**Module 1**

1. **What is natural language processing?**

* Natural Language Processing (NLP) is a field of artificial intelligence (AI) that focuses on the interaction between computers and humans through natural language. The goal of NLP is to enable computers to understand, interpret, and generate human language in a way that is both meaningful and useful.
* NLP combines computational linguistics—rule-based modeling of human language—with machine learning, statistical, and deep learning models. By analyzing and understanding the structure and meaning of language, NLP allows for more natural and effective human-computer interaction.

This involves a range of tasks and applications, including:

1. **Text and Speech Recognition**: Converting spoken language into text (speech-to-text) and vice versa (text-to-speech).
2. **Language Translation**: Automatically translating text or speech from one language to another, such as Google Translate.
3. **Sentiment Analysis**: Determining the sentiment or emotional tone behind a series of words, useful in understanding customer feedback, social media monitoring, etc.
4. **Text Summarization**: Creating concise summaries of longer texts while preserving the main points and overall meaning.
5. **Chatbots and Virtual Assistants**: Enabling interactive communication with users, such as with Siri, Alexa, or customer service bots.
6. **Information Retrieval**: Finding relevant information from large datasets, such as search engines and database queries.
7. **Named Entity Recognition (NER)**: Identifying and classifying key elements in text, such as names of people, organizations, dates, and locations.
8. **Part-of-Speech Tagging**: Identifying parts of speech in text (e.g., nouns, verbs, adjectives).
9. **Machine Translation**: Translating text from one language to another using algorithms.
10. **Speech Generation**: Producing human-like speech from text, used in applications like voice assistants and automated customer service.
11. **Need of NLP?**

The need for Natural Language Processing (NLP) arises from the growing amount of textual and spoken data generated daily and the necessity to derive meaningful insights and interactions from this data. Here are several key reasons why NLP is essential:

1. **Managing and Analyzing Large Volumes of Text Data:**

* With the explosion of data on the internet, social media, and digital communications, there's an overwhelming amount of unstructured text data. NLP helps in processing and making sense of this data efficiently.

1. **Enhancing Human-Computer Interaction:**

* NLP enables more natural and intuitive interactions between humans and computers, allowing users to communicate with machines using everyday language rather than specialized commands.

1. **Automating Routine Tasks:**

* Tasks like sorting emails, responding to customer queries, and managing documentation can be automated using NLP, increasing efficiency and freeing up human resources for more complex tasks.

1. **Improving Customer Service:**

* Chatbots and virtual assistants powered by NLP can provide immediate responses to customer inquiries, handle routine requests, and offer 24/7 support, enhancing customer satisfaction and operational efficiency.

1. **Facilitating Multilingual Communication:**

* NLP technologies like machine translation enable real-time translation of text and speech, breaking down language barriers and supporting global communication and collaboration.

1. **Extracting Insights from Data:**

* Sentiment analysis, topic modeling, and other NLP techniques help organizations understand public opinion, market trends, and customer feedback, which can inform business strategies and decision-making.

1. **Enhancing Accessibility:**

* NLP aids in creating accessible technologies, such as speech-to-text for individuals with hearing impairments and text-to-speech for those with visual impairments, making digital content more inclusive.

1. **Improving Information Retrieval and Knowledge Management:**

* Advanced search algorithms and information extraction tools powered by NLP can locate and organize relevant information from vast datasets quickly and accurately.

1. **Supporting Healthcare and Legal Fields:**

* In healthcare, NLP can be used to analyze patient records, assist in diagnostics, and manage clinical documentation. In legal fields, it helps in reviewing contracts, legal research, and extracting relevant information from legal texts.

1. **Enabling Advanced Research:**

* Researchers across various fields use NLP to analyze large corpora of text, derive patterns, and generate new insights, advancing knowledge in disciplines such as linguistics, social sciences, and more.

1. **Explain different levels of NLP?**

**Level of NLP**

Natural Language Processing (NLP) operates at multiple levels of linguistic analysis to understand and generate human language. The analysis of NL is broken down into various broad levels such as phonological, morphological, lexical, syntactic, semantic, pragmatic and discourse analysis.

1. **Phonology**

* This level is applied only if the input is a speech. Input is acoustic waveform and output is string of words.
* It deals with speech recognition, sound structure of spoken language and generation. That is interpretation of speech sounds within and across words.
* Speech sound might give a big hint about the meaning of a word or a sentence.
* The area of computational linguistic that deals with speech analysis is computational phonology.

1. **Morphological Analysis**

* Morphology is study of internal structure of words. Given a particular word in a language, what are the different meaningful units it is made up of and each small unit is called a morpheme. It involves tasks such as lemmatization (reducing words to their base or root form) and stemming (stripping suffixes from words).
* cat: stem cats: cat + s unhappy happily unhappily
* cats: N +PL cat: N+SG sort: V+SG sorts: V+PL sort: N+PL
* Example: Breaking down "running" into "run" (root) and "ing" (suffix).
* Computational tool to perform morphological parsing is finite state transducer.

1. **Lexical Analysis**

* It deals with understanding of everything about distinct words according to their position in the speech, their meanings, and their relation to other words. It involves task such as Word sense disambiguation (determining which meaning of a word is used in context) and managing lexical resources like dictionaries and thesauruses.
* It identifies and analyse the structure of words with respect to their lexical meaning and part-of-speech. Lexicon is a dictionary. Lexicon of a language means the collection of words and phrases in a language. Validity of word is checked according to lexicon.
* Example: Identifying that "bank" in "river bank" refers to the side of a river rather than a financial institution.

1. **Syntactic Analysis (Parsing) –**

* It involves analysis of words in the sentence for grammar and arranging words in a manner that shows the relationship among the words. The sentence such as “The school goes to boy” is rejected by English syntactic analyzer.
* Syntax refers to the study of formal relationships between words of sentences. Validity of a sentence is checked according to rules of grammar.
* To perform syntactic analysis, the knowledge of grammar and parsing techniques is required.
* Grammar is formal specification of rules allowable in the language. Parsing is a method of analysing a sentence to determine its structure according to the grammar. CFG is used for syntactic analysis.
* Two basic parsing techniques are top-down parsing and bottom-up parsing.
* I eat banana. (subject verb object)
* I banana eat. (subject object verb)

1. **Semantic Analysis**

* Semantics deals with the meaning of words, phrases, sentences. Meaning of the sentences is understood in this phase. It draws the exact meaning or the dictionary meaning from the text.
* The text is checked for meaningfulness. It is done by mapping syntactic structures and objects in the task domain. The semantic analyzer disregards sentence such as “hot ice-cream”.

e.g.

* She eats banana
* Machine eats banana.

1. **Pragmatic:**

* During this, what was said is re-interpreted on what it meant. It involves deriving those aspects of language which require real world knowledge.
* Explains how extra meaning is read into texts without being encoded in them. This requires much world knowledge, including the understanding of intentions, plans, and goals.
* Consider the following 2 sentences:

The city counsellors refused the demonstrators a permit because they feared violence.

The city counsellors refused the demonstrators a permit because they advocated revolution.

* The meaning of “they” in the 2 sentences is different. To figure out the difference, world knowledge in knowledge bases and inferencing modules should be utilized.

1. **Discourse Level:**

* The meaning of any sentence depends upon the meaning of the sentence just before it. In addition, it also brings about the meaning of immediately succeeding sentence.
* Focuses on the properties of the text as a whole that convey meaning by making connections between components sentences.

