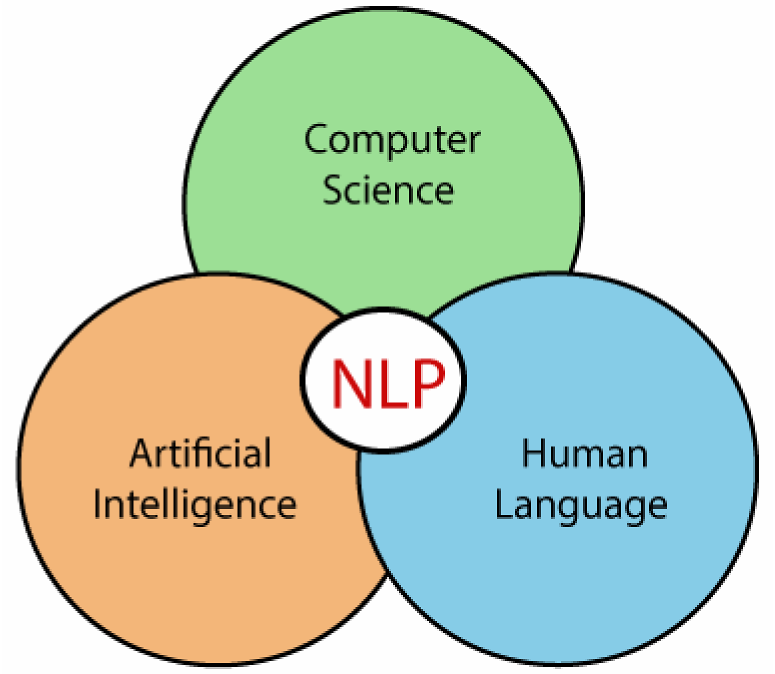
Module 1: Natural Language Processing

Components - Basics of Linguistics and Probability and Statistics – Words-Tokenization-Morphology:
Inflectional Morphology - Derivational Morphology. Finite-State Morphological Parsing - Porter Stemmer.

I. Introduction

Language is a method of communication with the help of which we can speak, read and write. For example, we think, we make decisions, plans and more in natural language; precisely, in words. However, the big question that confronts us in this AI era is that can we communicate in a similar manner with computers. In other words, can human beings communicate with computers in their natural language? It is a challenge for us to develop NLP applications because computers need structured data, but human speech is unstructured and often ambiguous in nature.
In this sense, we can say that Natural Language Processing (NLP) is the sub-field of Computer Science especially Artificial Intelligence (AI) that is concerned about enabling computers to understand and process human language. Technically, the main task of NLP would be to program computers for analyzing and processing huge amount of natural language data.



NLP is a part of **Computer Science, Human language,** and **Artificial Intelligence**. It is the technology that is used by machines to understand, analyze, manipulate, and interpret human languages. It helps developers to organize knowledge for performing tasks such as **translation, automatic summarization, Named Entity Recognition (NER), speech recognition, relationship extraction,** and **topic segmentation**.
Language is a crucial component for human lives and also the most fundamental aspect of our behavior. We can experience it in mainly two forms - written and spoken. In the written

form, it is a way to pass our knowledge from one generation to the next. In the spoken form, it is the primary medium for human beings to coordinate with each other in their day-to-day behavior. Language is studied in various academic disciplines. Each discipline comes with its own set of problems and a set of solution to address those.
Consider the following table to understand this −

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

History of NLP

Natural Language Processing started in 1950 When **Alan Mathison Turing** published an article in the name **Computing Machinery and Intelligence.** It is based on Artificial intelligence. It talks about automatic interpretation and generation of natural language. As the technology evolved, different approaches have come to deal with NLP tasks.

- **Heuristics-Based NLP:** This is the initial approach of NLP. It is based on defined rules. Which comes from domain knowledge and expertise. Example: regex
- **Statistical Machine learning-based NLP:** It is based on statistical rules and machine learning algorithms. In this approach, algorithms are applied to the data and learned from the data, and applied to various tasks. Examples: Naive Bayes, support vector machine (SVM), hidden Markov model (HMM), etc.
- **Neural Network-based NLP:** This is the latest approach that comes with the evaluation of neural network-based learning, known as Deep learning. It provides good accuracy, but it is a very data-hungry and time-consuming approach. It requires high computational power to train the model. Furthermore, it is based on neural

network architecture. Examples: Recurrent neural networks (RNNs), Long short-term memory networks (LSTMs), Convolutional neural networks (CNNs), Transformers, etc.

Advantages of NLP

- NLP helps users to ask questions about any subject and get a direct response within seconds.
- NLP offers exact answers to the question means it does not offer unnecessary and unwanted information.
- NLP helps computers to communicate with humans in their languages.
- It is very time efficient.
- Most of the companies use NLP to improve the efficiency of documentation processes, accuracy of documentation, and identify the information from large databases.

Disadvantages of NLP

A list of disadvantages of NLP is given below:

- NLP may not show context.
- NLP is unpredictable
- NLP may require more keystrokes.
- NLP is unable to adapt to the new domain, and it has a limited function that's why NLP is built for a single and specific task only.

Components of NLP

There are two components of Natural Language Processing:

- Natural Language Understanding
- Natural Language Generation

1. Natural Language Understanding (NLU)

Natural Language Understanding (NLU) helps the machine to understand and analyse human language by extracting the metadata from content such as concepts, entities, keywords, emotion, relations, and semantic roles.

NLU mainly used in Business applications to understand the customer's problem in both spoken and written language.

NLU involves the following tasks -

- It is used to map the given input into useful representation.
- It is used to analyze different aspects of the language.

2. Natural Language Generation (NLG)

Natural Language Generation (NLG) acts as a translator that converts the computerized data into natural language representation. It mainly involves Text planning, Sentence planning, and Text Realization.

- **Text planning** − It includes retrieving the relevant content from knowledge base.
- **Sentence planning** − It includes choosing required words, forming meaningful phrases, setting tone of the sentence.
- **Text Realization** − It is mapping sentence plan into sentence structure.

Difference between NLU and NLG

NLU	**NLG**
NLU is the process of reading and interpreting language.	NLG is the process of writing or generating language.
It produces non-linguistic outputs from natural language inputs.	It produces constructing natural language outputs from non-linguistic inputs.

Difficulties in NLU

NL has an extremely rich form and structure.
It is very ambiguous. There can be different levels of ambiguity −

- **Lexical ambiguity** − It is at very primitive level such as word-level.
- For example, treating the word "board" as noun or verb?
- **Syntax Level ambiguity** − A sentence can be parsed in different ways.
- For example, "He lifted the beetle with red cap." − Did he use cap to lift the beetle or he lifted a beetle that had red cap?
- **Referential ambiguity** − Referring to something using pronouns. For example, Rima went to Gauri. She said, "I am tired." − Exactly who is tired?
- One input can mean different meanings.
- Many inputs can mean the same thing.

Applications of NLP

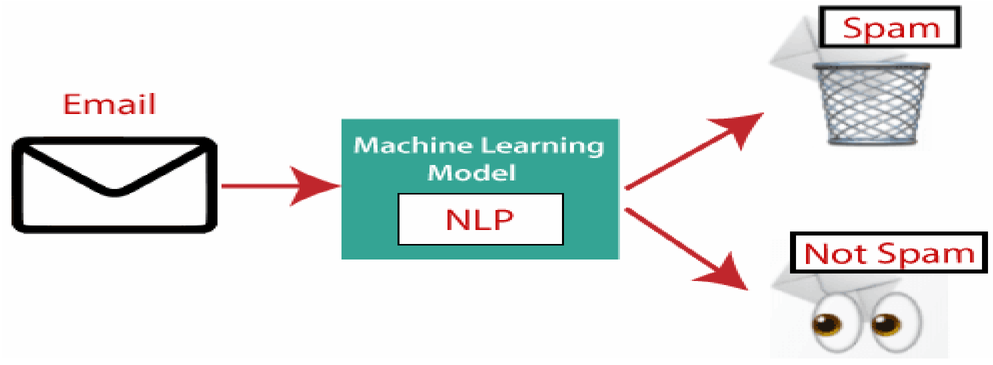
There are the following applications of NLP -

1. Question Answering

Question Answering focuses on building systems that automatically answer the questions asked by humans in a natural language.

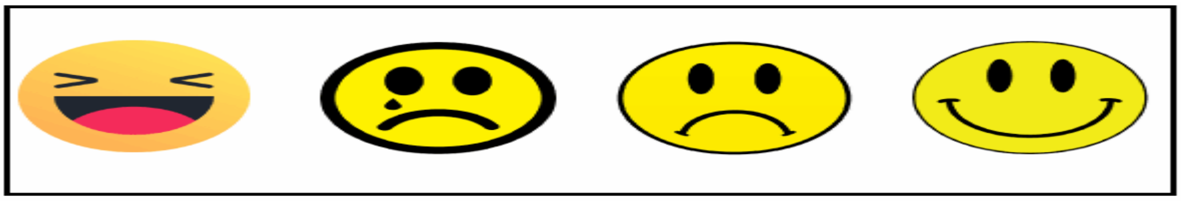
2. Spam Detection

Spam detection is used to detect unwanted e-mails getting to a user's inbox



3. Sentiment Analysis

Sentiment Analysis is also known as **opinion mining**. It is used on the web to analyse the attitude, behaviour, and emotional state of the sender. This application is implemented through a combination of NLP (Natural Language Processing) and statistics by assigning the values to the text (positive, negative, or natural), identify the mood of the context (happy, sad, angry, etc.)



4. Machine Translation
Machine translation is used to translate text or speech from one natural language to another natural language.

5. Spelling correction
Microsoft Corporation provides word processor software like MS-word, PowerPoint for the spelling correction.

6. Speech Recognition
Speech recognition is used for converting spoken words into text. It is used in applications, such as mobile, home automation, video recovery, dictating to Microsoft Word, voice biometrics, voice user interface, and so on.

