SRM INSTITUTE OF SCIENCE & TECHNOLOGY

DEPARTMENT OF COMPUTING TECHNOLOGIES

21CSE356T
NATURAL LANGUAGE PROCESSING

UNIT- 1

S.PRABU
Assistant Professor
C-TECH-SRM-IST-KTR

UNIT-1

Overview and Word Level Analysis 9 Hour Introduction to Natural Language Processing, Applications of NLP, Levels of NLP, Regular Expressions, Morphological Analysis, Tokenization, Stemming, Lemmatization, Feature extraction: Term Frequency (TF), Inverse Document Frequency (IDF), Modeling using TF-IDF, Parts of Speech Tagging, Named Entity Recognition, N-grams, Smoothing.

Topic : 1 INTRODUCTION TO NATURAL LANGUAGE PROCESSING (NLP)

What is NLP?

Natural Language Processing (NLP) is a branch of Artificial Intelligence (AI) that helps computers understand, interpret, and respond to human language (text or speech). It bridges the gap between human communication and machine understanding.

Natural Language Processing (NLP) plays a crucial role in bridging the gap between human communication and machine understanding. It enables computers to process large volumes of text or speech data, extracting meaningful insights and patterns. NLP is widely used in industries such as healthcare (for analyzing patient records), finance (for detecting fraudulent transactions), and customer service (for automated support systems). Modern NLP systems rely on deep learning models that learn from vast datasets to improve their accuracy and understanding. As technology advances, NLP continues to enhance human-computer interaction, enabling more personalized and intelligent digital experiences.

Why is NLP Important?

NLP is used in various real-world applications, such as:

- **Search Engines** (Google, Bing) – Helps understand search queries
- **Chatbots & Virtual Assistants** (Siri, Alexa) – Responds to human language
- **Translation Services** (Google Translate) – Converts text between languages

- **Spam Detection** – Filters unwanted emails
- **Sentiment Analysis** – Identifies opinions from reviews and social media

How Does NLP Work?

NLP works in multiple steps:

1. **Text Processing** – Cleaning and organizing text
2. **Understanding Meaning** – Identifying important words and their relationships
3. **Generating Output** – Providing responses based on processed data

History & Evolution of NLP

- **1950s** – Alan Turing proposed the "Turing Test" for AI
- **1960s-70s** – Early rule-based language models (ELIZA chatbot)
- **1980s-90s** – Machine Learning techniques introduced
- **2000s-Present** – Deep Learning and Transformer models (BERT, GPT)

Challenges in NLP

- **Ambiguity** – Words can have multiple meanings (e.g., "bank" as a financial institution or riverbank)
- **Context Understanding** – Computers struggle with sarcasm and emotions
- **Grammar & Syntax Variations** – Different languages and dialects

Topic 2 : APPLICATIONS OF NLP

Natural Language Processing (NLP) is widely used across various industries to enable machines to understand, process, and generate human language. Below are some key applications with detailed points:

1. Machine Translation

- Converts text from one language to another (e.g., Google Translate, Microsoft Translator).
- Uses deep learning and transformer models (e.g., BERT, GPT, T5) for improved accuracy.

- Helps in cross-border communication and global business expansion.
- Used in real-time translation apps like Skype Translator and mobile translation assistants.

2. Chatbots & Virtual Assistants

- Helps in customer support, answering FAQs, and guiding users (e.g., ChatGPT, Bard, IBM Watson).
- Virtual assistants like Siri, Alexa, and Google Assistant respond to voice commands.
- Uses NLP techniques like intent recognition and dialogue management. Reduces workload on human agents and improves customer service response time.

3. Sentiment Analysis

- Analyzes emotions in social media, product reviews, and customer feedback. Categorizes sentiments as positive, negative, or neutral.
- Helps businesses improve products by understanding customer opinions.
- Used in brand reputation monitoring and political opinion analysis.

4. Speech Recognition

- Converts spoken words into written text (e.g., Google Voice, Siri, Cortana).
- Used in voice-controlled systems, medical transcription, and virtual assistants.
- Helps in accessibility for people with disabilities (e.g., voice-to-text software).
- Powers real-time meeting transcription tools like Otter.ai and Zoom captions.

5. Text Summarization

- Generates concise summaries of long documents or articles.
- Used in news aggregation platforms (e.g., Inshorts, AI-generated news summaries).
- Helps researchers and professionals quickly scan lengthy reports.

- Two types: Extractive Summarization (selecting key sentences) and Abstractive Summarization (creating new sentences).

6. Named Entity Recognition (NER)

- Identifies key entities such as names, places, organizations, and dates in text.
- Used in financial reports, legal documents, and news articles.
- Helps in information retrieval and knowledge graph building.
- Supports fraud detection by identifying fake entities in transactions.