1. I only like travelling to Europe. So I submitted a paper to ACL.
2. I only like travelling to Europe. Nevertheless, I submitted a paper to ACL.

* In example (1), you can infer ACL-is-in-Europe-this-year and in (2) you can infer the negation of that.

1. **Describe Generic NLP system with a diagram?**

A diagram of a language processing process

Description automatically generated

**There are 3 main components:**

* The input is where the raw text, typed entries, message and speech enters the system. In the case of speech, speech recogniser is used which converts the input speech into string of words. Speech recognition is important because the model can't understand spoken language on its own.
* These inputs are then given for preprocessing.

**Preprocessing:**

* Once the input data is received, it goes through preprocessing. This step involves cleaning and formatting the text data to make it suitable for further analysis. Preprocessing task may include:

1. **Tokenization:** Segmenting text into smaller units (tokens) such as words or phrases for analysis, essential for many NLP tasks to operate on discrete elements of text efficiently.
2. **Lowercasing:** Converting all text to lowercase to ensure uniformity in word representation and prevent duplication of words due to case variations.
3. **Removing Punctuation:** Eliminating punctuation marks to focus on the semantic content of the text and avoid noise in downstream processing steps.
4. **Removing Stopwords:** Filtering out common words like "the" and "and" that contribute little semantic value, reducing data dimensionality and improving algorithm efficiency.
5. **Stemming and Lemmatization:** Simplifying words to their root forms (stemming) or dictionary entries (lemmatization) to standardize variations and enhance text analysis accuracy.
6. **Handling Special Characters and Numbers:** Managing special characters, emojis, and numerical digits appropriately, either by removal, replacement, or conversion, to maintain text integrity and relevance in analysis.

**Natural Language Processing (NLP):** After preprocessing, the text data is passed to the NLP component, which performs various linguistic tasks to understand the meaning and structure of the text. Some common NLP tasks include:

1. **Tokenization**: Breaking text into smaller units such as words, phrases, or sentences.
2. **Part-of-Speech (POS) Tagging:** Assigning grammatical labels (e.g., noun, verb, adjective) to each word in the text.
3. **Syntax:** Analyze sentence grammar to simplify subsequent processing and detect nuances.
4. **Semantics:** Extract meaning from text, considering context and language use.
5. **Discourse and Pragmatics**: Handle complex context and language phenomena like pronoun references and elliptical sentence fragments.
6. **AI Algorithms and Models:** Utilize deep learning techniques such as recurrent neural networks (RNNs) and transformer models like BERT and GPT for advanced NLP tasks.
7. **Named Entity Recognition (NER):** Identifying and classifying named entities such as person names, locations, organizations, etc., in the text.
8. **Parsing:** Analyzing the grammatical structure of sentences to determine their syntactic relationships.
9. **Sentiment Analysis:** Determining the sentiment or opinion expressed in the text (positive, negative, neutral).
10. **Topic Modeling:** Identifying the main topics or themes present in a collection of text documents.
11. **Word Embeddings:** Representing words as dense numerical vectors in a high-dimensional space, capturing semantic relationships between words.
12. **Machine Translation:** Translating text from one language to another.
13. **Text Summarization:** Generating concise summaries of longer text documents.

**Meaning Representation**: Once the NLP tasks are performed, the system needs to derive meaning or representation from the processed text data. This could involve creating semantic representations, knowledge graphs, or other structured formats that capture the essential information conveyed by the text.

**Output Processing**: The meaning or representation derived from the input text is then passed to the output processor. This component is responsible for representation the predict made by algorithm or model in the required format.

1. **What are the challenges in NLP?**
2. **Contextual words and phrases and homonyms:**

* The same words and phrases can have diverse meanings according to the context of a sentence and many words have the exact same pronunciation but completely different meanings.
* For example:

I ran to the store because we ran out of milk.

Can I run something past you really quick?

The house is looking really run down.

In the above three sentences the meaning of the run is different according to the context

* Homonyms means the pronunciation of two or more words is same but have different meaning. For example, their and there, right and write. This will create problem in question answering and speech-to-text applications.

1. **Synonyms:**

* Different words can mean the same thing but are used in various contexts. For example, "small," "little," "tiny," and "minute" all mean similar things but can have slight nuances in meaning.

1. **Irony and Sarcasm:**

* Sarcastic and ironic statements often mean the opposite of what the words literally say, making it hard for machines to understand the true intent. For example, saying "Great job!" sarcastically when someone makes a mistake.

1. **Ambiguity:**

* Multiple Interpretations: Sentences can have more than one meaning. For example, "I saw the man with the telescope" can mean seeing a man through a telescope or a man holding a telescope. This includes:
  + Lexical Ambiguity: Multiple meanings for a word.
  + Syntactic Ambiguity: Multiple ways to parse a sentence.
  + Semantic Ambiguity: Multiple interpretations of a sentence's meaning.

1. **Errors in Text or Speech:**

* Misspellings and Mispronunciations: Typos, autocorrect errors, and misused words can confuse text analysis. Spoken language adds challenges like different accents and stammers, which machines struggle to interpret correctly.

1. **Idioms and slang**

* Phrases that don't mean what the words literally say, like "kick the bucket" for "die," are hard for models to understand. Informal phrases, expressions, idioms, and culture-specific lingo present a number of problems for NLP. Slang changes rapidly and can be region-specific, adding another layer of complexity.

1. **Domain-Specific Language**

* Different business and industries often use very different language.

A model trained for healthcare terminology would struggle with legal jargon, and vice versa.

1. **Low-Resource Languages:**

* Less Common Languages: NLP models are mainly built for widely spoken languages, leaving out many lesser-known languages with limited data available. This is a significant issue for many African languages and others with fewer speakers.

1. **What are the ambiguities in NLP design?**

Ambiguity, generally used in natural language processing, can be referred as the ability 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.
* There are various types of lexical ambiguity, including:
* **Polysemy:** A single word having multiple related meanings. For example, the word "bank" can refer to a financial institution or the side of a river.
* **Homonymy:** Different words with the same spelling or pronunciation but different meanings. For instance, "bat" can refer to a flying mammal or a piece of sports equipment.
* **Heteronymy:** Words with the same spelling but different pronunciations and meanings. An example is "tear," which can mean to rip or a drop of liquid from the eye.

1. **Syntactic Ambiguity**

* This kind of ambiguity occurs when a sentence is parsed in different ways.
* types of syntactic ambiguity:
* **Structural Ambiguity:** This occurs when the syntactic structure of a sentence allows for multiple interpretations. For example:
* "I saw the man with the telescope." Is the man holding the telescope or did the speaker use the telescope to see the man?
* **Attachment Ambiguity:** In sentences with multiple clauses, the attachment of a phrase or clause to one part of the sentence over another can lead to ambiguity. For instance:
* "I told my friend I would help." Did the speaker tell their friend that they would help, or did they tell their friend about their intention to help someone else?
* **Modifier Attachment Ambiguity:** This occurs when a modifier, such as an adjective or adverb, can be associated with more than one word in a sentence. For example:
* "She almost told him everything." Did she almost tell him everything, or did she tell him almost everything?
* **Coordination Ambiguity:** Ambiguity can arise in sentences with coordination, where multiple phrases are joined by conjunctions. For example:
* "She likes chocolate and vanilla ice cream." Does she like both flavors, or does she like chocolate ice cream and vanilla ice cream separately?