7. Chatbot
Implementing the Chatbot is one of the important applications of NLP. It is used by many companies to provide the customer's chat services.

8. Information extraction
Information extraction is one of the most important applications of NLP. It is used for extracting structured information from unstructured or semi-structured machine-readable documents.

9. Natural Language Understanding (NLU)
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.

Basics of Linguistics:

1. Phonetics and Phonology
- **Phonetics** deals with the physical sounds of speech and how they are produced and perceived.
- **Phonology** focuses on how sounds function within a particular language.

Relevance in NLP:
  - Used in **speech recognition** and **speech synthesis** systems.
  - Helps in analyzing the sound structure for tasks like **text-to-speech** and **voice assistants**.

2. Morphology
- **Morphology** is the study of the structure of words and how they are formed from morphemes, which are the smallest units of meaning.

  - **Inflectional Morphology**: Modifies a word to express different grammatical categories (e.g., tense, number).
  - **Derivational Morphology**: Creates new words by adding prefixes or suffixes (e.g., *happy → unhappy*).

Relevance in NLP:

  - Used in **tokenization** (breaking text into individual words or subwords) and **morphological analysis**.
  - Important for building **stemming** and **lemmatization** algorithms, which are used to normalize text.
  - Helps in understanding word formation for **language models** and **machine translation**.

3. Syntax

- **Syntax** is the study of how words combine to form sentences, and the rules governing sentence structure.

Relevance in NLP:

  - Crucial for **parsing** and understanding the grammatical structure of sentences.
  - Used in **dependency parsing** and **constituency parsing** to break down a sentence into its grammatical components.
  - Important for **machine translation**, **question answering systems**, and **text generation**.

4. Semantics

- **Semantics** is concerned with the meaning of words, phrases, and sentences.
  - **Lexical Semantics**: Meaning of individual words.
  - **Compositional Semantics**: Meaning of phrases and sentences as a whole.

Relevance in NLP:

  - Helps in building systems that understand the meaning of text, such as **semantic analysis**, **sentiment analysis**, and **machine translation**.
  - Important for **word embeddings** like Word2Vec or GloVe, which capture semantic relationships between words.
  - Key in **natural language understanding (NLU)** tasks like **information retrieval** and **text classification**.

5. Pragmatics

- **Pragmatics** deals with how context influences the interpretation of meaning.
  - For example, understanding the intent behind a question, sarcasm, or politeness.

Relevance in NLP:

  - Important for **dialogue systems**, **chatbots**, and **context-aware applications**.
  - Helps in improving **text generation** and understanding **user intent** in **conversational agents**.

6. Discourse

- **Discourse analysis** looks at how sentences are structured in larger texts or conversations to create meaning.

Relevance in NLP:

  - Used in building **coherent text generation** systems like **story generation** or **summarization**.

  - Important for **dialogue management** and ensuring that interactions in chatbots and virtual assistants are contextually consistent.

7. Pragmatic and Sociolinguistics

- Focuses on how language is used in different social contexts, such as formal versus informal speech or how dialects influence communication.

Relevance in NLP:

  - Important in **sentiment analysis**, **emotion detection**, and understanding **social media language**.
  - Helps in developing models that adapt to different speaking or writing styles, tones, or regional variations.

Applications of Linguistics in NLP

1. **Speech Recognition**: Phonetics and phonology help in mapping speech sounds to words.
2. **Machine Translation**: Syntax, semantics, and morphology help translate text between languages.
3. **Sentiment Analysis**: Semantics and pragmatics allow models to infer the sentiment behind text.
4. **Chatbots and Conversational AI**: Syntax, semantics, and pragmatics work together to help chatbots understand and respond naturally.

Let us dive into the details of the above concepts.

Phonetics and Phonology in Linguistics

1. Phonetics

Phonetics is the study of the physical sounds of human speech. It focuses on how speech sounds are produced, transmitted, and received. Phonetics deals with the articulation (how sounds are made using the mouth, tongue, and vocal cords) and the acoustic properties of sounds (such as pitch, loudness, and duration). It is concerned with describing all sounds in a language.

- **Example**: In the word *bat*, the phonetic sounds (or phones) can be represented as:
  - /b/ (a voiced bilabial stop, where the vocal cords vibrate as the lips come together)
  - /æ/ (a low-front vowel sound, as in *cat* or *hat*)
  - /t/ (a voiceless alveolar stop, where the tongue touches the ridge behind the upper front teeth).

Each sound can be represented phonetically using the **International Phonetic Alphabet (IPA)**, which provides a unique symbol for each distinct sound in human speech.

2. Phonology

Phonology, on the other hand, is the study of how sounds function within a particular language or languages. It focuses on how sounds are organized in the mind, how they pattern, and how they change in different linguistic environments. Phonology deals with **phonemes**, which are the smallest units of sound that can change the meaning of a word.

- **Example**: In English, the sounds /p/ and /b/ are two different **phonemes**. This means that if you change the /p/ sound in *pat* to a /b/, the word becomes *bat*, which has a completely different meaning.
    - /p/ and /b/ are thus **contrastive** in English because replacing one with the other results in a different word.
    - However, in some languages, like Arabic, the difference between these sounds may not change meaning, and they could be considered variants (or **allophones**) of the same phoneme.

Phonetics vs. Phonology Example

Let's consider the words **"tap"** and **"pat"** in English:

- **Phonetic analysis** focuses on the exact sounds (phones) in these words.
    - "Tap" can be broken down into the phonetic symbols: /t/ /æ/ /p/
    - "Pat" can be broken down into the phonetic symbols: /p/ /æ/ /t/

The sounds /t/ and /p/ are articulated differently but can be described using phonetic symbols regardless of their language.

- **Phonological analysis** examines the role of these sounds (phonemes) within the system of the English language.
    - In English, /t/ and /p/ are separate phonemes because replacing /t/ with /p/ (or vice versa) changes the meaning of the word.
    - Phonology also considers rules about how phonemes behave in different contexts, such as how the /t/ in "tap" might sound different from the /t/ in "butter" (where it can become a "flap" sound like /ɾ/).

Phonetic vs. Phonological Focus:

- **Phonetics** asks: "How are these sounds physically made and what are their acoustic properties?"
    - Example: Describing the /t/ sound in *tap* as a voiceless alveolar stop.
- **Phonology** asks: "How do these sounds function in the language, and how do they interact with other sounds?"
    - Example: In English, /t/ and /p/ are different phonemes, but in some languages, they might not be distinct.

Both concepts are crucial in **speech recognition** and **speech synthesis** in NLP, as understanding the physical properties of sounds (phonetics) and how they function in language (phonology) helps systems interpret and generate human speech more effectively.

Morphology in Linguistics

Morphology is the branch of linguistics that studies the structure and formation of words. It focuses on **morphemes**, which are the smallest meaningful units of language. Morphemes can be:

- **Free morphemes**: Can stand alone as words (e.g., *book*, *run*).
- **Bound morphemes**: Cannot stand alone and must attach to other morphemes (e.g., *-s*, *un-*).

Morphology helps in understanding how words are created and modified, which is crucial for tasks like **tokenization**, **stemming**, and **lemmatization** in Natural Language Processing (NLP).

Types of Morphology

1. **Inflectional Morphology**
    - Deals with changes in a word to express grammatical relationships without changing the word's core meaning.
    - Examples:
        - Plural: *cat* → *cats* (adding *-s* to indicate more than one)
        - Past tense: *play* → *played* (adding *-ed* to show past tense)
        - Comparative: *fast* → *faster* (adding *-er* to compare)
2. **Derivational Morphology**
    - Involves creating a new word with a new meaning by adding prefixes or suffixes to a base word. This often changes the word's grammatical category.
    - Examples:
        - Verb to noun: *teach* → *teacher* (adding *-er* to form a noun)
        - Adjective to adverb: *happy* → *happily* (adding *-ly* to form an adverb)
        - Prefixes: *unhappy* (adding *un-* to change the meaning to the opposite)

Examples of Morphemes

- In the word **"unhappiness"**:
    - *un-* (prefix, bound morpheme) means "not"
    - *happy* (root word, free morpheme) is the base meaning
    - *-ness* (suffix, bound morpheme) turns the adjective into a noun

In this example, we can see both derivational morphology (prefix *un-* and suffix *-ness*).

Morphological Parsing

Morphological parsing is the process of breaking down a word into its morphemes to understand its structure and meaning.

Example:

- Word: **"rewritten"**
    - *re-* (prefix, means "again")
    - *write* (root, meaning "to write")
    - *-en* (suffix, participle marker indicating passive or past participle)

Sample Questions and Answers on Morphology

1. Identify Morphemes in the Given Words:

- **Question**: Break down the following words into their morphemes:
    1. Unbelievable
    2. Plays
    3. Irreplaceable
- **Answer**:
    1. **Unbelievable** → *un-* (prefix, bound) + *believe* (root, free) + *-able* (suffix, bound)
    2. **Plays** → *play* (root, free) + *-s* (suffix, bound, indicating plural)
    3. **Irreplaceable** → *ir-* (prefix, bound) + *replace* (root, free) + *-able* (suffix, bound)

2. Classify the Following Words as Inflectional or Derivational:

- **Question**: Determine whether the morphological changes in the following words are inflectional or derivational:
    1. Cats

2. Teacher
3. Happier
4. Unhappiness

- **Answer**:
    1. **Cats** → Inflectional (plural form of *cat*)
    2. **Teacher** → Derivational (derived from *teach* by adding *-er* to create a noun)
    3. **Happier** → Inflectional (comparative form of *happy*)
    4. **Unhappiness** → Derivational (derived from *happy* with prefix *un-* and suffix *-ness*)

3. Word Formation:

- **Question**: Create new words using the given root and the appropriate morphemes.
    1. Root: **hope** → Add a derivational morpheme to make a noun.
    2. Root: **run** → Add an inflectional morpheme to show past tense.
- **Answer**:
    1. **Hope** → *hopeful* (Adding *-ful* to make an adjective) or *hopeless* (Adding *-less* to form another adjective with a negative meaning).
    2. **Run** → *ran* (Past tense, irregular form).

