NATURAL LANGUAGE PROCESSSING

NLP focuses on the interaction between computers and human language, enabling machines to understand, interpret, and generate human language in a way that is both meaningful and useful.

NLP Techniques

- NLP encompasses a wide array of techniques that aimed at enabling computers to process and understand human language.
- These tasks can be categorized into several broad areas, each addressing different aspects of language processing.

Here are some of the key NLP techniques:

1. **Text Processing and Preprocessing In NLP**
    - **Tokenization**: Dividing text into smaller units, such as words or sentences.
    - **Stemming and Lemmatization**: Reducing words to their base or root forms.
    - **Stopword Removal**: Removing common words (like “and”, “the”, “is”) that may not carry significant meaning.
    - **Text Normalization**: Standardizing text, including case normalization, removing punctuation, and correcting spelling errors.

2. Syntax and Parsing In NLP

- **Part-of-Speech (POS) Tagging**: Assigning parts of speech to each word in a sentence (e.g., noun, verb, adjective).
- **Dependency Parsing**: Analyzing the grammatical structure of a sentence to identify relationships between words.
- **Constituency Parsing**: Breaking down a sentence into its constituent parts or phrases (e.g., noun phrases, verb phrases).

3. Semantic Analysis

- **Named Entity Recognition (NER)**: Identifying and classifying entities in text, such as names of people, organizations, locations, dates, etc.

- **Word Sense Disambiguation (WSD)**: Determining which meaning of a word is used in a given context.
- **Coreference Resolution**: Identifying when different words refer to the same entity in a text (e.g., “he” refers to “John”).

4. Information Extraction

- **Entity Extraction**: Identifying specific entities and their relationships within the text.
- **Relation Extraction**: Identifying and categorizing the relationships between entities in a text.

5. Text Classification in NLP

- **Sentiment Analysis**: Determining the sentiment or emotional tone expressed in a text (e.g., positive, negative, neutral).
- **Topic Modeling**: Identifying topics or themes within a large collection of documents.
- **Spam Detection**: Classifying text as spam or not spam.

6. Language Generation

- **Machine Translation**: Translating text from one language to another.
- **Text Summarization**: Producing a concise summary of a larger text.
- **Text Generation**: Automatically generating coherent and contextually relevant text.

7. Speech Processing

- **Speech Recognition**: Converting spoken language into text.
- **Text-to-Speech (TTS) Synthesis**: Converting written text into spoken language.

8. Question Answering

- **Retrieval-Based QA**: Finding and returning the most relevant text passage in response to a query.
- **Generative QA**: Generating an answer based on the information available in a text corpus.

9. **Dialogue Systems**
    - **Chatbots and Virtual Assistants**: Enabling systems to engage in conversations with users, providing responses and performing tasks based on user input.

10. **Sentiment and Emotion Analysis in NLP**
    - **Emotion Detection**: Identifying and categorizing emotions expressed in text.
    - **Opinion Mining**: Analyzing opinions or reviews to understand public sentiment toward products, services, or topics.

Working of Natural Language Processing (NLP)

Working in natural language processing (NLP) typically involves using computational techniques to analyze and understand human language. This can include tasks such as language understanding, language generation, and language interaction.

1. Text Input and Data Collection
- **Data Collection**: Gathering text data from various sources such as websites, books, social media, or proprietary databases.
- **Data Storage**: Storing the collected text data in a structured format, such as a database or a collection of documents.

2. Text Preprocessing

Preprocessing is crucial to clean and prepare the raw text data for analysis. Common preprocessing steps include:
- **Tokenization**: Splitting text into smaller units like words or sentences.
- **Lowercasing**: Converting all text to lowercase to ensure uniformity.
- **Stopword Removal**: Removing common words that do not contribute significant meaning, such as “and,” “the,” “is.”
- **Punctuation Removal**: Removing punctuation marks.
- **Stemming and Lemmatization**: Reducing words to their base or root forms. Stemming cuts off suffixes, while lemmatization

considers the context and converts words to their meaningful base form.
- **Text Normalization**: Standardizing text format, including correcting spelling errors, expanding contractions, and handling special characters.