1. **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”.

1. **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.

1. **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 and the context does not provide the information needed to clarify the statement. Information is missing and must be inferred.
* 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. **List out five applications in NLP?**
2. **Sentiment Analysis:**

* **Description:** Sentiment analysis, also known as opinion mining, involves determining the sentiment expressed in a piece of text. It categorizes text as positive, negative, or neutral based on the emotions and opinions conveyed.
* **Example Use Case:** Analyzing social media posts to gauge public opinion about a product or service. Companies use sentiment analysis to understand customer feedback and sentiment towards their brand.

1. **Chatbots and Conversational Agents:**

* **Description:** Chatbots and conversational agents are AI-powered systems designed to interact with users in natural language. They simulate human conversation and can answer questions, provide information, and assist users with various tasks.
* **Example Use Case:** Customer service chatbots deployed on websites to handle customer inquiries and provide support. Virtual assistants like Siri, Alexa, and Google Assistant are also examples of conversational agents.

1. **Named Entity Recognition (NER):**

* **Description:** Named Entity Recognition (NER) is a task in NLP that involves identifying and classifying named entities in text into predefined categories such as names of people, organizations, locations, dates, and more.
* **Example Use Case:** Extracting important information from news articles, such as identifying names of politicians, companies, and locations. NER is also used in medical texts to identify names of diseases, medications, and procedures.

1. **Machine Translation:**

* **Description:** Machine translation is the process of automatically translating text from one language to another using NLP techniques. It involves analyzing the input text, understanding its meaning, and generating an equivalent translation in the target language.
* **Example Use Case:** Online translation services like Google Translate and DeepL that translate text between multiple languages. Machine translation is also used by global businesses to translate documents, websites, and communication materials.

1. **Text Classification:**

* **Description:** Text classification is the task of categorizing text documents into predefined classes or categories based on their content. It involves training machine learning models to classify text based on features extracted from the text data.
* **Example Use Case:** Spam detection in email systems, where incoming emails are classified as either spam or non-spam based on their content. Text classification is also used in sentiment analysis, topic categorization, and content moderation.

1. **Information Extraction:**

* **Description:** Information extraction involves extracting structured information from unstructured text data. It identifies and extracts specific entities, relationships, and facts from text documents.
* **Example Use Case:** Extracting key information from resumes for recruitment purposes, such as names, contact details, education, and work experience. Information extraction is also used in extracting data from legal documents, financial reports, and scientific articles.

1. **Question Answering Systems:**

* **Description:** Question Answering (QA) systems are designed to automatically answer questions posed in natural language. They analyze the input question, search for relevant information, and generate a concise and accurate answer.
* **Example Use Case:** Search engines like Google that provide direct answers to user queries in addition to search results. QA systems are also used in customer support chatbots to answer frequently asked questions and provide assistance.

**Module 3**

1. **Write a note on N gram language model. How to overcome zero probability?** **Explain perplexity of any language mode**

* An n-gram is a contiguous sequence of n items from a given sample of text or speech. The items can be phonemes, syllables, letters, words or base pairs according to the application. The n-grams typically are collected from a text or speech corpus.
* An N-gram language model predicts the probability of a given N-gram within any sequence of

words in the language. If we have a good N-gram model, we can predict p(w | h) - what is the

probability of seeing the word w given a history of previous words h - where the history

contains n-1 words.

The estimation of probability of the of a sentence is achieved by decomposing sentence

probability into a product of conditional probabilities.

A math equations on a white background

Description automatically generated

**apply chain rule as follows:**

A white background with black text

Description automatically generated

**Probability of words in sentences:**

* **Unigram(1-gram):** No history is used.
* **Bi-gram(2-gram):** One word history. Probability is
* **Tri-gram(3-gram):** Two words history. Probability is
* **Four-gram(4-gram):** Three words history

A white background with black text

Description automatically generated

A math equations and numbers

Description automatically generated with medium confidence

A white background with black text

Description automatically generated

A maths and math equations

Description automatically generated with low confidence

A white background with black text

Description automatically generated

A white paper with black text

Description automatically generated

**Techniques to Overcome Zero Probability**

1. **Log Probability**

* Probabilities of sequences can become very small, especially when multiplying many probabilities together, leading to underflow in computer arithmetic. Using log probabilities helps to prevent this issue and makes calculations more manageable.
* **How It Works:**

Convert the probabilities to log scale using the natural logarithm.

Instead of multiplying probabilities, sum their log probabilities.

Log probabilities are always negative, and their sum is less likely to underflow.

* **Formula:** If 𝑃(𝑤1,𝑤2,...,𝑤𝑛)P(w1​,w2​,...,wn​) is the probability of a sequence of words,

the log probability is:

1. **Laplace Smoothing (Add-One Smoothing)**

* Laplace smoothing is a simple technique to handle zero probabilities by adding a small constant (typically 1) to each count. This ensures that no probability is ever zero.
* **How It Works:**

Add one to the count of each N-gram.

Adjust the denominator to account for the added counts.

* **Formula:** For a bigram model: ​

where V is the size of the vocabulary.

For a trigram model:

* **Benefits:**
* Simple to implement.
* Provides non-zero probabilities for unseen N-grams.

**Example**

A screenshot of a computer

Description automatically generated

A screenshot of a math equation

Description automatically generated with medium confidence

1. **Perplexity**

* Perplexity is a metric used to evaluate the performance of a language model. It measures how well a probability distribution predicts a sample and can be used to compare different models or smoothing techniques.
* **How It Works:**

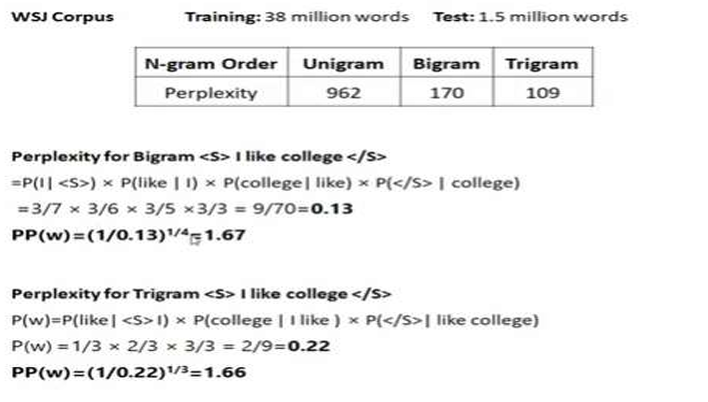
Compute the inverse probability of the test set, normalized by the number of words.

Lower perplexity indicates a better model (i.e., the model predicts the test data more accurately).

* **Formula:** Given a test set 𝑊= 𝑤1, 𝑤2,...,𝑤𝑁, the perplexity of a language model is:
* Lower the value of perplexity: Better Model
* More value of perplexity: Confused for prediction.

**Application:**

* Compare models with and without smoothing.
* Validate improvements made by different techniques in handling zero probabilities.