4. Stemming and Lemmatization:

- **Question**: Identify the root for the following inflected words:
    1. **Playing**
    2. **Quickest**
    3. **Unhappily**
- **Answer**:
    1. **Playing** → Root: *play*
    2. **Quickest** → Root: *quick*
    3. **Unhappily** → Root: *happy*

5. True/False Questions:

- **Question**: Indicate whether the following statements are true or false.
    1. Inflectional morphemes change the meaning and part of speech of a word.
    2. The morpheme *-ed* in *walked* is an example of derivational morphology.
- **Answer**:
    1. False (Inflectional morphemes change grammatical aspects like tense or number but don't change the core meaning or part of speech).
    2. False (The *-ed* suffix is an example of inflectional morphology, showing past tense).

In NLP, morphological analysis helps with tasks like:

- **Tokenization**: Breaking down text into meaningful units.
- **Stemming**: Reducing words to their root forms (e.g., *running* → *run*).
- **Lemmatization**: Returning words to their dictionary form based on context (e.g., *was* → *be*).

Understanding morphology is crucial for building accurate models for tasks like **machine translation**, **information retrieval**, and **text generation**.

Syntax in Linguistics

Syntax is the set of rules that govern how words are arranged to form phrases, clauses, and sentences in a language. It dictates the structure of a sentence and ensures that words are in the correct order so the sentence is grammatically correct and makes sense.
Syntax also defines the relationships between words in a sentence, such as subject, object, and verb, and how they interact to convey meaning.

Key Concepts in Syntax:

1. **Word Order**: The most basic rule of syntax is the order of words in a sentence. Different languages follow different word orders. In English, the most common word order is **Subject-Verb-Object (SVO)**.
2. **Sentence Structure**: Sentences can be simple (one clause) or complex (multiple clauses), but all sentences must follow syntactic rules to be grammatically correct.

Example of Syntax:

Consider the sentence:

"The cat chased the mouse."

- **Subject**: *The cat*
- **Verb**: *chased*
- **Object**: *the mouse*

This sentence follows the basic **SVO** (Subject-Verb-Object) word order in English. The subject (*the cat*) comes before the verb (*chased*), and the object (*the mouse*) follows the verb.

Syntactically Incorrect Example:

- *"Chased the cat the mouse."*

This sentence doesn't follow the correct word order in English, so it's syntactically incorrect even though it contains the same words as the correct sentence.

Types of Sentences Based on Syntax:

1. **Simple Sentence**: Contains one independent clause.
   - Example: *"The dog barks."*
2. **Compound Sentence**: Contains two or more independent clauses joined by a conjunction.
   - Example: *"The dog barks, and the cat runs."*
3. **Complex Sentence**: Contains one independent clause and one or more dependent clauses.
   - Example: *"The dog barks when the cat runs."*

Questions and Answers on Syntax:

Question 1: Identify the syntactically correct sentence:

1. *"The man the car drives."*
2. *"The man drives the car."*

Answer: Sentence 2 is syntactically correct because it follows the proper SVO word order.

Question 2: Correct the syntactically incorrect sentence:

"Playing the children are in the park."

Answer: *"The children are playing in the park."*

This correction places the subject (*the children*) before the verb (*are playing*), following proper syntactic rules.

Syntax in NLP:
In Natural Language Processing, understanding syntax is essential for parsing sentences, enabling machines to determine the structure of a sentence. This helps in tasks like machine translation, question answering, and text generation.
For example, a syntactic parser will identify the subject, verb, and object in a sentence to understand its grammatical structure and ensure that translations or responses are coherent.

Semantics in Linguistics
Semantics is the study of meaning in language. It focuses on the interpretation of words, phrases, and sentences, and how meaning is conveyed and understood. While syntax is concerned with the structure of sentences, **semantics** deals with the meaning behind those structures.
Semantics plays a crucial role in understanding how words relate to each other and how meaning can change based on word choice or context.

Key Concepts in Semantics:
1. **Lexical Semantics**: The meaning of individual words and the relationships between them. This includes synonyms (words with similar meanings), antonyms (words with opposite meanings), and homonyms (words that are spelled or pronounced the same but have different meanings).
2. **Compositional Semantics**: How the meaning of individual words combines to form the meaning of a larger expression, such as a phrase or sentence. This is often referred to as the "principle of compositionality," which states that the meaning of a sentence is determined by the meanings of its parts and the rules used to combine them.
3. **Ambiguity**: Sentences or phrases can have multiple meanings. Semantic analysis helps distinguish between these meanings based on context.
    - **Lexical Ambiguity**: When a word has multiple meanings (e.g., *bank* can refer to a financial institution or the side of a river).
    - **Structural Ambiguity**: When the structure of a sentence can lead to different interpretations (e.g., *"I saw the man with the telescope"* could mean either that the speaker used a telescope or the man had a telescope).

Example of Semantics:
Consider the sentence:
"The bank is on the river."
- **Lexical Semantics**: The word *bank* here can have two different meanings:
    - A financial institution.
    - The side of a river.

Based on context, we infer that in this case, *bank* refers to the land beside the river, not a place where you store money.

Compositional Semantics Example:
Consider the sentence:
"John kicked the ball."

- **Word Meaning**:
  - *John*: a proper noun, referring to a specific person.
  - *kicked*: an action involving hitting something with the foot.
  - *the ball*: a round object.
- **Compositional Meaning**: The sentence means that a person named John performed the action of kicking a round object.

Semantically Incorrect Example:

- **Syntactically Correct but Semantically Incorrect**: *"Colorless green ideas sleep furiously."*
  - The sentence is grammatically correct, but it doesn't make sense semantically. The meaning of the words doesn’t logically fit together, as something cannot be both "colorless" and "green," and ideas cannot "sleep."

Questions and Answers on Semantics:

Question 1: Which of the following sentences is semantically incorrect?

1. *"The sun rises in the east."*
2. *"The chair laughed at the joke."*

Answer: Sentence 2 is semantically incorrect. A chair, being an inanimate object, cannot "laugh."

Question 2: What is the meaning of the word *bank* in the sentence: *"He walked along the bank."*?

Answer: In this sentence, *bank* refers to the land beside the river (riverbank), not a financial institution.

Semantics in NLP:

In Natural Language Processing, understanding semantics is crucial for tasks like **machine translation**, **sentiment analysis**, and **question answering**. NLP systems need to understand the meaning behind the text, not just its structure.

For example:

- In **machine translation**, semantic analysis ensures that words are translated into their correct meanings based on context (e.g., translating *bank* correctly based on whether it refers to a financial institution or a riverbank).
- In **sentiment analysis**, semantics helps determine whether a sentence expresses positive, negative, or neutral sentiment based on the meanings of the words.

Pragmatics in Linguistics

Pragmatics is the branch of linguistics that studies how context influences the interpretation of meaning in communication. It focuses on how speakers use language in real-world situations and how listeners interpret it beyond just the literal meaning of words.

Unlike **semantics**, which deals with the literal meanings of words and sentences, pragmatics takes into account factors like:

- The speaker's intention.
- The listener's interpretation.
- The social and physical context of the conversation.

Pragmatics explains how meaning can change depending on context, tone, and other external factors, including shared knowledge between the speaker and listener.

Key Concepts in Pragmatics:

1. **Speech Acts**:
    - Utterances can function as actions. For example, when someone says, *"I apologize,"* they are not just stating a fact, they are performing the act of apologizing.
    - Types of speech acts include **requests**, **promises**, **apologies**, **commands**, etc.
2. **Context**:
    - **Physical context**: Where and when the communication takes place.
    - **Linguistic context**: What has been said before in the conversation.
    - **Social context**: The relationship between the speaker and the listener.
3. **Deixis**:
    - Words like *this*, *that*, *here*, *there*, *I*, *you* depend on the context to convey meaning. For example, the meaning of *"here"* depends on where the speaker is located.
4. **Implicature**:
    - When a speaker implies something without directly stating it. For instance, *"It's cold in here,"* could imply a request to close the window, even though that is not directly said.

Examples of Pragmatics:

1. Same Sentence, Different Meanings Based on Context:

- Sentence: *"Can you pass the salt?"*
- **Literal meaning (semantics)**: The speaker is asking about the listener's ability to pass the salt.
- **Pragmatic meaning**: In everyday conversation, the speaker is not asking about the listener's ability but is actually making a polite request for the salt to be passed.

2. Speech Acts:

- **Apology**: *"I'm sorry for being late."*
    - The sentence is not just a statement of fact but also the act of apologizing.
- **Request**: *"Could you please open the window?"*
    - Even though it's framed as a question, pragmatically it functions as a polite request for the listener to open the window.

3. Deixis (Context-Dependent Words):

- **Example**: *"I will meet you here tomorrow."*
    - The meaning of *"I"*, *"you"*, *"here"*, and *"tomorrow"* can only be understood with knowledge of who the speaker and listener are, the location of the conversation, and the current date.

4. Implicature:

- **Example**:
    - *"It's really noisy in here."*
        - The literal meaning is simply an observation about the noise level.

        - The **pragmatic meaning** might be that the speaker is indirectly suggesting to move to a quieter place.
- **Example**:
    - *"Do you know what time it is?"*
        - The literal meaning is asking whether the listener knows the time.
        - Pragmatically, it's likely a polite way of asking for the time.

Pragmatics vs. Semantics:
- **Semantics**: Deals with the literal meaning of words and sentences.
    - Example: *"John is tall"* simply means that John has a tall stature.
- **Pragmatics**: Takes context and intention into account, providing an understanding beyond the literal meaning.
    - Example: *"John is tall"* could imply that John is taller than expected or that John should reach something high, depending on context.