3. **Text Representation**

- **Bag of Words (BoW)**: Representing text as a collection of words, ignoring grammar and word order but keeping track of word frequency.
- **Term Frequency-Inverse Document Frequency (TF-IDF)**: A statistic that reflects the importance of a word in a document relative to a collection of documents.
- **Word Embeddings**: Using dense vector representations of words where semantically similar words are closer together in the vector space (e.g., Word2Vec, GloVe).

4. Feature Extraction

Extracting meaningful features from the text data that can be used for various NLP tasks.

- **N-grams**: Capturing sequences of N words to preserve some context and word order.
- **Syntactic Features**: Using parts of speech tags, syntactic dependencies, and parse trees.
- **Semantic Features**: Leveraging word embeddings and other representations to capture word meaning and context.

5. Model Selection and Training

Selecting and training a machine learning or deep learning model to perform specific NLP tasks.

- **Supervised Learning**: Using labelled data to train models like Support Vector Machines (SVM), Random Forests, or deep learning models like Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs).

- **Unsupervised Learning**: Applying techniques like clustering or topic modelling (e.g., Latent Dirichlet Allocation) on unlabelled data.
- **Pre-trained Models**: Utilizing pre-trained language models such as BERT, GPT, or transformer-based models that have been trained on large corpora.

6. Model Deployment and Inference

Deploying the trained model and using it to make predictions or extract insights from new text data.

- **Text Classification**: Categorizing text into predefined classes (e.g., spam detection, sentiment analysis).
- **Named Entity Recognition (NER)**: Identifying and classifying entities in the text.
- **Machine Translation**: Translating text from one language to another.
- **Question Answering**: Providing answers to questions based on the context provided by text data.

7. Evaluation and Optimization

Evaluating the performance of the NLP algorithm using metrics such as accuracy, precision, recall, F1-score, and others.

- **Hyperparameter Tuning**: Adjusting model parameters to improve performance.
- **Error Analysis**: Analyzing errors to understand model weaknesses and improve robustness.

8. Iteration and Improvement

Continuously improving the algorithm by incorporating new data, refining preprocessing techniques, experimenting with different models, and optimizing features.

Technologies related to Natural Language Processing

There are a variety of technologies related to natural language processing (NLP) that are used to analyze and understand human language. Some of the most common include:

1. **Machine learning:** NLP relies heavily on machine learning techniques such as supervised and unsupervised learning, deep learning, and reinforcement learning to train models to understand and generate human language.

2. **Natural Language Toolkits (NLTK)** and other libraries: NLTK is a popular open-source library in Python that provides tools for NLP tasks such as tokenization, stemming, and part-of-speech tagging. Other popular libraries include spaCy, OpenNLP, and CoreNLP.

3. **Parsers:** Parsers are used to analyze the syntactic structure of sentences, such as dependency parsing and constituency parsing.

4. **Text-to-Speech (TTS) and Speech-to-Text (STT) systems:** TTS systems convert written text into spoken words, while STT systems convert spoken words into written text.

5. **Named Entity Recognition (NER) systems:** NER systems identify and extract named entities such as people, places, and organizations from the text.

6. **Sentiment Analysis:** A technique to understand the emotions or opinions expressed in a piece of text, by using various techniques like Lexicon-Based, Machine Learning-Based, and Deep Learning-based methods

7. **Machine Translation:** NLP is used for language translation from one language to another through a computer.

8. **Chatbots:** NLP is used for chatbots that communicate with other chatbots or humans through auditory or textual methods.

9. **AI Software:** NLP is used in question-answering software for knowledge representation, analytical reasoning as well as information retrieval.

Applications of Natural Language Processing (NLP):

- **Spam Filters:** One of the most irritating things about email is spam. Gmail uses natural language processing (NLP) to discern which emails are legitimate and which are spam. These spam filters look at the text in all the emails you receive and try to figure out what it means to see if it's spam or not.