****

1. **Differentiate between bigram and trigram**

|  |  |  |
| --- | --- | --- |
| **Aspect** | **Bigram Model** | **Trigram Model** |
| **Definition** | Considers the probability of a word given the previous one word | Considers the probability of a word given the previous two words |
| **N** | 2 (two consecutive words) | 3 (three consecutive words) |
| **Context** | Considers only one preceding word for prediction | Considers two preceding words for prediction |
| **Example Sequence** | "I love natural language processing" -> Bigrams: "I love", "love natural", "natural language", "language processing" | "I love natural language processing" -> Trigrams: "I love natural", "love natural language", "natural language processing" |
| **Probability Calculation** |  |  |
| **Data Sparsity** | Less sparse compared to trigram | More sparse than bigram |
| **Complexity** | Simpler model | More complex model |
| **Storage Requirements** | less | more |
| **Prediction Accuracy** | Lower accuracy due to limited context | Higher accuracy with more context, but dependent on data quality and quantity |
| **Training Data Requirement** | less | more |
| **Use Cases** | Basic text processing tasks, simple autocomplete systems | More advanced text processing tasks, better performance in predictive text and language generation |
| **Handling Ambiguity** | Less effective at disambiguating words in different contexts | More effective at disambiguating words in different contexts due to longer context |

1. **Explain Porter Stemmer?**

* **Stemming** is the process of reducing a word to its word stem that affixes to suffixes and prefixes or to the roots of words known as a lemma. For example : words such as "Likes", "liked", " likely" and "liking" will be reduced to "like" after stemming.
* The process of determining the root of words is known as stemming.

**(A) Over-stemming**

* Over-stemming is the term we use to describe the process through which our algorithm stems many unrelated words to the same root.
* Despite sharing a core word and being related, the words "universal," "university," and "universe" have quite different meanings.
* These terms should return quite distinct search results when we enter them into a reputable search engine, and they shouldn't be considered synonyms. Such a mistake is known as a false positive.

A diagram of a flowchart

Description automatically generated

**(B) Under-stemming**

The converse of that phenomenon, known as under-Stemming, occurs when numerous words are not stemmed from a single root even when they should. A former university student is referred to as a "alumnus" and is often a male individual. The word "alumni" refers to several former students of an institution, whereas "alumnae" is the feminine equivalent.

A close up of a paper

Description automatically generated

**Algorithm:**

To present the suffix stripping algorithm in its entirety we will need a few definitions. A consonant in a word is a letter other than A, E, I, O or U, and other than Y preceded by a consonant. So, in TOY the consonants are T and Y, and in SYZYGY they are S, Z and G. If a letter is not a consonant it is a vowel.

A consonant will be denoted by c, a vowel by v. A list ccc ... of length greater than 0 will be denoted by C, and a list vvv ... of length greater than 0 will be denoted by V. Any word, or part of a word, therefore has one of the four forms :

* CVCV ... C
* CVCV ... V
* VCVC ... C
* VCVC , .. V

These may all be represented by the single form

[C]VCVC ... [V]

where the square brackets denote arbitrary presence of their contents. Using (VC)m to denote VC repeated m times, this may again be written as

[C] (VC)m [V]

m will be called the measure of any word or word part when represented in this form. The case m = 0 covers the null word. Here are some examples :

* m=0 OTR, EE, TREE, Y, BY.
* m=1 TROUBLE, OATS, TREES, IVY.
* m=2 TROUBLES, PRIVATE, OATEN, ORRERY.

**Rule for removing a suffix**

* The rules are of the form: (condition) S1 -> S2 Where S1 and S2 are suffixes.
* This means that if a word ends with a suffix S1, and the stem before S1 satisfies the given condition then S1 is replaced by S2.

eg (m>1) EMENT ->

* Here S1 is 'EMENT' and S2 is null. This would map REPLACEMENT to REPLAC, since REPLAC is a word part for which m = 2.

The 'condition' part may also contain the following :

|  |  |
| --- | --- |
| m | The measure of the stem |
| \*S | The stem ends with S |
| \*v\* | The stem contains a vowel |
| \*d | The stem ends with a double consonant (TT, SS) |
| \*o | The stem ends in CVC (second C not W, X, or Y)  E.g. WIL, HOP |

And the condition part may also contain expressions with and, or and not, so that :

(m>1 and (\*S or \*T)) tests for a stem with m>1 ending in S or T, while (\*d and not (\*L or \*S or \*Z)) tests for a stem ending with a double consonant other than L, S or Z. Elaborate conditions like this are required only rarely.

**Step 1a :**

* SSES -> SS (Example : caresses -> caress)
* IES -> I (Example : ponies -> poni ; ties -> ti)
* SS -> SS (Example : caress -> caress)
* S -> € (Example : cats -> cat)
* SSES -> SS: This rule reduces words ending in "sses" to just "ss". It's primarily aimed at plural nouns. Example: Input: caresses, Output: caress
* IES -> I: This rule changes words ending in "ies" to end in just "i". It's also primarily focused on plural nouns. Example: Input: ponies, Output: poni
  + SS -> SS: This part is more of a placeholder in the algorithm. It doesn't change the word but serves as a reference point.
  + S -> (remove the "s" suffix): This part removes the trailing "s" from words, primarily aimed at singular and plural nouns. Example: Input: cats, Output: cat

Rule 1a helps normalize words by reducing them to their base forms, especially in cases where pluralization or verb conjugation adds extra letters that aren't part of the word's core meaning.

**Step 1b:**

* + (m>1) EED -> EE

Condition verified: agreed -> agree

Condition not verified: needed -> need

* + (\*V\*) ED -> €

Condition verified: plastered -> plaster

Condition not verified: bled -> bled

* + (\*V\*) ING -> €

Condition verified: motoring -> motor

Condition not verified: sing -> sing

Rule 1b of the Porter Stemmer algorithm addresses additional suffixes beyond those covered in Rule 1a. Here's an explanation of each part of Rule 1b with examples:

* 1. **EED or ED -> EE**: This rule checks for words ending in "eed" or "ed" and replaces them with "ee" or simply removes the "ed" suffix. It primarily targets past tense verbs.

Input: needed, Output: need

* 1. **ING ->** (remove the "ing" suffix): 2. This part removes the "-ing" suffix from words, typically indicating a present participle or gerund.

Example: Input: playing, Output: play

* 1. **ATIONAL -> ATE:** This rule converts words ending in "ational" to end in "ate". It's focused on transforming adjectives or nouns into verbs.

Example: Input: relational, Output: relate

* 1. **TIONAL -> TION:** Similar to the previous rule, this one converts words ending in "tional" to end in "tion".

Example: Input: conditional, Output: condition

These examples illustrate how Rule 1b of the Porter Stemmer algorithm applies various transformations to reduce words to their base forms by removing or replacing specific suffixes.

If the second or third of the rules in Step 1b is successful, the following is done:

* + AT-> ATE (Example : conflat(ed) -> conflate)
  + BL-> BLE (Example : troubl(ed) -> trouble)
  + IZ-> IZE (Example : siz(ed) -> size)
  + S-> (Example : cats-> cat)

(\*d and not (\*L or \*S or \*Z)) -> single letter

(Example : hopp(ing) -> hop ; tann(ed) -> tan ; fall(ing) -> fall ; hiss(ing) -> hiss ; fizz(ed) -> fizz)

(m=1 and \*o)-> E (Example : fail(ing)-> fail ; fil(ing)-> file)

The rule to map to a single letter causes the removal of one of the double letter pair. The -E is put back on -AT, -BL and -IZ, so that the suffixes -ATE, -BLE and -IZE can be recognised later. This E may be removed in step 4.