Pragmatics in Real Life:
1. **Indirect Requests**:
    - **Example**:
        - Sentence: *"The trash is full."*
        - **Literal meaning**: A statement about the trash.
        - **Pragmatic meaning**: This could be an indirect request for someone to take out the trash.
2. **Politeness**:
    - **Example**:
        - Sentence: *"Could you close the door?"*
        - **Literal meaning**: A question about the listener's ability to close the door.
        - **Pragmatic meaning**: It's actually a polite request for the listener to close the door, not a question about capability.
3. **Sarcasm**:
    - **Example**:
        - Sentence: *"Oh, great! Another traffic jam."*
        - **Literal meaning**: The speaker is expressing excitement.
        - **Pragmatic meaning**: The speaker is actually expressing frustration, using sarcasm to convey the opposite of what the words literally mean.

Questions and Answers on Pragmatics:
Question 1: What is the pragmatic meaning of the sentence *"Can you pass the salt?"* when spoken at a dinner table?
1. A question about the listener's ability to pass the salt.
2. A polite request to pass the salt.

Answer: Option 2, a polite request to pass the salt.

Question 2: Which of the following is an example of implicature?
1. *"Could you open the door?"* (Literal request for opening the door)

2. *"It's really hot in here."* (Implying that the speaker wants the listener to open a window)

Answer: Option 2. The speaker is implying that they want the listener to do something (open a window) without directly stating it.

Pragmatics in NLP:

In **Natural Language Processing (NLP)**, understanding pragmatics is important for tasks like:

- **Dialogue systems**: Virtual assistants need to understand not only what is said but also the context and intention behind it.
- **Sentiment analysis**: Pragmatic understanding helps in detecting sarcasm or politeness in a sentence.
- **Contextual understanding**: Chatbots and voice assistants use pragmatics to handle user queries based on the context of the conversation.

Role of Probability in Language Models

In Natural Language Processing (NLP), **probability** plays a central role in building **language models**. A language model is a statistical tool that helps predict the likelihood of a sequence of words, and it relies on probability to make these predictions. By understanding the probabilistic relationships between words and phrases, language models can generate, recognize, and interpret natural language text.

The role of probability in language models can be broken down into the following key aspects:

1. Predicting the Next Word

A fundamental application of probability in language models is predicting the likelihood of the next word in a sequence, given the previous words. This is known as **next-word prediction** or **conditional probability**.

- **Conditional Probability**: The probability of a word w_n occurring given the preceding words $w_1, w_2, \dots, w_{n-1}$ is denoted as $P(w_n | w_1, w_2, \dots, w_{n-1})$.

Example:

Suppose you have the partial sentence:

"The dog is..."

A language model might assign the following probabilities to the next word:

- $P(\text{barking}|\text{The dog is}) = 0.6$
- $P(\text{running}|\text{The dog is}) = 0.3$
- $P(\text{blue}|\text{The dog is}) = 0.01$

The word *"barking"* has the highest probability based on the context, so the model is most likely to predict it as the next word.

2. Language Modeling

Language models are used to estimate the probability distribution over sequences of words. These models help assess how likely a sentence is by calculating the joint probability of all the words in a sentence. This is useful in applications such as machine translation, speech recognition, and text generation.

- **Unigram Model**: Treats each word as independent of others.
  - $P(w_1, w_2, w_3, ..., w_n) = P(w_1) \times P(w_2) \times P(w_3) \times ... \times P(w_n)$
- **Bigram Model**: Considers the probability of a word based on the previous word.
  - $P(w_1, w_2, w_3, ..., w_n) = P(w_1) \times P(w_2|w_1) \times P(w_3|w_2) \times ... \times P(w_n|w_{n-1})$
- **n-Gram Model**: Generalizes the bigram model by considering the previous $n-1$ words.
  - $P(w_1, w_2, ..., w_n) = P(w_1) \times P(w_2|w_1) \times P(w_3|w_1, w_2) \times ... \times P(w_n|w_{n-2}, w_{n-1})$

The **n-gram** approach uses probability to predict the likelihood of word sequences based on how often they occur together in large text corpora.

3. Handling Ambiguity

In many cases, a sentence can be interpreted in multiple ways, especially when it contains ambiguous words. Probability helps resolve this ambiguity by calculating which interpretation is more likely based on context.

Example:

- Sentence: *"I saw the man with the telescope."*
  - One interpretation: I used the telescope to see the man.
  - Another interpretation: The man I saw had a telescope.

A language model will calculate the probability of each interpretation based on the surrounding context and choose the one with the higher probability.

4. Smoothing Techniques

In practice, some word sequences may not occur frequently or may not exist at all in the training data, resulting in a probability of zero. **Smoothing** techniques are used to handle these cases and assign a small probability to unseen word sequences.

- **Additive Smoothing (Laplace Smoothing)**: Adds a small constant to all possible word sequences to ensure none have zero probability.
- $P(w_n|w_{n-1}) = \frac{C(w_{n-1}, w_n) + \alpha}{C(w_{n-1}) + \alpha \cdot V}$
  - Where $C(w_{n-1}, w_n)$ is the count of the bigram, α is a small constant, and V is the vocabulary size.

Backoff and Interpolation: These techniques combine n-grams of different lengths. If a higher-order n-gram has zero probability, a lower-order n-gram (such as a bigram or unigram) is used to estimate the probability.

5. Perplexity

Perplexity is a metric used to evaluate the performance of a language model. It measures how well a probabilistic model predicts a sample. A lower perplexity score indicates that the model is better at predicting the probability distribution of the words in the text.

- Formula:

$$\text{Perplexity}(P) = 2^{-\frac{1}{N}\sum_{i=1}^{N}\log_2 P(w_i|w_1,w_2,\ldots,w_{i-1})}$$

In essence, perplexity quantifies how uncertain a model is when predicting the next word in a sentence. Better models have lower perplexity scores.

6. Bayesian Inference in Language Models
Some language models, such as **Hidden Markov Models (HMMs)** and **Latent Dirichlet Allocation (LDA)**, use **Bayesian inference** to predict hidden variables (such as word categories or topics). Probability helps these models infer the underlying structure of text.
For example, in **part-of-speech tagging**, an HMM uses the probabilities of transitions between parts of speech and the likelihood of observing specific words to infer the most probable part of speech for each word in a sentence.

Applications of Probability in Language Models

1. **Text Generation**: Probabilistic models can generate coherent sentences by predicting the next word based on the previous ones. This is used in AI applications like GPT (Generative Pre-trained Transformer) models.
2. **Speech Recognition**: By modeling the probability of word sequences, language models help in transcribing speech into text, making sure the most likely sentence is selected based on context.
3. **Machine Translation**: Probabilistic language models translate sentences by estimating the likelihood of word sequences in both the source and target languages.
4. **Autocorrect and Predictive Text**: Probabilistic models predict the most likely next word a user intends to type or suggest corrections for mistyped words based on their likelihood in context.

Conditional Probability
Conditional probability refers to the probability of an event occurring given that another event has already occurred. It's a way of refining the probability of an event based on new information.
Mathematically, if A and B are two events, the conditional probability of A happening given that B has occurred is denoted as:

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

Where:

- $P(A|B)$ is the probability of event A given that B has occurred.
- $P(A \cap B)$ is the probability of both A and B occurring.
- $P(B)$ is the probability of event B.

The idea is that you're adjusting the probability of A by factoring in the occurrence of B.

Example 1: Drawing Cards from a Deck

Consider a standard deck of 52 cards. Let’s calculate the conditional probability of drawing an Ace (event A) given that the card drawn is a Spade (event B).

- There are 52 cards in the deck, and 13 of them are Spades.
- Out of the 13 Spades, only 1 is an Ace.

Thus:

- $P(A \cap B)$ (probability of drawing the Ace of Spades) = $\frac{1}{52}$.
- $P(B)$ (probability of drawing any Spade) = $\frac{13}{52}$.

The conditional probability of drawing an Ace given that the card is a Spade is:

$$P(A|B) = \frac{P(A \cap B)}{P(B)} = \frac{\frac{1}{52}}{\frac{13}{52}} = \frac{1}{13}$$

Example 2: Weather and Sports

Let’s say we are interested in the probability that a cricket match is played (event A) given that it is not raining (event B).

- Suppose the probability that a match is played on any given day is $P(A) = 0.4$.
- The probability that it does not rain on any given day is $P(B) = 0.7$.
- If it is not raining, the probability that the match is played is higher, say $P(A \cap B) = 0.35$ (since it's more likely the match will be played in clear weather).

Using the formula for conditional probability:

$$P(A|B) = \frac{P(A \cap B)}{P(B)} = \frac{0.35}{0.7} = 0.5$$

So, if it’s not raining, the probability that the cricket match is played increases to 0.5.

Example 3: Language Models (NLP)

In Natural Language Processing (NLP), conditional probability is widely used in predicting the next word in a sequence. For example, suppose we want to calculate the probability of the word "sun" occurring after the words "the" and "bright" in a sentence.

We use conditional probability to estimate this:

$$P(\text{sun}|\text{the, bright}) = \frac{P(\text{the, bright, sun})}{P(\text{the, bright})}$$

- $P(\text{the, bright, sun})$ is the probability of the three words occurring together.
- $P(\text{the, bright})$ is the probability of "the" and "bright" occurring together.

The model would use this conditional probability to predict "sun" as the next word given that "the bright" has already appeared in the text.

Example 4: Medical Diagnosis

In a medical diagnosis scenario, suppose the probability of having a certain disease D is $P(D) = 0.01$ (1% of the population). A diagnostic test is 95% accurate, meaning if a person has the disease, the test is positive with probability $P(+|D) = 0.95$. However, the test also has a false-positive rate, i.e., 5% of the time, the test is positive even for a healthy person, so $P(+|\neg D) = 0.05$.

If a person tests positive, the question is: What is the probability that the person actually has the disease, i.e., $P(D|+)$?

Using **Bayes' Theorem** (which is built on conditional probability):

$$P(D|+) = \frac{P(+|D) \cdot P(D)}{P(+)}$$

Where $P(+) = P(+|D) \cdot P(D) + P(+|\neg D) \cdot P(\neg D)$.