- **Algorithmic Trading:** Algorithmic trading is used for predicting stock market conditions. Using NLP, this technology examines news headlines about companies and stocks and attempts to comprehend their meaning in order to determine if you should buy, sell, or hold certain stocks.

- **Questions Answering:** NLP can be seen in action by using Google Search or Siri Services. A major use of NLP is to make search engines understand the meaning of what we are asking and generate natural language in return to give us the answers.

- **Summarizing Information:** On the internet, there is a lot of information, and a lot of it comes in the form of long documents or articles. NLP is used to decipher the meaning of the data and then provides shorter summaries of the data so that humans can comprehend it more quickly.

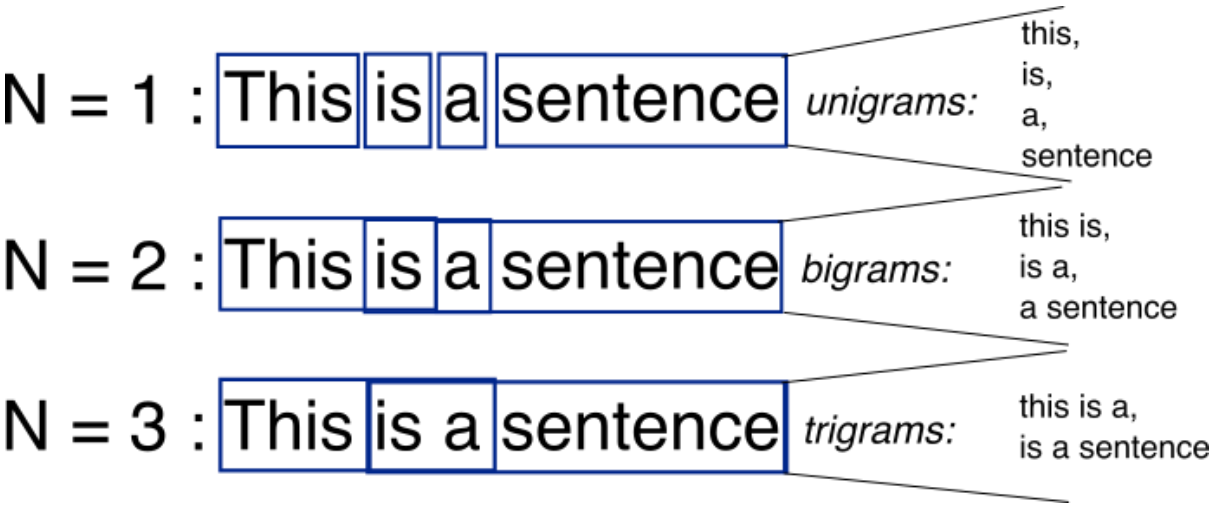
SUBSECTION OF NLP

Aspect	NLP	NLU	NLG
Definition	Processing of human language by a computer program	Subset of NLP focused on deeper understanding of language	Production of human-like language by a computer program
Tasks	Text classification, named entity recognition, sentiment analysis, etc.	Semantic analysis, intent recognition, dialogue management, etc.	Summarization, translation, content generation, etc.
Goal	Analysing and processing language	Understanding the meaning and context of language	Generating language
Input	Text, speech, and other language data	Text, speech, and other language data	Data and parameters to generate text
Output	Processed language data	Understanding of the meaning and context behind language	Human-like language
Applications	Sentiment analysis, language translation, speech recognition, etc.	Chatbots, virtual assistants, voice assistants, etc.	Automated content generation, data-to-text generation, etc.

EDA IN NLP

1. N-gram

N-gram can be defined as the continuous sequence of n items from a given sample of text or speech. The items can be letters, words, or base pairs according to the application. The N-grams typically are collected from a text or speech corpus (A long text dataset).



Word Cloud

-

Word Cloud is a data visualization technique used for representing text data in which the size of each word indicates its frequency or importance.