7. Spam Detection

- Filters spam emails, fraudulent messages, and phishing attacks.
- Used by Gmail, Outlook, and email service providers to detect suspicious messages.
- Employs machine learning models to classify emails as spam or legitimate.
- Helps in cybersecurity by identifying fake and harmful content.

8. Automatic Text Generation

- AI-powered models generate human-like text based on input prompts.
- Used in content creation, storytelling, and automated report generation.
- NLP-based tools like ChatGPT, Jasper AI, and Copy.ai assist writers.
- Helps businesses create marketing copy, product descriptions, and blog content.

9. Question Answering Systems

- Provides direct answers to questions instead of retrieving documents.
- Used in search engines (Google's Featured Snippets, Bing AI search).
- Helps in educational AI tutors that answer student queries.
- Powers customer support bots that provide instant responses.

10. Healthcare Applications

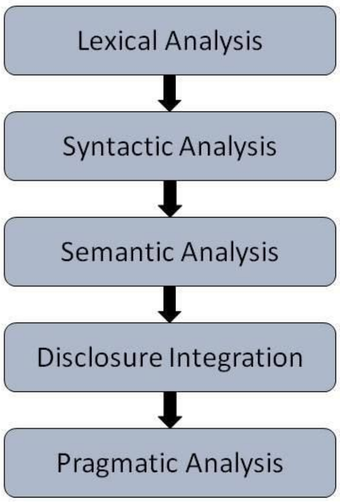
- Assists in medical transcription and electronic health record (EHR) analysis.
- NLP-powered chatbots help in preliminary diagnosis and patient assistance.
- Used in radiology report summarization and drug discovery research.
- Helps extract useful information from medical literature and research papers.

Topic 3 : Levels of NLP

The process of NLP can be divided into five distinct phases:

1. Lexical Analysis,
2. Syntactic Analysis,
3. Semantic Analysis,
4. Discourse Integration, and
5. Pragmatic Analysis.

Each phase plays a crucial role in the overall understanding and processing of natural language.



Phase I: Lexical or morphological analysis

The first phase of NLP is word structure analysis, which is referred to as lexical or morphological analysis. A lexicon is defined as a collection of words and phrases in a given language, with the analysis of this collection being the process of splitting the lexicon into components, based on what the user sets as parameters – paragraphs, phrases, words, or characters.

Similarly, morphological analysis is the process of identifying the morphemes of a word. A morpheme is a basic unit of English language construction, which is a small element of a word, that carries meaning. These can be either a free morpheme (e.g. walk) or a bound morpheme (e.g. -ing, -ed), with the difference between the two being that the latter cannot stand on it's own to produce a word with meaning, and should be assigned to a free morpheme to attach meaning.

For example, *irrationally* can be broken into *ir* (prefix), *rational* (root) and -*ly* (suffix). Lexical Analysis finds the relation between these morphemes and converts the word into its root form. A lexical analyzer also assigns the possible Part-Of-Speech (POS) to the word. It takes into consideration the dictionary of the language.

Phase II: Syntax analysis (parsing)

Syntax Analysis is the second phase of natural language processing. Syntax analysis or parsing is the process of checking grammar, word arrangement, and overall – the identification of relationships between words and whether those make sense. The process involved examination of all words and phrases in a sentence, and the structures between them.

As part of the process, there's a visualisation built of **semantic relationships referred to as a syntax tree** (like a knowledge graph). This process ensures that the structure and order and grammar of sentences makes sense, when considering the words and phrases that make up those sentences. Syntax analysis also involves tagging words and phrases with POS tags. There are two common methods, and multiple approaches to construct the syntax tree – top-down and bottom-up, however, both are logical and check for sentence formation, or else they reject the input.

Syntax Analysis ensures that a given piece of text is correct structure. It tries to parse the sentence to check correct grammar at the sentence level. Given the possible

POS generated from the previous step, a syntax analyzer assigns POS tags based on the sentence structure.

For example:

Correct Syntax: *Sun rises in the east.*

Incorrect Syntax: *Rise in sun the east.*

Phase III: Semantic analysis

Semantic analysis is the third stage in NLP, when an analysis is performed to understand the meaning in a statement. This type of analysis is focused on uncovering the definitions of words, phrases, and sentences and identifying whether the way words are organized in a sentence makes sense semantically.

This task is performed by **mapping the syntaxic structure**, and checking for logic in the presented relationships between entities, words, phrases and sentences in the text. There are a couple of important functions of semantic analysis, which allow for natural language understanding:

- To ensure that the data types are used in a way that's consistent with their definition.
- To ensure that the flow of the text is consistent.
- Identification of synonyms, antonyms, homonyms, and other lexical items.
- Overall word sense disambiguation.
- Relationship extraction from the different entities identified from the text.

Consider the sentence: "The apple ate a banana". Although the sentence is syntactically correct, it doesn't make sense because apples can't eat. Semantic analysis looks for meaning in the given sentence. It also deals with combining words into phrases.