This allows the doctor to calculate the likelihood that the person has the disease based on a positive test result, factoring in the accuracy and false-positive rate of the test.

Conditional Probability
Problem 1

You toss a fair coin three times:

a. What is the probability of three heads, HHHHHH?
b. What is the probability that you observe exactly one heads?
c. Given that you have observed *at least* one heads, what is the probability that you observe at least two heads?

We assume that the coin tosses are independent.

a. $P(HHH) = P(H) \cdot P(H) \cdot P(H) = 0.5^3 = \frac{1}{8}$.

b. To find the probability of exactly one heads, we can write

$$\begin{aligned} P(\text{One heads}) &= P(HTT \cup THT \cup TTH) \\ &= P(HTT) + P(THT) + P(TTH) \\ &= \tfrac{1}{8} + \tfrac{1}{8} + \tfrac{1}{8} \\ &= \tfrac{3}{8}. \end{aligned}$$

c. Given that you have observed *at least* one heads, what is the probability that you observe at least two heads? Let A_1 be the event that you observe at least one heads, and A_2 be the event that you observe at least two heads. Then

$$A_1 = S - \{TTT\}, \text{ and } P(A_1) = \frac{7}{8};$$

$$A_2 = \{HHT, HTH, THH, HHH\}, \text{ and } P(A_2) = \frac{4}{8}.$$

Thus, we can write

$$\begin{aligned} P(A_2|A_1) &= \tfrac{P(A_2 \cap A_1)}{P(A_1)} \\ &= \tfrac{P(A_2)}{P(A_1)} \\ &= \tfrac{4}{8} \cdot \tfrac{8}{7} = \tfrac{4}{7}. \end{aligned}$$

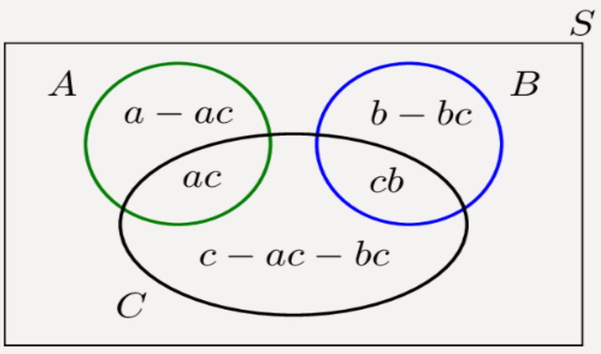
Problem 2

For three events A, B, and C, we know that

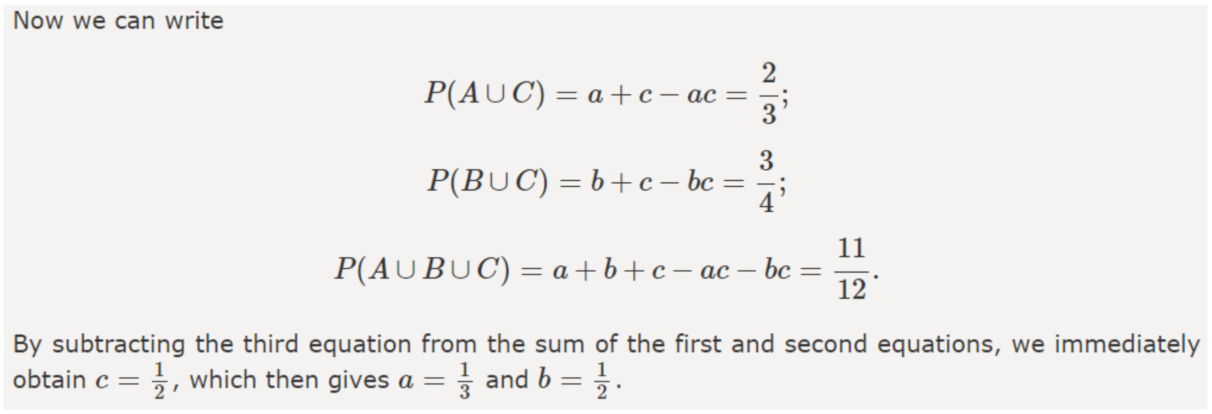
- A and C are independent,
- B and C are independent,
- A and B are disjoint,
- $P(A \cup C) = \frac{2}{3}, P(B \cup C) = \frac{3}{4}, P(A \cup B \cup C) = \frac{11}{12}$

Find $P(A), P(B)$, and $P(C)$.

We can use the Venn diagram in Figure 1.26 to better visualize the events in this problem. We assume $P(A) = a, P(B) = b$, and $P(C) = c$. Note that the assumptions about independence and disjointness of sets are already included in the figure.



$P(A) = a, P(B) = b, P(C) = c$



Now we can write

$$P(A \cup C) = a + c - ac = \frac{2}{3};$$

$$P(B \cup C) = b + c - bc = \frac{3}{4};$$

$$P(A \cup B \cup C) = a + b + c - ac - bc = \frac{11}{12}.$$

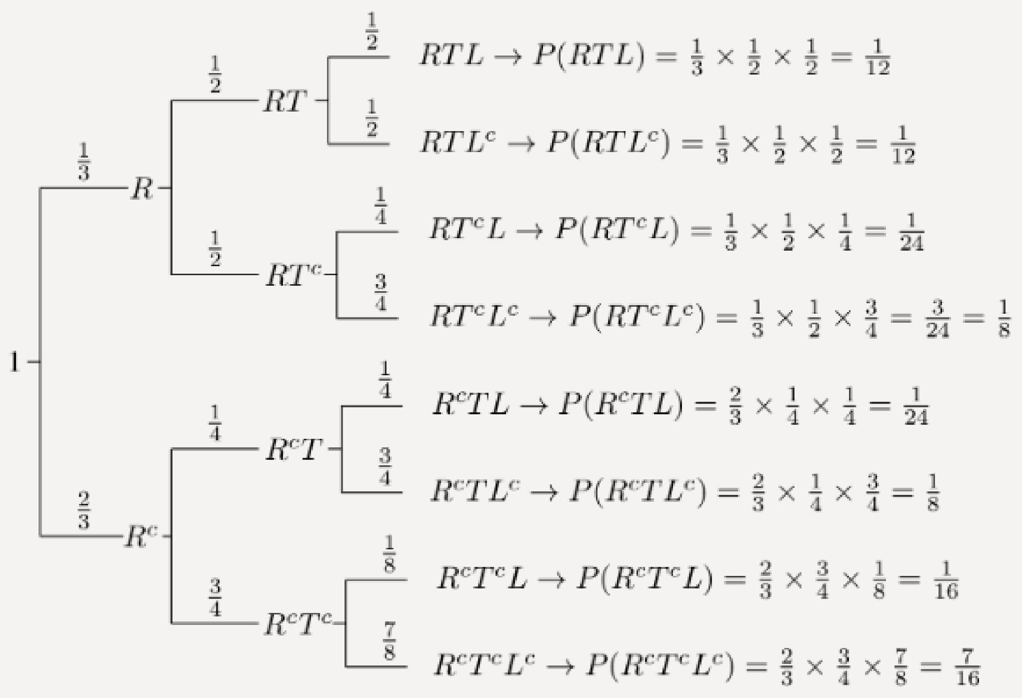
By subtracting the third equation from the sum of the first and second equations, we immediately obtain $c = \frac{1}{2}$, which then gives $a = \frac{1}{3}$ and $b = \frac{1}{2}$.

Problem 3

In my town, it's rainy one third of the days. Given that it is rainy, there will be heavy traffic with probability 1/2, and given that it is not rainy, there will be heavy traffic with probability 1/4. If it's rainy and there is heavy traffic, I arrive late for work with probability 1/2. On the other hand, the probability of being late is reduced to 1/8 if it is not rainy and there is no heavy traffic. In other situations (rainy and no traffic, not rainy and traffic) the probability of being late is 0.25. You pick a random day.

a. What is the probability that it's not raining and there is heavy traffic and I am not late?
b. What is the probability that I am late?
c. Given that I arrived late at work, what is the probability that it rained that day?

Let R be the event that it's rainy, T be the event that there is heavy traffic, and L be the event that I am late for work. As it is seen from the problem statement, we are given conditional probabilities in a chain format. Thus, it is useful to draw a tree diagram. Figure 1.27 shows a tree diagram for this problem. In this figure, each leaf in the tree corresponds to a single outcome in the sample space. We can calculate the probabilities of each outcome in the sample space by multiplying the probabilities on the edges of the tree that lead to the corresponding outcome.


1
$\frac{1}{3}$
R
$\frac{2}{3}$
R^c
$\frac{1}{2}$
RT
$\frac{1}{2}$
RT^c
$\frac{1}{4}$
R^cT
$\frac{3}{4}$
R^cT^c
$\frac{1}{2}$
$\frac{1}{2}$
$\frac{1}{4}$
$\frac{3}{4}$
$\frac{1}{4}$
$\frac{3}{4}$
$\frac{1}{8}$
$\frac{7}{8}$
$RTL \rightarrow P(RTL) = \frac{1}{3} \times \frac{1}{2} \times \frac{1}{2} = \frac{1}{12}$
$RTL^c \rightarrow P(RTL^c) = \frac{1}{3} \times \frac{1}{2} \times \frac{1}{2} = \frac{1}{12}$
$RT^cL \rightarrow P(RT^cL) = \frac{1}{3} \times \frac{1}{2} \times \frac{1}{4} = \frac{1}{24}$
$RT^cL^c \rightarrow P(RT^cL^c) = \frac{1}{3} \times \frac{1}{2} \times \frac{3}{4} = \frac{3}{24} = \frac{1}{8}$
$R^cTL \rightarrow P(R^cTL) = \frac{2}{3} \times \frac{1}{4} \times \frac{1}{4} = \frac{1}{24}$
$R^cTL^c \rightarrow P(R^cTL^c) = \frac{2}{3} \times \frac{1}{4} \times \frac{3}{4} = \frac{1}{8}$
$R^cT^cL \rightarrow P(R^cT^cL) = \frac{2}{3} \times \frac{3}{4} \times \frac{1}{8} = \frac{1}{16}$
$R^cT^cL^c \rightarrow P(R^cT^cL^c) = \frac{2}{3} \times \frac{3}{4} \times \frac{7}{8} = \frac{7}{16}$

a. The probability that it's not raining and there is heavy traffic and I am not late can be found using the tree diagram which is in fact applying the chain rule:

$$\begin{aligned} P(R^c \cap T \cap L^c) &= P(R^c)P(T|R^c)P(L^c|R^c \cap T) \\ &= \frac{2}{3} \cdot \frac{1}{4} \cdot \frac{3}{4} \\ &= \frac{1}{8}. \end{aligned}$$

b. The probability that I am late can be found from the tree. All we need to do is sum the probabilities of the outcomes that correspond to me being late. In fact, we are using the law of total probability here.