**Step 1c:**

(\*v\*) Y -> I (Example : happy -> happi ; sky -> sky)

* Step 1c of the Porter Stemmer algorithm aims to handle certain cases where the algorithm needs to address the removal of an "e" suffix from the end of a word.
* E -> (remove the "e" suffix if preceded by a short syllable):
* This rule checks if a word ends with an "e" and whether the preceding part of the word constitutes a "short syllable." A short syllable, in this context, generally refers to a syllable with a vowel followed by a consonant (VC). If these conditions are met, the final "e" is removed from the word.
* Example: Input: loveable

"lo" is considered a short syllable because it's followed by a consonant "v". Hence, the "e" at the end is removed.

Output: lovab

Step 1 deals with plurals and past participles. The subsequent steps are much more straightforward.

**Step 2**: Derivational Morphology, I

* (m>0) ATIONAL -> ATE, Relational -> relate
* (m>0) IZATION -> IZE, generalization-> generalize
* (m>0) BILITI -> BLE, sensibiliti -> sensible

**Step 3:** Derivational Morphology, II

* (m>0) ICATE -> IC, triplicate -> triplic
* (m>0) FUL -> €, hopeful -> hope
* (m>0) NESS -> E, goodness -> good

**Step 4:** Derivational Morphology, III

* (m>0) ANCE -> €, allowance-> allow
* (m>0) ENT -> €, dependent-> depend

**Step 5a**

* (m>1) E -> €, probate -> probat
* (m=1 & !\* o) NESS -> €, goodness -> good

**Step 5b**

* (m>1 & \*d & "L) -> single letter
* Condition verified: controll -> control
* Condition not verified: roll -> roll

A diagram of steps to a specific language

Description automatically generated

1. **What is Tokenization?**

* Tokenization is breaking a text chunk in smaller parts. Whether it is breaking Paragraph in sentences, sentence into words or word in characters.
* Tokenization substitute sensitive information with equivalent non sensitive information.
* The nonsensitive, replacement information is called a token.
* Tokens can be created in various ways:
  + Using a mathematically reversible cryptographic function with a key
  + Using a non reversible function such as a hash function.
  + Using an index function or randomly generated number.
* Tokenization is a fundamental task in natural language processing (NLP) where text is divided into smaller units called tokens.
* These tokens can be words, characters, or sub words depending on the granularity required for the task at hand.
* Tokenization is the first step in processing textual data for tasks such as sentiment analysis, named entity recognition, machine translation, and more.

**Types of Tokenization**

1. **Word Tokenization**: This type of tokenization breaks the text into words based on whitespace or punctuation.

* Input: "Tokenization is important in NLP."
* Output: ["Tokenization", "is", "important", "in", "NLP", "."]

1. **Sentence Tokenization**: This divides the text into sentences based on punctuation or other indicators of sentence boundaries.

* Input: "Tokenization is important in NLP. It helps in text processing."
* Output: ["Tokenization is important in NLP.", "It helps in text processing."]

1. **Character Tokenization:** This breaks the text into individual characters.

* Input: "Tokenization"
* Output: ["T", "o", "k", "e", "n", "i", "z", "a", "t", "i", "o", "n"]

1. **Subword Tokenization:** This divides the text into smaller units that may or may not correspond to complete words. It is often used in morphologically rich languages or when dealing with rare words.

* Input: "Tokenization"
* Output: ["To", "ken", "iz", "ation"]
* Stop words are words in the text that add no sense to the phrase and removing them will no influence the processing of text for the specified purpose. They are deleted from the lexicon to minimise noise and the feature set's dimension.
* There are numerous tokenization strategies accessible, depending on the language and goal of modelling. A few tokenization approaches used in NLP are listed below.
* **Whitespace Tokenization**: This simple approach splits the text based on whitespace characters (e.g., space, tab, newline). While straightforward, it may not handle all cases properly, especially in languages where words are not separated by spaces.

Example: **Original Text:** "Hello world"

* **Tokenization:**

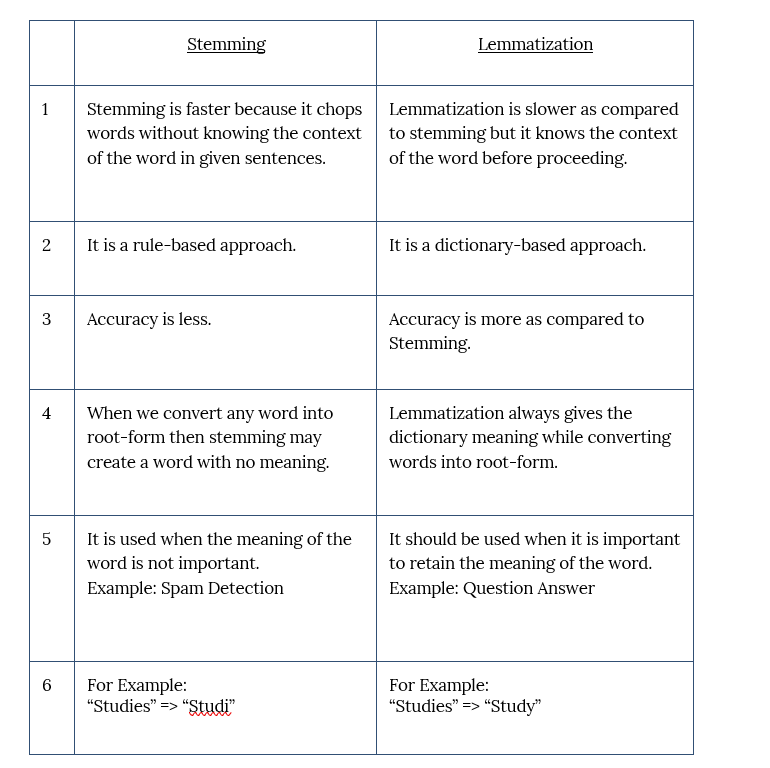
Token 1: "Hello"

Token 2: "world“

* **Regular Expression Tokenization**: Regular expressions (regex) can be used to define patterns for word boundaries. For example, \w+ matches one or more word characters (letters, digits, or underscores), effectively tokenizing based on word boundaries.
* Example Text: "This is a sample text”
* Tokenization:
* Token 1: "This"
* Token 2: "is"
* Token 3: "a"
* Token 4: "sample"
* Token 5: "text"
* **Spacy Tokenizer:** Spacy is another popular NLP library that provides a tokenization module. Its tokenizer splits the text into tokens based on rules specific to each language. It also provides additional linguistic annotations.
* **Byte Pair Encoding (BPE):** BPE is a subword tokenization algorithm that iteratively merges the most frequent pairs of consecutive symbols (characters or character sequences). It's particularly useful for handling rare words and morphologically rich languages.
* **WordPiece Tokenization**: Similar to BPE, WordPiece tokenization is a subword tokenization algorithm used by models like BERT. It iteratively selects the most frequent subword units from a fixed vocabulary.
* **SentencePiece**: SentencePiece is a subword tokenization library that provides several tokenization algorithms, including unigram language model, BPE, and WordPiece. It is widely used in various NLP tasks and supports multiple languages.
* **Custom Rule-based Tokenization**: Depending on the specific requirements of your task or the characteristics of the text data you're working with, you may develop custom tokenization rules to handle specific cases effectively.
* **Penn TreeBank**: It is a rule-based tokenization method that separates out clitics ( words that normally occur only in combination with another word, for example in *I’m*), keeps hyphenated words together, and separates out all punctuation.