Advantages of Word Clouds :

1. Analyzing customer and employee feedback.
2. Identifying new SEO keywords to target.

Drawbacks of Word Clouds :

1. Word Clouds are not perfect for every situation.
2. Data should be optimized for context.

Keyword Extraction

Keyphrase or keyword extraction in NLP is a text analysis technique that extracts important words and phrases from the input text. These key phrases can be used in a variety of tasks, including information retrieval, document summarization, and content categorization. This task is performed in two stages:

Candidate Generation: This process involves the identification of all possible keywords from the input text.

Keyphrase Ranking: After the candidate keywords are generated, they are ranked in order of importance for the identification of the best keywords.

RAKE

RAKE stands for Rapid Automatic Keyword Extraction and it is a frequency-based key phrase extractor

- RAKE is a domain-independent algorithm for extracting keywords from individual documents. It operates by looking for frequent word sequences and co-occurrences in the text.
- It divides the text into phrases by identifying stop words and punctuation. Then, it calculates a score for each phrase based on the frequency of words and their degree of association with other words in the document.

2. **Key Features**:

   - **Stop Words**: Uses a list of stop words to break the text into meaningful phrases.
   - **Co-occurrence**: Considers how often words appear together.
   - **Simple and Fast**: Known for its simplicity and speed, making it suitable for real-time applications.

3. **Scoring**:

   - Each word in a phrase is scored based on its frequency and its degree (the number of words it co-occurs with). The final score for a phrase is the sum of its constituent word scores.

YAKE

YAKE stands for Yet Another Keyword Extractor and it is an unsupervised approach for automatic keyword extraction by leveraging text features.

RAKE (Rapid Automatic Keyword Extraction) and YAKE (Yet Another Keyword Extractor) are both algorithms designed for automatic keyword extraction from text. They are used to identify important terms and phrases that capture the essence of a document. Here are the key differences and characteristics of each:

YAKE (Yet Another Keyword Extractor)

1. **Methodology**:
    - YAKE is an unsupervised, domain-independent, and single-document keyword extraction algorithm. It focuses on the statistical properties of text and the relationship between words within the text.
    - It calculates five features for each candidate keyword: casing, word position, word frequency, word relatedness to context, and word differentiation.
2. **Key Features**:
    - **Local Context Analysis**: Emphasizes local context and word position in the document.
    - **Statistical Properties**: Analyzes various statistical properties of words to determine their importance.
    - **Flexibility**: Can adapt to different languages and types of text without the need for extensive pre-processing.
3. **Scoring**:
    - Combines the five features into a single score to rank keywords. It penalizes common words and words appearing in predictable patterns (e.g., titles, headers).

Comparative Summary

- **Complexity**: RAKE is simpler and faster, making it suitable for quick and straightforward keyword extraction tasks. YAKE, on the other hand, is more sophisticated and considers a broader range of features, potentially leading to more accurate keyword extraction in complex documents.
- **Customizability**: YAKE offers more parameters to tweak and can be adapted more easily to different languages and text types.
- **Performance**: RAKE performs well with minimal computational resources and is effective for real-time applications. YAKE may provide better results in terms of keyword relevance and precision, especially in varied textual contexts.

Use Cases

- **RAKE**: Ideal for applications where quick keyword extraction is needed without much customization, such as preliminary text analysis, simple document indexing, or as part of a larg pipeline where speed is crucial.
- **YAKE**: Better suited for scenarios requiring more precise keyword extraction, such as detailed text analytics, content summarization, and applications dealing with heterogeneous text corpora.

Both RAKE and YAKE have their strengths and are chosen based on the specific needs and constraints of the task at hand.

PREPROCESSING

Tokenization in **natural language processing (NLP)** is a technique that involves dividing a sentence or phrase into smaller units known as tokens.