$$\begin{aligned} P(L) &= P(R,T,L) + P(R,T^c,L) + P(R^c,T,L) + P(R^c,T^c,L) \\ &= \frac{1}{12} + \frac{1}{24} + \frac{1}{24} + \frac{1}{16} \\ &= \frac{11}{48}. \end{aligned}$$

c. We can find $P(R|L)$ using $P(R|L) = \frac{P(R\cap L)}{P(L)}$. We have already found $P(L) = \frac{11}{48}$, and we can find $P(R\cap L)$ similarly by adding the probabilities of the outcomes that belong to $R\cap L$. In particular,

$$\begin{aligned} P(R\cap L) &= P(R,T,L) + P(R,T^c,L) \\ &= \frac{1}{12} + \frac{1}{24} \\ &= \frac{1}{8}. \end{aligned}$$

Thus, we obtain

$$\begin{aligned} P(R|L) &= \frac{P(R\cap L)}{P(L)} \\ &= \frac{1}{8} \cdot \frac{48}{11} \\ &= \frac{6}{11}. \end{aligned}$$

Joint probability

Joint probability refers to the probability of two or more events happening simultaneously. For two events A and B, the joint probability is denoted as $P(A\cap B)$, which is the probability that both A and B occur at the same time.

In general, for events A and B:

$$P(A\cap B) = P(A \text{ and } B)$$

The concept can be extended to multiple events. For three events A, B, and C, the joint probability would be $P(A\cap B\cap C)$, which means the probability of all three events occurring together.

Formula for Joint Probability

If the events are **independent** (meaning the occurrence of one event does not affect the occurrence of the other), the joint probability is the product of their individual probabilities:

$$P(A\cap B) = P(A) \times P(B)$$

For **dependent** events, the joint probability is given by:

$$P(A\cap B) = P(A) \times P(B|A)$$

Where:

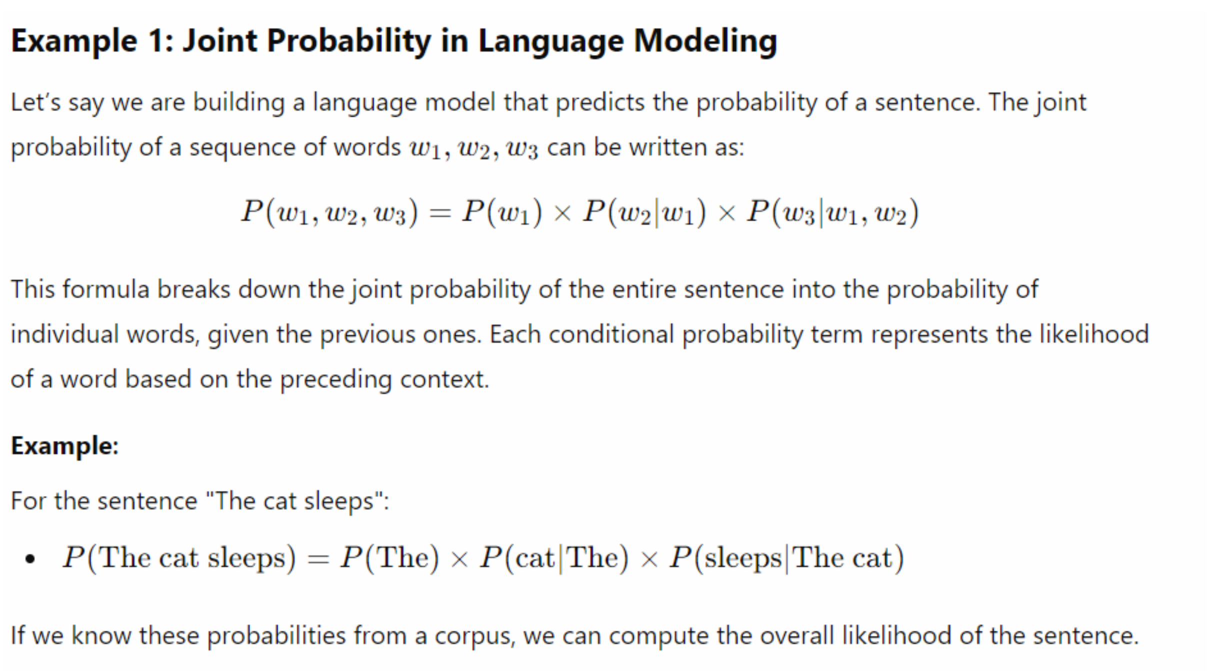
- $P(A)$ is the probability of event A occurring.
- $P(B|A)$ is the conditional probability of event B occurring given that A has occurred.

Joint Probability in NLP

In the field of **Natural Language Processing (NLP)**, joint probability is often used to model the likelihood of word sequences, which is fundamental to tasks like:

- **Language modeling**.
- **Speech recognition**.
- **Machine translation**.

Joint probability helps in determining how likely a sequence of words is in a particular language, and it serves as the basis for making predictions about what words might come next in a sentence.



Example 1: Joint Probability in Language Modeling

Let's say we are building a language model that predicts the probability of a sentence. The joint probability of a sequence of words w_1, w_2, w_3 can be written as:

$$P(w_1, w_2, w_3) = P(w_1) \times P(w_2|w_1) \times P(w_3|w_1, w_2)$$

This formula breaks down the joint probability of the entire sentence into the probability of individual words, given the previous ones. Each conditional probability term represents the likelihood of a word based on the preceding context.

Example:

For the sentence "The cat sleeps":

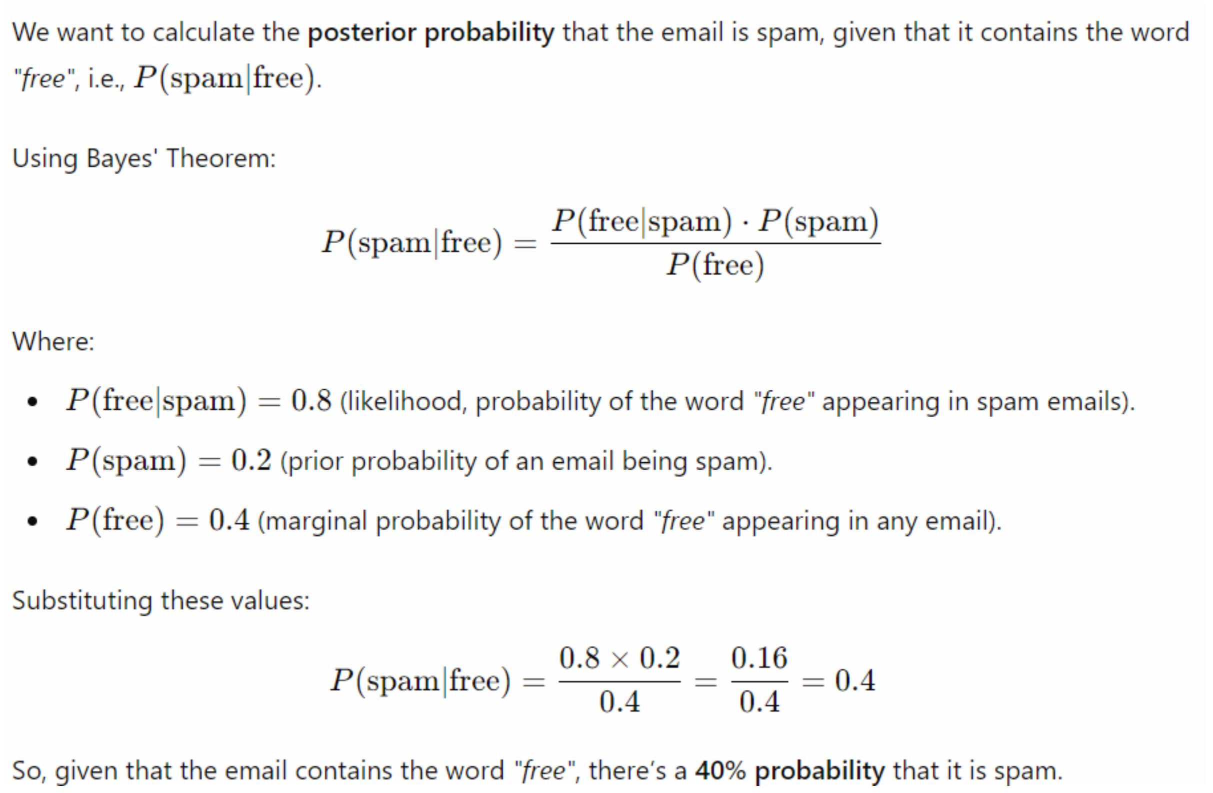
- $P(\text{The cat sleeps}) = P(\text{The}) \times P(\text{cat}|\text{The}) \times P(\text{sleeps}|\text{The cat})$

If we know these probabilities from a corpus, we can compute the overall likelihood of the sentence.

In **Natural Language Processing (NLP)**, Bayesian inference is commonly applied in **spam filtering**, where we want to classify an email as spam or not based on its content. Words in the email serve as evidence, and we calculate the probability that an email is spam given the presence of certain words.