**Need of Tokenization:**

* **Text normalization:** Tokenization helps to normalize text by breaking it down into smaller units, such as words or phrases, which can be analyzed more effectively.
* **Feature extraction:** Tokenization aids in the extraction of meaningful features from text data, which can be used as input for machine learning algorithms.
* **Handling of ambiguity:** NLP algorithms can better handle ambiguous text by breaking it down into smaller units and analyzing it more fine-grained.
* **Improved processing speed:** Tokenization can help to speed up NLP algorithms by reducing the size of the text data that needs to be processed.
* **Word representation**: Tokenization can help NLP algorithms to better represent words, by breaking them down into smaller units and preserving their structure and meaning.
* **Consistent representation:** Tokenization helps to ensure that different forms of the same word or phrase are treated consistently, which is important for many NLP tasks, such as sentiment analysis and language translation.



1. **Edit Distance**

* In Natural Language Processing (NLP), the edit distance is a measure of how different two strings of characters are from each other.
* It is the minimum number of single-character insertions, deletions, or substitutions needed to transform one string into the other.
* The edit distance is used in a variety of NLP applications, such as spellchecking, auto-correction, and machine translation.
* For example, in spell-checking, the edit distance is used to suggest corrections for misspelled words. In machine translation, the edit distance is used to align source and target language sentences and to identify the best translation.
* The algorithm for calculating edit distance involves building a matrix that represents the number of edit operations required to transform one string into another.
* The matrix is initialized with values that represent the cost of each operation (insertion, deletion, or substitution), and then filled in using dynamic programming.
* The edit distance can be calculated efficiently using dynamic programming algorithms such as the Wagner-Fisher algorithm or the Hirschberg algorithm.
* These algorithms have a time complexity of O(mn), where m and n are the lengths of the two strings being compared.
* Let's say we have two strings:

A black and grey background

Description automatically generated with medium confidence

* To find the edit distance between these two strings, we can use a dynamic programming algorithm that builds a matrix of edit distances between substrings of the two strings.
* The matrix has dimensions (m+1) x (n+1), where m and n are the lengths of the two strings.
* We can initialize the matrix by setting the first row and column to be the edit distances between each string and the empty string. For example:

A black screen with a black border

Description automatically generated

The edit distance between an empty string and any non-empty string is simply the length of the non-empty string, so the first row and column are just counting the number of characters in each string.

This relation says that if the characters at positions i-1 and j-1 in the two strings are the same, then the edit distance between those substrings is the same as the edit distance.

A screen shot of a computer

Description automatically generated

A black screen with a black border

Description automatically generated

1. **With respect to Porter Stemmer, evaluate the measure of the following words: { queue, cryogenic, onomatopoeia, neuropsychoticdisorder, larynx }**

* queue: The stemmer will likely reduce this to "queu."
* cryogenic: The stemmer may reduce this to "cryogen."
* onomatopoeia: This word is often left unchanged by stemming algorithms because it's already quite close to its root form.
* neuropsychoticdisorder: Since this is a compound word, the stemmer may not handle it effectively and may not return a meaningful stem.
* larynx: The stemmer may reduce this to "laryn."

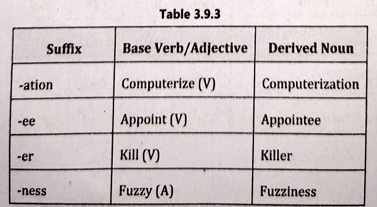
1. **Explain derivational and inflectional morphology**

* Morphology is study of internal structure of words. Given a particular word in a language, what are the different meaningful units it is made up of and each small unit is called a morpheme. It involves tasks such as lemmatization (reducing words to their base or root form) and stemming (stripping suffixes from words).
* cat: stem cats: cat + s unhappy happily unhappily
* cats: N +PL cat: N+SG sort: V+SG sorts: V+PL sort: N+PL
* Example: Breaking down "running" into "run" (root) and "ing" (suffix).
* Computational tool to perform morphological parsing is finite state transducer.

There are 3 types of word formation.

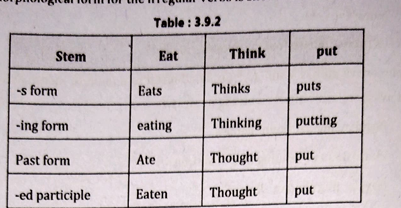
1. **Derivational Morphology:**

* Derivational morphology is concerned with the creation of new words by morpheme or adding affixes (prefixes, suffixes, infixes, or circumfixes) to base words or roots(stem).
* These affixes alter the meaning or grammatical category of the base word. For example, in English, adding the suffix "-ness" to the adjective "happy" creates the noun "happiness," changing the word's grammatical category from an adjective to a noun while also altering its meaning.
* Derivational morphology often results in a change in the lexical category (e.g., noun to verb, verb to adjective) or semantic meaning of the base word.



1. **Inflectional Morphology:**

* Inflectional morphology involves the modification of words to express grammatical relationships such as tense, aspect, mood, number, case, gender, or person.
* Inflectional morphemes are bound morphemes (affixes) that attach to the base word (stem) without substantially changing its meaning or lexical category.
* These morphemes typically indicate grammatical features such as plurality, possession, tense, or comparison. For example, in English, the suffix "-s" added to the noun "dog" indicates plurality, changing it to "dogs."
* Inflectional morphology is more about grammatical relationships within a sentence rather than creating entirely new words.



1. **Compounding**

* A word is composed of a number of morphemes concatenated together.
* In English, compounds can be written together, for example, notebook, desktop, overlook, bookstore, fireman etc.
* Separately the compounds are written as Living room, dinner table, full moon etc.

1. **What is the role of FSA in morphological analysis?**

Finite State Automata (FSA) play a significant role in morphological analysis, particularly in computational linguistics and natural language processing. Here's how FSA contributes to morphological analysis:

1. **Lexical Analysis**: FSA can be used to perform lexical analysis, which involves breaking down words into smaller units, such as morphemes or syllables. By constructing an FSA that recognizes valid morphological patterns and transitions between them, linguists and computer scientists can effectively analyze the structure of words in a language.
2. **Tokenization**: FSA can aid in tokenization, the process of breaking a stream of text into words or tokens. By designing an FSA that recognizes valid word boundaries based on linguistic rules and patterns, tokenization becomes more efficient and accurate.
3. **Morphological Parsing**: FSA is commonly used for morphological parsing, which involves analyzing the grammatical structure and properties of words. By defining states and transitions that represent different morphological features (e.g., tense, number, gender), an FSA can identify and classify the morphological properties of words in a sentence.
4. **Stemming and Lemmatization**: FSA is utilized in stemming and lemmatization algorithms, which aim to reduce words to their base or root forms. By designing an FSA that models the affixation and morphological rules of a language, stemming and lemmatization can be performed efficiently and accurately.
5. **Spell Checking and Correction**: FSA is employed in spell checking and correction systems to identify and correct misspelled words. By constructing an FSA that represents valid word forms in a language, spelling errors can be detected by comparing input text against the accepted word forms defined by the FSA.
6. **Identify the morphological type (Noun phrase, Verb Phrase, Adjective Phrase) of following sentence segments**
7. important to Bill
8. looked up the tree

**"important to Bill"**

* This segment consists of the adjective "important" and the prepositional phrase "to Bill."
* "important" is an adjective modifying the noun "Bill."
* "to Bill" is a prepositional phrase acting as an adjective phrase modifying "important."
* Therefore, the morphological type of this segment is an Adjective Phrase.