Phase IV: Discourse integration

Discourse integration is the fourth phase in NLP, and simply means contextualisation. Discourse integration is the analysis and identification of the larger context for any smaller part of natural language structure (e.g. a phrase, word or sentence).

During this phase, it's important to ensure that each phrase, word, and entity mentioned are mentioned within the appropriate context. This analysis involves considering not only sentence structure and semantics, but also sentence combination and meaning of the text as a whole.

Otherwise, when analyzing the structure of text, sentences are broken up and analyzed and also considered in the context of the sentences that precede and follow them, and the impact that they have on the structure of text. Some common tasks in this phase include: **information extraction, conversation analysis, text summarisation, discourse analysis**.

Discourse deals with the effect of a previous sentence on the sentence in consideration. In the text, “Jack is a bright student. He spends most of the time in the library.” Here, discourse assigns “he” to refer to “Jack”.

Phase V: Pragmatic analysis

Pragmatic analysis is the fifth and final phase of NLP, that learnings from all other, preceding phases of NLP. Pragmatic analysis involves the process of abstracting or extracting meaning from the use of language, and translating a text, using the gathered knowledge from all other NLP steps performed beforehand.

The overall communicative and social content, as well as its impact on interpretation, are the focus of pragmatic analysis. Pragmatic Analysis uses a set of rules that describe cooperative dialogues to help you find the intended result. It covers things like word repetition, who said what to whom, and so on. It comprehends how people communicate with one another, the context in which they converse, and a variety of other factors. It refers to the process of abstracting or extracting the meaning of a situation's use of language. It translates the given text using the knowledge gathered in the preceding stages. “Switch on the TV” when used in a sentence, is an order or request to switch the TV on.

Topic 4 : Regular Expression

Regular Expressions (RegEx) are patterns used to search, match, or manipulate text efficiently. In Natural Language Processing (NLP), RegEx helps in text preprocessing, information extraction, tokenization, and pattern-based text filtering.

Why Use Regular Expressions in NLP?

Text Cleaning – Remove unwanted symbols, special characters, and extra spaces.
Tokenization – Split text into words or sentences.
Named Entity Recognition (NER) – Extract specific patterns (e.g., dates, emails, phone numbers).
Part-of-Speech (POS) Tagging – Identify word structures in text.
Information Extraction – Find important words or phrases in a document.

Common RegEx Patterns in NLP

Pattern	Matches	Example
`\d+`	Digits (0-9)	`"123", "42"`
`\w+`	Words (letters, digits, underscore)	`"Hello", "NLP_123"`
`\s+`	Whitespace (spaces, tabs, newlines)	`" "`
`[a-zA-Z]+`	Only letters	`"Hello", "Python"`
`[^a-zA-Z]`	Non-letters	`"$100", "@home!"`

Applications of RegEx in NLP

1. Text Preprocessing
    - Remove HTML tags from web-scraped text
    - Normalize text (convert to lowercase, remove punctuation)
    - Filter out special characters
2. Named Entity Recognition (NER)
    - Identify names, dates, and locations in text

- Extract structured information from unstructured data

3. Information Retrieval
   - Search for keywords or phrases in documents
   - Find patterns in log files, databases, or chat messages
4. Spell Checking
   - Detect misspelled words and correct them
   - Example: "teh" → "the" using regex-based dictionaries
5. Pattern-Based Sentiment Analysis
   - Identify positive or negative phrases
   - Example: "not happy", "very good", "extremely bad"

Topic 5 : Morphological Analysis in NLP

What is Morphological Analysis?

Morphological analysis is the process of studying the structure, formation, and components of words in a language. It helps in understanding how words are formed from roots, prefixes, and suffixes and how they change in different contexts.

Why is Morphological Analysis Important?

- Helps in **text normalization** (converting words to their base form).
- Improves **search engines** by recognizing different forms of a word.
- Aids in **machine translation, speech recognition, and information retrieval**.

Components of Morphology

1. **Inflectional Morphology**
   - Changes the **tense, number, or gender** of a word without altering its meaning.
   - Example:
     - "run" → "running" (present participle)
     - "book" → "books" (plural)

2. **Derivational Morphology**
    - Creates new words by adding prefixes or suffixes, changing the word's meaning.
    - Example:
        - "happy" → "unhappy" (prefix changes meaning)
        - "teach" → "teacher" (suffix changes word type)

Key Techniques in Morphological Analysis

1. Tokenization

Tokenization is one of the most common tasks in text processing. It is the process of separating a given text into smaller units called tokens.

An input text is a group of multiple words which make a sentence. We need to break the text in such a way that machines can understand this text and tokenization helps us to achieve that.