Let's say the word *"free"* appears in an email. Based on previous data:

- 80% of emails containing the word *"free"* are spam.
- Only 20% of all emails are spam (this is our prior probability, P(spam)=0.2).
- The word *"free"* appears in 40% of all emails (spam and non-spam combined).



We want to calculate the **posterior probability** that the email is spam, given that it contains the word "*free*", i.e., $P(\text{spam}|\text{free})$.

Using Bayes' Theorem:

$$P(\text{spam}|\text{free}) = \frac{P(\text{free}|\text{spam}) \cdot P(\text{spam})}{P(\text{free})}$$

Where:

- $P(\text{free}|\text{spam}) = 0.8$ (likelihood, probability of the word "*free*" appearing in spam emails).
- $P(\text{spam}) = 0.2$ (prior probability of an email being spam).
- $P(\text{free}) = 0.4$ (marginal probability of the word "*free*" appearing in any email).

Substituting these values:

$$P(\text{spam}|\text{free}) = \frac{0.8 \times 0.2}{0.4} = \frac{0.16}{0.4} = 0.4$$

So, given that the email contains the word "*free*", there's a **40% probability** that it is spam.

Tokenization in NLP

Tokenization is the process of splitting a sequence of text (like a sentence or a document) into smaller pieces, called **tokens**. These tokens can be words, subwords, or characters, depending on the level of tokenization. Tokenization is often the first step in Natural Language Processing (NLP) tasks like text analysis, machine translation, and information retrieval.

In simpler terms, **tokenization** breaks down a stream of text into meaningful units that can be used for further processing, such as parsing or understanding the structure and meaning of the text.

Types of Tokenization

1. **Word Tokenization**:
    - The most common form of tokenization, where a text is split into words.
    - For example, given the sentence:
        - "The cat sat on the mat."
        - After tokenization: "The","cat","sat","on","the","mat","."

    Here, each word is treated as a token. Special characters like punctuation (e.g., the period in this case) can either be treated as separate tokens or ignored, depending on the tokenization approach.

    Common Libraries for Word Tokenization:

    - **NLTK**: nltk.word_tokenize()
    - **spaCy**: spacy.tokenizer()

2. **Subword Tokenization**:

- This breaks words into smaller units when dealing with languages or datasets where certain words are rare or unknown.
- For example, **Byte Pair Encoding (BPE)** or **WordPiece** is used in models like BERT. A word like *"running"* could be split into "run", "##ning".
- This helps in reducing the vocabulary size and handling unknown or rare words.

3. **Character Tokenization**:

- Here, the text is split into individual characters. This method is useful in tasks like speech recognition or handling languages like Chinese, where words may not be clearly separated by spaces.
- Example: "Hello" becomes "H","e","l","l","o"

4. **Sentence Tokenization**:

1. This splits text into sentences. It is useful in certain NLP tasks where the focus is on sentence-level understanding rather than word-level.
2. Example: "I love programming. It's fun!" becomes "Iloveprogramming.","It′sfun!"

Libraries:

NLTK: nltk.sent_tokenize()

Why Tokenization is Important?

1. **Preprocessing Step**: In NLP, algorithms generally work with structured inputs. Tokenization helps in breaking down the raw text into pieces that can be fed into models.
2. **Feature Extraction**: Tokens serve as the basic features for many NLP tasks, such as text classification, sentiment analysis, and machine translation. Without tokenization, the model would struggle to interpret the structure of the input text.
3. **Efficiency**: Tokenization reduces the complexity of text and allows models to focus on smaller, more meaningful units, improving both the efficiency and accuracy of NLP tasks.

Challenges in Tokenization

1. **Ambiguity**: In some languages or cases, it's difficult to define clear token boundaries. For instance, in "Let's go", should "Let's" be tokenized into "Let" and "'s"?
2. **Handling Punctuation**: Deciding whether punctuation marks should be treated as separate tokens or merged with adjacent words.
3. **Multilingual Tokenization**: Tokenization rules vary across languages. For example, Chinese text doesn't have spaces between words, so segmenting text into words (tokenization) becomes more complex.
4. **Dealing with Compound Words**: Languages like German have compound words (e.g., "Donaudampfschiffahrtsgesellschaft"), and determining where to split such words can be tricky.

Example of Tokenization in NLP

Consider a sentence:

Input Sentence: "She’s a great teacher."
Word Tokenization:

- After tokenization: "She","′s","a","great","teacher","."

Here, the contraction *"She’s"* is split into two tokens: "She" and "'s".

Tokenization in Modern NLP Models
In modern NLP models like **BERT**, **GPT**, and **T5**, subword tokenization methods like **WordPiece** or **Byte Pair Encoding (BPE)** are used. These methods break words into subword units, which allows the models to handle rare and unseen words more efficiently. For example:

- The word "unhappiness" might be tokenized as: "un", "##happi", "##ness".

This approach helps in reducing the model's vocabulary size and improves the generalization of unseen words.

5. **Whitespace Tokenization**

This is the simplest form of tokenization that splits the text based solely on spaces (whitespace) without considering punctuation.
Example:
Input Sentence:
"She loves programming in Python!"
Tokenized Output:
"She","loves","programming","in","Python!"

- The punctuation is kept attached to the word (e.g., "Python!").

Use Case:

- Useful when punctuation marks are meaningful or when simplicity is required.

6. **Regex Tokenization**

Regular expressions (regex) can be used for highly customized tokenization, where specific patterns in the text are used to split tokens. Regex tokenization allows for control over what is considered a token.
Example:
Input Sentence:
"I paid $10 for 2 apples."
Regex Tokenizer (splitting by non-alphabetic characters):
r'\W+'
Tokenized Output:
"I","paid","10","for","2","apples"

- Here, numbers are treated as tokens, and special characters like "$" are removed.

Use Case:

- Used in domain-specific tasks where traditional tokenization may not be sufficient, like extracting specific patterns from text (e.g., phone numbers, dates, etc.).

7. **Rule-Based Tokenization**

In **rule-based tokenization**, predefined linguistic rules are used to split text into tokens. This method is useful for languages that do not use spaces between words (e.g., Chinese,

Japanese), or when more complex word boundaries need to be detected (e.g., contractions, possessives).
Example:
Input Sentence:
"Let's go!"
Tokenized Output:
"Let","'s","go","!"
Here, the tokenization splits the contraction "Let's" into two tokens: "Let" and "'s".
Use Case:

- Languages where word boundaries are not clearly defined by spaces, such as **Chinese, Japanese, Thai**, etc.

8. **Morpheme-Based Tokenization (Morphological Tokenization)**

Morpheme-based tokenization involves splitting words into their morphemes, which are the smallest grammatical units that carry meaning. This is particularly useful for morphologically rich languages (e.g., Turkish, Finnish) where a word may have many suffixes or prefixes.
Example:
Input Word:
"unbelievable"
Tokenized Output:
"un","believe","able"

- Here, the word "unbelievable" is split into its morphemes: the prefix "un", the root "believe", and the suffix "able".

Use Case:

- Applied in morphological analysis to study the structure of words.

Introduction to Morphology

Morphology is the branch of linguistics that studies the structure and formation of words. It focuses on how words are built from smaller meaningful units called **morphemes**. Morphemes are the smallest grammatical units in a language that carry meaning.
Morphology plays a significant role in natural language processing (NLP) because understanding the internal structure of words helps computers process and interpret language more effectively. Morphology helps in various NLP tasks such as tokenization, part-of-speech tagging, lemmatization, and more.

Key Concepts in Morphology

1. **Morphemes**:
    - A morpheme is the smallest unit of meaning in a word.
    - Morphemes can be divided into two categories:
        - **Free Morphemes**: These can stand alone as independent words. Example: *"book"*, *"run"*.
        - **Bound Morphemes**: These cannot stand alone and must be attached to a free morpheme. Example: *"-ed"* (past tense), *"-s"* (plural).
2. **Types of Morphology**:

    - **Inflectional Morphology**: Deals with the modification of words to express different grammatical categories such as tense, case, mood, or number. It does not change the word's core meaning or part of speech.
        - Example: *"run"* → *"runs"*, *"run"* → *"running"*.
    - **Derivational Morphology**: Involves creating new words by adding prefixes or suffixes to existing words, often changing their meaning or part of speech.
        - Example: *"happy"* → *"unhappy"*, *"teach"* → *"teacher"*.

3. **Root, Stem, and Affixes**:

- **Root**: The core part of a word that contains its basic meaning.
    - Example: In *"unhappiness"*, the root is *"happy"*.
- **Stem**: The form of the word that can take on inflectional morphemes.
    - Example: In *"working"*, the stem is *"work"*.
- **Affix**: A morpheme that is attached to a root or stem to modify its meaning. Affixes are either:
    - **Prefix**: Comes before the root (e.g., *"un-"* in *"unhappy"*).
    - **Suffix**: Comes after the root (e.g., *"-ness"* in *"happiness"*).

Types of Morphemes

1. **Free Morphemes**:
    - These are words that can function independently in a sentence.
    - Example: *"dog"*, *"book"*, *"run"*.
2. **Bound Morphemes**:
    - These must be attached to another morpheme to convey meaning.
    - Example: *"-s"* (plural), *"-ed"* (past tense).

Inflectional Morphology

Inflectional morphology involves the modification of words to express different grammatical features without changing the word's core meaning or its part of speech.

Common Inflectional Morphemes:

- **Plural**: *"-s"* (e.g., *"dogs"*)
- **Past Tense**: *"-ed"* (e.g., *"played"*)
- **Present Participle**: *"-ing"* (e.g., *"playing"*)
- **Comparative**: *"-er"* (e.g., *"faster"*)
- **Superlative**: *"-est"* (e.g., *"fastest"*)

Example of Inflectional Morphology:

- *"play"* → *"playing"* (inflected for present participle)
- *"play"* → *"played"* (inflected for past tense)

Derivational Morphology

Derivational morphology creates new words by adding affixes to a base word, often changing its meaning or part of speech.