Types of Tokenization

Tokenization can be classified into several types based on how the text is segmented. Here are some types of tokenization:

Word Tokenization:

Word tokenization divides the text into individual words. of meaning.

```
Input: "Tokenization is an important NLP task."
Output:["Tokenization", "is", "an", "important", "NLP", "task","."]
```

Sentence Tokenization:

The text is segmented into sentences during sentence tokenization. This is useful for tasks requiring individual sentence analysis or processing.

```
Input: "Tokenization is an important NLP task. It helps break down text into smaller units."

Output: ["Tokenization is an important NLP task.", "It helps break down text into smaller units."]
```

Normalization

When we normalize text, we attempt to reduce its randomness, bringing it closer to a predefined “standard”. This helps us to reduce the amount of different information that the computer has to deal

with, and therefore improves efficiency. The goal of normalization techniques like **stemming** and **lemmatization** is to reduce inflectional forms and sometimes derivationally related forms of a word to a common base form.

Key Techniques of Normalization

1. **Lowercasing**:
    - Converts all characters in the text to lowercase. This helps in reducing the dimensionality of the text and treating words like "Apple" and "apple" as the same.
    - Example: "Apple" -> "apple"
2. **Removing Punctuation**:
    - Eliminates punctuation marks from the text. This can help in simplifying the text and focusing on the actual words.
    - Example: "Hello, world!" -> "Hello world"
3. **Removing Stop Words**:
    - Stop words are common words (e.g., "and," "the," "is") that are often removed from the text as they usually do not contribute significant meaning.
    - Example: "This is a sample sentence." -> "sample sentence"
4. **Stemming**:
    - Reduces words to their base or root form by removing suffixes. This helps in treating different forms of a word as a single term.
    - Example: "running," "runner," "ran" -> "run"
    - Common algorithms: Porter Stemmer, Snowball Stemmer
5. **Lemmatization**:
    - Similar to stemming, but more sophisticated. It reduces words to their base or dictionary form (lemma) using morphological analysis.
    - Example: "running" -> "run," "better" -> "good"

- Lemmatization often requires a more comprehensive understanding of the word's context compared to stemming.

6. **Handling Contractions**:
    - Expands contractions into their full forms to ensure consistency.
    - Example: "can't" -> "cannot," "I'm" -> "I am"
7. **Removing Special Characters**:
    - Removes characters that are not part of the language being processed, such as emojis, symbols, and non-alphanumeric characters.
    - Example: "Hello @world!" -> "Hello world"
8. **Whitespace Normalization**:
    - Trims extra spaces and standardizes spacing between words.
    - Example: "Hello world" -> "Hello world"
9. **Accent and Diacritic Removal**:
    - Converts characters with accents or diacritics to their simple forms.
    - Example: "café" -> "cafe," "naïve" -> "naive"
10. **Canonicalization**:
    - Converts different forms or synonyms of a word to a single, standardized form.
    - Example: "USA," "U.S.A," "United States" -> "USA"

Importance of Normalization

1. **Consistency**: Ensures that variations of a word or phrase are consistently represented in a standard form, improving the reliability of text processing.
2. **Reduction of Dimensionality**: Reduces the vocabulary size, making it easier to analyze and model the text.
3. **Improved Accuracy**: Enhances the accuracy of various NLP tasks, such as text classification, information retrieval, and machine translation, by treating semantically similar terms as equivalent.
4. **Simplified Processing**: Simplifies downstream tasks, such as tokenization and vectorization, by standardizing the text format.

Application Areas

- **Text Preprocessing**: Essential in preparing text data for machine learning models and various NLP applications.
- **Search Engines**: Helps in improving search results by matching different variations of search queries.
- **Sentiment Analysis**: Enhances the analysis by treating different forms of words expressing similar sentiments equivalently.

- **Chatbots and Conversational Agents**: Ensures that user inputs are consistently interpreted.

Normalization is a fundamental step in NLP that significantly impacts the performance and accuracy of text processing and analysis tasks.