**"looked up the tree"**

* This segment consists of the verb "looked" and the prepositional phrase "up the tree."
* "looked" is the main verb of the sentence.
* "up the tree" is a prepositional phrase acting as an adverbial phrase modifying "looked."
* Therefore, the morphological type of this segment is a Verb Phrase.

**Lemmatization**

* Lemmatization is performing the task properly by using the vocabulary and morphological analysis of words, and it generally aims to eliminate inflectional endings and to return the base form of the word/ dictionary form of a word, that is called lemma.
* If we try the word saw, stemming might return just s, but lemmatization would try to return either see or saw depending on whether the use of the token was as a verb or a noun.
* In other words, Lemmatization is a technique responsible for grouping diverse inflected forms of words into the root form, having the similar meaning. It is similar to stemming in order, it provides the stripped word that has some dictionary meaning. The Morphological analysis would need the mining of the accurate lemma of each word.
* For example :

troubled' -> Lemmatization ->'troubled', and error

* The applications of lemmatizations are information retrieval, sentiment analysis, and document clustering.

**Discuss the various steps involved in NLP? Different stages involved in NLP process with suitable example?**

1. **Lexical Analysis**

* Lexical Analysis is the first stage in NLP. It is also known as morphological analysis.
* At this stage the structure of the words is identified and analysed.
* Lexicon of a language means the collection of words and phrases in a language.
* Lexical analysis is dividing the whole portion of text into paragraphs, sentences, and words.

1. **Syntactic Analysis (Parsing)**

* It involves analysis of words in the sentence for grammar and ordering words in a way that shows the relationship among the words.
* The sentence such as The school goes to girl is rejected by English syntactic analyser.

1. **Semantic Analysis**

* Semantic analysis draws the exact meaning or the dictionary meaning from the text.
* The text is checked for meaningfulness. It is done by mapping syntactic structures and objects in the task domain.
* The semantic analyser neglects sentence such as "hot ice-cream".

1. **Discourse Integration**

* The meaning of any sentence depends upon the meaning of the sentence just before it. Furthermore, it also brings about the meaning of immediately following sentence.
* For example: Meena is a girl, she goes to school here "she" is a dependency pointing to Meena.

1. **Pragmatic Analysis**

* During this, what was said is re-interpreted on what it truly meant. It contains deriving those aspects of language which necessitate real world knowledge.
* For example, John saw Mary in a garden with a cat, here we can't say that John is with cat or Mary is with cat.

**Module 2**

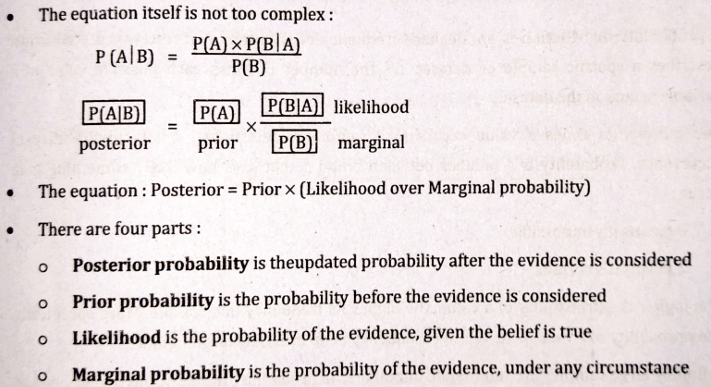
1. **Write short notes Bayes theorem?**

* It depicts the likelihood of an occurrence based on past knowledge of factors that may be associated with that event. It may also be used as an example of conditional probability.
* Bayes theorem (also known as the Bayes Rule or Bayes Law) is used to determine the conditional probability of event A when event B has already occurred**.**
* The general statement of Bayes’ theorem is “**The conditional probability of an event A, given the occurrence of another event B, is equal to the product of the event of B, given A and the probability of A divided by the probability of event B.”** i.e.
* where,

P(A) and P(B) are the probabilities of events A and B

P(A|B) is the probability of event A when event B happens

P(B|A) is the probability of event B when A happens

****

1. **Define random variables with example?**

* A Random Variable Probability is a mathematical concept that assigns numerical values to outcomes of a [sample space](https://www.geeksforgeeks.org/sample-space-probability/).
* There are two types of Random Variables- Discrete and Continuous.
* A random variable is considered a discrete random variable when it takes specific, or distinct values within an interval. Conversely, if it takes a continuous range of values, then it is classified as a continuous random variable.

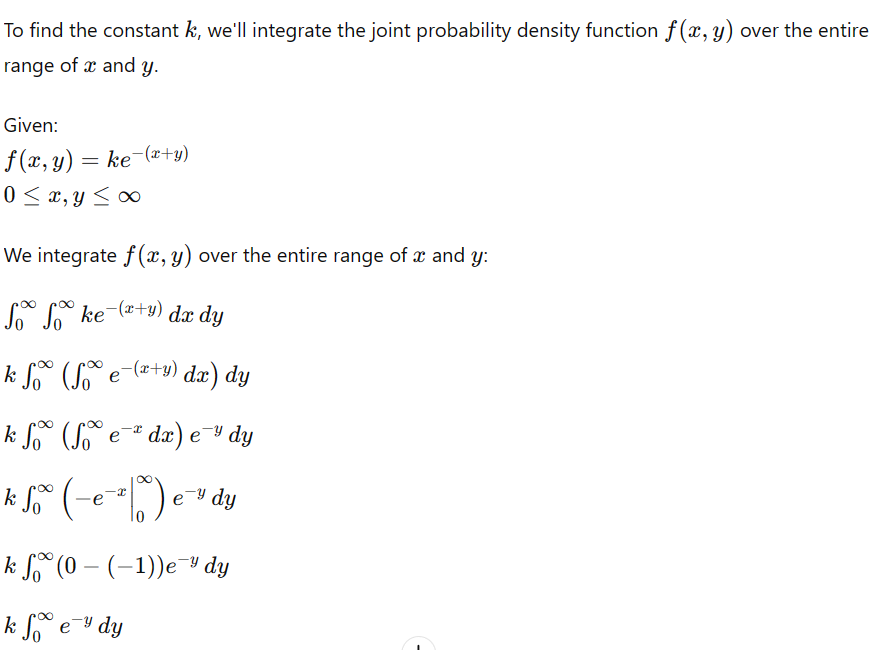
**Discrete Random Variable**

* A [Discrete Random Variable](https://www.geeksforgeeks.org/discrete-random-variable/) takes on a finite number of values. The probability function associated with it is said to be PMF(Probability Mass Function)
* If X is a discrete random variable and the PMF of X is P(xi), then
* 0 ≤ pi ≤ 1
* ∑pi = 1 where the sum is taken over all possible values of x

**Continuous Random Variable**

* Continuous Random Variable takes on an infinite number of values. The probability function associated with it is said to be [PDF (Probability Density Function)](https://www.geeksforgeeks.org/probability-density-function/).
* If X is a continuous random variable. P (x < X < x + dx) = f(x)dx then,
* 0 ≤ f(x) ≤ 1; for all x
* ∫ f(x) dx = 1 over all values of x
* **Discrete Random Variable - Coin Toss:**
* Imagine you're flipping a fair coin. Let 𝑋 be the random variable representing the outcome of the coin toss. The possible values of 𝑋 are either "Heads" or "Tails". This is a discrete random variable because there are only two distinct outcomes.
* **Continuous Random Variable - Temperature Measurement:**
* Now, picture measuring the temperature outside. Let 𝑌 be the random variable representing the temperature. The temperature can take on any value within a range, say from 0 to 100 degrees Celsius. This is a continuous random variable because temperature can vary continuously within this range, and there are infinitely many possible values between any two temperatures.