Example of Derivational Morphology:

- *"happy"* → *"unhappy"* (prefix *"un-"* changes the meaning to the opposite)
- *"teach"* → *"teacher"* (suffix *"-er"* changes the word from a verb to a noun)
- *"create"* → *"creation"* (suffix *"-tion"* changes the verb into a noun)

Derivational morphology often results in a more significant change than inflectional morphology, as it can lead to the formation of completely new words.

Morphological Parsing
Morphological parsing involves breaking down a word into its constituent morphemes. This is useful in various NLP tasks, especially in languages with rich morphology.
Example of Morphological Parsing:
For the word *"unhappiness"*:

- **un-**: Prefix (negative)
- **happy**: Root (basic meaning of the word)
- **-ness**: Suffix (turns the adjective into a noun)

Thus, *"unhappiness"* is parsed as "un−","happy","−ness".

Importance of Morphology in NLP

1. **Text Preprocessing**: Morphological analysis helps in stemming and lemmatization, which are important preprocessing steps in NLP to reduce words to their base forms.
2. **Machine Translation**: Understanding morphological structures helps machines translate languages more accurately, especially for morphologically rich languages like Turkish or Finnish.
3. **Speech Recognition**: Morphology helps in improving speech-to-text systems by understanding the variations in word forms.
4. **Named Entity Recognition (NER)**: Recognizing entities such as person names, locations, and organizations benefit from morphology, especially in morphologically complex languages.

Stemming
Stemming is a natural language processing (NLP) technique that reduces a word to its **stem** or base form, typically by stripping affixes (such as prefixes and suffixes). Unlike **lemmatization**, which reduces words to their dictionary form (lemma), stemming is a more rule-based and heuristic-driven process that often leads to non-lexical stems, meaning the resulting stem might not be a valid word.
The goal of stemming is to group together different forms of a word so that they can be treated as the same item in tasks like **information retrieval**, **search engines**, or **text mining**.

How Stemming Works
Stemming works by applying rules to remove known suffixes and prefixes from words. The resulting word, called the **stem**, may not always be a valid dictionary word but captures the core meaning of the original word.
For example:

- **Word**: *"running"*
- **Stem**: *"run"*

Common stemming rules in English involve removing suffixes like *-ing*, *-ed*, *-ly*, *-es*, etc.

Types of Stemming Algorithms

There are several types of stemming algorithms, with the **Porter Stemmer** being one of the most widely used. Other popular algorithms include the **Snowball Stemmer** and the **Lancaster Stemmer**.

1. **Porter Stemmer**:
    - One of the oldest and most widely used stemming algorithms.
    - It applies a series of rules and transformations to remove common suffixes.
2. **Snowball Stemmer**:
    - An improvement on the Porter Stemmer, more flexible and easier to modify for different languages.
3. **Lancaster Stemmer**:
    - A more aggressive stemming algorithm, which can result in very short stems, often leading to over-stemming.

Example of Stemming

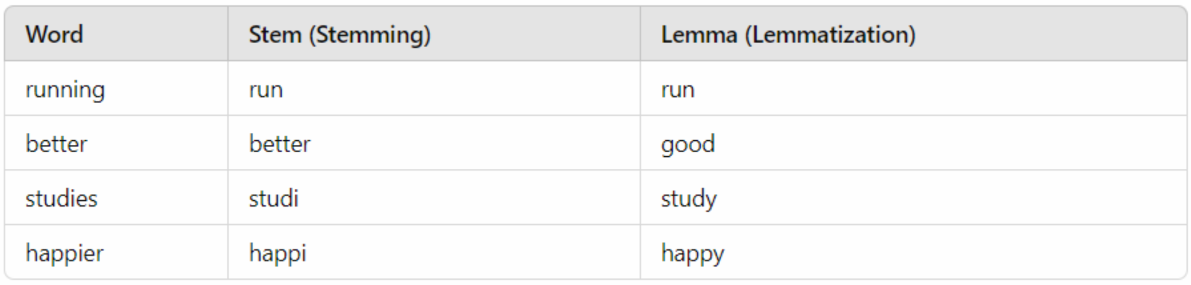
Let's apply stemming to a few different words:

1. **Word**: *"playing"*
    - **Stem**: *"play"*
2. **Word**: *"played"*
    - **Stem**: *"play"*
3. **Word**: *"player"*
    - **Stem**: *"play"*
4. **Word**: *"happily"*
    - **Stem**: *"happi"*

Here, the word *"play"* remains the same after stemming, but the word *"happily"* is reduced to *"happi"*. While *"happi"* is not a valid word, it still captures the core meaning of the original word.

Stemming vs. Lemmatization

- **Stemming**:
    - Strips affixes according to rules.
    - Often results in non-lexical forms (e.g., *"happi"* instead of *"happy"*).
    - Computationally cheaper and faster than lemmatization.
- **Lemmatization**:
    - Reduces words to their base form (lemma) using a dictionary.
    - Produces valid dictionary words (e.g., *"better"* → *"good"*).
    - Requires more computational resources than stemming.



Word	Stem (Stemming)	Lemma (Lemmatization)
running	run	run
better	better	good
studies	studi	study
happier	happi	happy

Applications of Stemming

1. **Search Engines**:

   - Search engines use stemming to treat different forms of a word (e.g., *"run"*, *"running"*, *"runner"*) as equivalent. This ensures that a query for *"run"* will return documents containing *"running"* and *"runner"* as well.
2. **Information Retrieval**:
   - In information retrieval systems (such as databases and document searches), stemming helps improve the recall of relevant documents by matching word variations.
3. **Text Classification**:
   - Stemming helps in reducing the dimensionality of textual data by consolidating multiple forms of the same word into a single representation.
4. **Text Mining**:
   - In text mining tasks like sentiment analysis or topic modeling, stemming helps group different word forms, leading to more concise and generalizable results.

Q1: Apply stemming to the following words: *"jumping"*, *"happily"*, *"studies"*, and *"players"*.
Answer:
- **"jumping"** → *"jump"*
- **"happily"** → *"happi"*
- **"studies"** → *"studi"*
- **"players"** → *"player"*

Porter Stemmer Algorithm
The **Porter Stemmer Algorithm** is one of the most widely used stemming algorithms in natural language processing (NLP). Developed by Martin Porter in 1980, it applies a series of heuristic rules to strip common suffixes from English words, thus reducing them to their "stem" form. The goal is to remove morphological affixes and normalize words to their base form for tasks like search engines, information retrieval, and text mining.

Porter Stemmer Algorithm Steps and Rules
The algorithm consists of five main steps, each containing a series of rules that are applied sequentially to a word. These rules focus on stripping suffixes, such as *-ing*, *-ed*, *-ly*, etc., to achieve the stem. The rules are based on conditions about the structure of the word, particularly its **measure** (or "m-value"), which helps in determining whether the word is modified or not.
Measure (m-value)
The measure (m) is defined as the number of **VC sequences** (vowel-consonant) in the word. For example, in the word *"happy"*, the sequence is VC, so the measure (m) = 1.

Step-by-Step Breakdown of the Porter Stemmer Algorithm
Step 1: Removing Plural and Past Participle Suffixes
- **Step 1a**: Remove the plural suffix *"-s"*.
  - If the word ends in *"sses"*, replace it with *"ss"*.
  - If the word ends in *"ies"*, replace it with *"i"*.
  - If the word ends in *"ss"*, leave it unchanged.
  - If the word ends in *"s"*, remove it (but only if there is a vowel in the word).

Example:

- *"caresses" → "caress"*
- *"ponies" → "poni"*
- *"cats" → "cat"*
- **Step 1b**: Remove *"-ed"* or *"-ing"* if the word contains a vowel before the suffix.
    - Replace *"-ed"* with the base form.
    - Replace *"-ing"* with the base form.

Example:

- *"hoped" → "hope"*
- *"dancing" → "danc"*

After removing *"-ed"* or *"-ing"*, if the resultant word ends with:

- *"at"*, *"bl"*, *"iz"*, add an *"e"*.
- Double the last consonant if the word ends in a short vowel + consonant (e.g., *"hop" → "hopp"*).
- If the word ends in a consonant and a vowel followed by a consonant and has measure = 1, add *"e"* (e.g., *"plan" → "plane"*).

Step 2: Replace Double Suffixes

- Replace common double suffixes like *"-ational"*, *"-izer"*, and *"-fulness"* with shorter forms.

Rules:

- *"ational" → "ate"*
- *"izer" → "ize"*

Example:

- *"relational" → "relate"*
- *"optimizer" → "optimize"*

Step 3: Replace Other Common Suffixes

- Replace suffixes like *"-icate"*, *"-ful"*, *"-ness"*, and *"-ness"* with simpler forms.

Rules:

- *"icate" → "ic"*
- *"ness"* → Remove entirely.

Example:

- *"fortunate" → "fortun"*
- *"goodness" → "good"*

Step 4: Remove Suffixes Based on Measure (m > 1)

- Remove suffixes like *"-ant"*, *"-ence"*, *"-ment"* if the remaining word has measure (m) > 1.

Rules:

- If the word ends with *"ement"*, *"ant"*, *"ence"*, remove them if the word has a measure greater than 1.

Example:

- *"dependent" → "depend"*

Step 5: Remove the Final *"-e"* (if measure > 1)

- Remove a trailing *"-e"* if the word's measure (m) > 1. If m = 1 and the word ends with *"-e"*, remove it unless the word is "short".

Example:

- *"probate"* → *"probat"*
- *"rate"* → *"rat"*

Example of the Porter Stemmer in Action

Let's break down the Porter Stemmer's application on the word **"relational"**:

1. **Step 1**: No changes in plural or past participle forms.
2. **Step 2**: Replace *"ational"* with *"ate"*.
    - **Result**: *"relate"*

Q1) Apply the Porter Stemmer to the word *"happiness"*.

Answer:

- **Step 1**: No changes.
- **Step 2**: Remove *"-ness"*.
    - **Result**: *"happi"*