PRUNING

Stemming	Lemmatization
Stemming is a process that stems or removes last few characters from a word, often leading to incorrect meanings and spelling.	**Lemmatization** considers the context and converts the word to its meaningful base form, which is called Lemma.
For instance, stemming the word '**Caring**' would return '**Car**'.	For instance, lemmatizing the word '**Caring**' would return '**Care**'.
Stemming is used in case of large dataset where performance is an issue.	Lemmatization is computationally expensive since it involves look-up tables and what not.

Word Embeddings

Word Embeddings are numeric representations of words in a lower-dimensional space, capturing semantic and syntactic information.

Need for Word Embedding?

- To reduce dimensionality
- To use a word to predict the words around it.
- Inter-word semantics must be captured.

How are Word Embeddings used?

- They are used as input to machine learning models.
  Take the words —-> Give their numeric representation —-> Use in training or inference.
- To represent or visualize any underlying patterns of usage in the corpus that was used to train them.

FREQUENCY BASED WORD EMBEDDING

- Frequency-based Word Embedding Technique in NLP.
- Frequency-based embeddings are representations of words in a corpus based on their frequency of occurrence and relationships with other words.

CountVectorizer

- It is used to transform a given text into a vector on the basis of the frequency (count) of each word that occurs in the entire text.
- CountVectorizer creates a matrix in which each unique word is represented by a column of the matrix, and each text sample from the document is a row in the matrix.
- The value of each cell is nothing but the count of the word in that particular text sample

TF-IDF

- **TF-IDF** stands for Term Frequency Inverse Document Frequency of records. It can be defined as the calculation of how relevant a word in a series or corpus is to a text.
- The meaning increases proportionally to the number of times in the text a word appears but is compensated by the word frequency in the corpus (data-set).

Terminologies:

- **Term Frequency:** In document d, the frequency represents the number of instances of a given word t. Therefore, we can see that it becomes more relevant when a word appears in the text, which is rational. Since the ordering of terms is not significant,

we can use a vector to describe the text in the bag of term models. For each specific term in the paper, there is an entry with the value being the term frequency.

The weight of a term that occurs in a document is simply proportional to the term frequency.

```
tf(t,d) = count of t in d / number of words in d
```

- **Document Frequency:** This tests the meaning of the text, which is very similar to TF, in the whole corpus collection. The only difference is that in document d, TF is the frequency counter for a term t, while df is the number of occurrences in the document set N of the term t. In other words, the number of papers in which the word is present is DF.

```
df(t) = occurrence of t in documents
```

- **Inverse Document Frequency:** Mainly, it tests how relevant the word is. The key aim of the search is to locate the appropriate records that fit the demand. Since tf considers all terms equally significant, it is therefore not only possible to use the term frequencies to measure the weight of the term in the paper. First, find the document frequency of a term t by counting the number of documents containing the term:

```
df(t) = N(t)
where
df(t) = Document frequency of a term t
N(t) = Number of documents containing the term t
```

Term frequency is the number of instances of a term in a single document only; although the frequency of the document is the number of separate documents in which the term appears, it depends on the entire corpus. Now let's look at the definition of the frequency of the inverse paper. The IDF of the word is the number of documents in the corpus separated by the frequency of the text.

```
idf(t) = N/ df(t) = N/N(t)
```

The more common word is supposed to be considered less significant, but the element (most definite integers) seems too harsh. We then take the logarithm (with base 2) of the inverse frequency of the paper. So the if of the term t becomes:

```
idf(t) = log(N/ df(t))
```

- **Computation:** Tf-idf is one of the best metrics to determine how significant a term is to a text in a series or a corpus. tf-idf is a weighting system that assigns a weight to each word in a document based on its term frequency (tf) and the reciprocal document frequency (tf) (idf). The words with higher scores of weight are deemed to be more significant.

Usually, the tf-idf weight consists of two terms-

1. **Normalized Term Frequency (tf)**
2. **Inverse Document Frequency (idf)**

```
tf-idf(t, d) = tf(t, d) * idf(t)
```