1. **f(x,y)=ke-(x+y)   0 ≤ x,y ≤ ∞. Find k  and marginal densities of x and y.**

****

**A white paper with black text

Description automatically generated**

**A math equations on a white background

Description automatically generated**

1. **Define mean and variance of a random variable?**

The term average of a random variable in probability and statistics is the mean or the expected value.

Variance is the expected value of the squared variation of a random variable from its mean value

**Discrete Random Variable:**

Mean (Expected Value): For a discrete random variable 𝑋, the mean, denoted as 𝜇 or 𝐸[𝑋], is calculated as the sum of each possible value of 𝑋 multiplied by its corresponding probability 𝑃(𝑋): 𝜇=𝐸[𝑋]=∑𝑖𝑥𝑖⋅𝑃(𝑋=𝑥𝑖) where 𝑥i​ are the possible values of 𝑋.

Variance: The variance of a discrete random variable 𝑋, denoted as 𝜎2 or 𝑉𝑎𝑟[𝑋], measures the spread of its values around the mean. It is calculated as the sum of the squared differences between each value of 𝑋 and the mean, weighted by their respective probabilities:

𝜎2 = 𝑉𝑎𝑟[𝑋] = ∑𝑖(𝑥𝑖−𝜇)2⋅𝑃(𝑋=𝑥𝑖)

**Continuous Random Variable:**

Mean (Expected Value): For a continuous random variable 𝑌, the mean, denoted as 𝜇 or 𝐸[𝑌], is calculated as the integral of the product of the variable and its probability density function 𝑓𝑌(𝑦) over all possible values:

𝜇=𝐸[𝑌]=

Variance: The variance of a continuous random variable 𝑌Y, denoted as 𝜎2σ2 or 𝑉𝑎𝑟[𝑌]Var[Y], is calculated as the integral of the squared differences between each value of 𝑌Y and the mean, weighted by their respective probabilities:

1. **Compute the conditional probability P(okay | okay) from the following input data**:

And it's okay okay to feel this way. Feel like you're lost. Dreaming and hoping. And it's okay okay to feel this way. But just know that it's all gonna be okay. And it's okay okay to feel this way. Feel like you're lost. Dreaming and hoping. And it's okay okay to feel this way. But just know that it's all gonna be okay.

To compute the conditional probability 𝑃(okay∣okay), we need to find the probability of the word "okay" appearing after the word "okay" in the given text.

Let's count the occurrences:

The word "okay" appears 10 times.

Out of these 10 occurrences, the word "okay" appears immediately after another "okay" 4 times.

So, 𝑃(okay∣okay)=4/10

Therefore, the conditional probability of seeing "okay" given that the previous word was "okay" in the provided text is approximately 4/10​, or about 0.4

1. **Identify derivational morphemes from the given data: With every day passing. She felt distant. Don't know if it was her anxiety, her thoughts, insecurities, these feelings suppressed her**

**Understanding Morphemes**

* Morphemes: The smallest units of meaning in a language.
* Derivational Morphemes: Morphemes that change the meaning or part of speech of a word when added (e.g., "happy" to "happiness").

**Given Data**

* With every day passing.
* She felt distant.
* Don't know if it was her anxiety, her thoughts, insecurities,
* These feelings suppressed her.

**Steps to Identify Derivational Morphemes**

* Break down each word into its base form and any added morphemes.
* Identify if any added morphemes change the meaning or part of speech of the base word.

**Analysis**

1. **With every day passing**.

* "passing" (base: "pass", suffix: "-ing" which is an inflectional morpheme, not derivational)

1. **She felt distant.**

* "distant" (base: "distance", suffix: "-ant" which is a derivational morpheme changing the noun "distance" to an adjective "distant")

1. **Don't know if it was her anxiety, her thoughts, insecurities,**

* "anxiety" (base: "anxious", suffix: "-ity" which is a derivational morpheme changing the adjective "anxious" to the noun "anxiety")
* "thoughts" (base: "thought", suffix: "-s" which is an inflectional morpheme, not derivational)
* "insecurities" (base: "insecure", suffix: "-ity" and prefix: "in-" both derivational morphemes changing the adjective "secure" to the noun "insecurity", and then pluralized with the inflectional morpheme "-s")

1. **These feelings suppressed her.**

* "feelings" (base: "feel", suffix: "-ing" which is an inflectional morpheme, not derivational, and then pluralized with the inflectional morpheme "-s")
* "suppressed" (base: "suppress", suffix: "-ed" which is an inflectional morpheme, not derivational)

**Derivational Morphemes Identified**

* distant: "distance" + "-ant"
* anxiety: "anxious" + "-ity"
* insecurities: "secure" + "in-" (prefix) + "-ity" (suffix)

1. **Compute the cosine distance between the two given documents:**

D1 = For want of a nail, the shoe was lost.

D2 = For want of a shoe, the horse was lost.

**Step-by-Step Tabular Form**

**1. Tokenize the Documents**

* D1: "For want of a nail, the shoe was lost."
* Tokens: ["for", "want", "of", "a", "nail", "the", "shoe", "was", "lost"]
* D2: "For want of a shoe, the horse was lost."
* Tokens: ["for", "want", "of", "a", "shoe", "the", "horse", "was", "lost"]

**2. Build the Vocabulary**

Vocabulary: ["for", "want", "of", "a", "nail", "the", "shoe", "was", "lost", "horse"]

**3. Vectorize the Documents**

|  |  |  |
| --- | --- | --- |
| Word | D1 Frequency | D2 Frequency |
| for | 1 | 1 |
| want | 1 | 1 |
| of | 1 | 1 |
| a | 1 | 1 |
| nail | 1 | 0 |
| the | 1 | 1 |
| shoe | 1 | 1 |
| was | 1 | 1 |
| lost | 1 | 1 |
| horse | 0 | 1 |

**Document Vectors**

* D1 Vector: [1, 1, 1, 1, 1, 1, 1, 1, 1, 0]
* D2 Vector: [1, 1, 1, 1, 0, 1, 1, 1, 1, 1]

**Cosine Similarity and Distance Calculation**

**4. Compute the Cosine Similarity**

Cosine similarity formula: cosine similarity=

**Dot Product:** 𝐴⋅𝐵=1×1+1×1+1×1+1×1+1×0+1×1+1×1+1×1+1×1+0×1=8

**Magnitudes:**

* ∥𝐴∥=
* ∥𝐵∥=

Cosine Similarity: cosine similarity== ≈ 0.888

**5. Calculate the Cosine Distance**

cosine distance=1−cosine similarity=1−0.888≈0.111