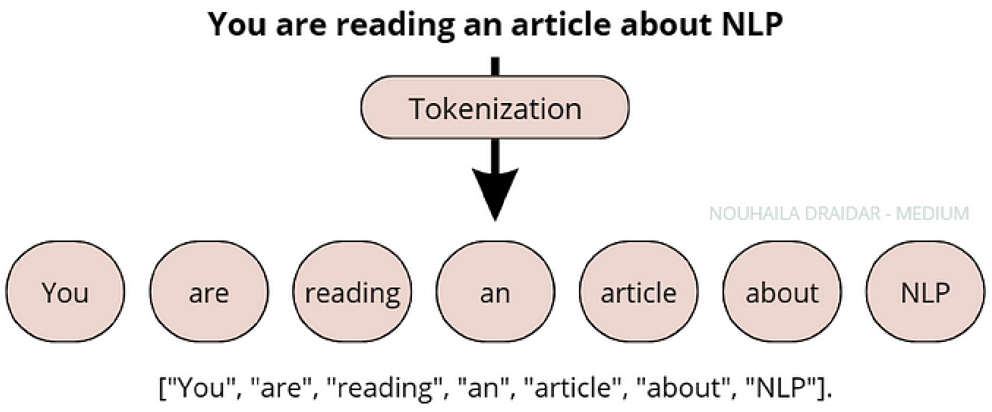
2. **Stemming**
    - Reduces a word to its root form by removing prefixes/suffixes.
    - Example:
        - "running" → "run"
        - "happiness" → "happi" (incorrect root in some cases)
3. **Lemmatization**
    - Converts a word to its dictionary (base) form using language rules.
    - Example:
        - "running" → "run"
        - "better" → "good" (handles irregular words)

Applications of Morphological Analysis

- **Search Engines:** Recognizes different word forms (e.g., "run" and "running").
- **Speech Recognition:** Understands spoken words by analyzing their structure.
- **Machine Translation:** Helps translate words with correct forms.
- **Text Analysis & NLP Models:** Used for **sentiment analysis, information retrieval, and chatbot development**.

Topic 6: TOKENIZATION

It is the act of breaking down a text into individual units, usually words or phrases, these fragments named tokens, enable machines to navigate and understand the complexities of human language.


You are reading an article about NLP
Tokenization
NOUHAILA DRAIDAR - MEDIUM
You
are
reading
an
article
about
NLP
["You", "are", "reading", "an", "article", "about", "NLP"].

Why is Tokenization Important?

- Text Preprocessing: Tokenization helps break down complex text into manageable parts, making it easier for machines to understand and process.
- Feature Extraction: Tokenized text can be used for tasks like sentiment analysis, language modeling, and machine translation.
- Building NLP Models: Models need tokenized data to learn patterns in language, such as identifying topics or understanding relationships between words.

Types of Tokenization

1. **Word Tokenization**
   - The text is divided into individual words.
   - Example:
     - Text: "I love programming."
     - Tokens: ["I", "love", "programming"]
2. **Sentence Tokenization**
   - The text is divided into sentences.
   - Example:
     - Text: "I love programming. It's fun!"
     - Tokens: ["I love programming.", "It's fun!"]

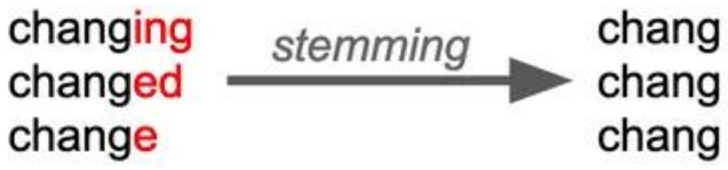
3. **Subword Tokenization**
   - The text is broken down into smaller units than words, typically used in modern NLP models like BERT or GPT.
   - Example:
     - Text: "unhappiness"
     - Tokens: ["un", "happiness"]
     - This is useful for dealing with out-of-vocabulary words or rare words.
     -

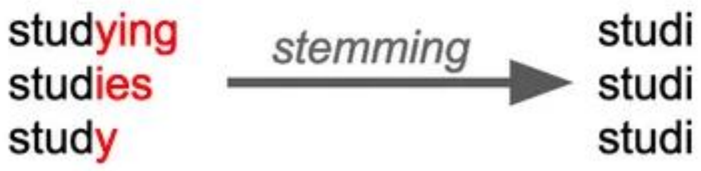
Tokenization Techniques

1. **Whitespace Tokenization**
   - The text is split based on spaces between words.
   - Example: "I love programming" → ["I", "love", "programming"]
   - Simple but not ideal for handling punctuation.
2. **Punctuation-Based Tokenization**
   - Treats punctuation marks as individual tokens.
   - Example: "Hello, world!" → ["Hello", ",", "world", "!"]
3. **Regular Expression Tokenization**
   - Uses regular expressions (regex) to split text based on patterns, allowing for more control.
   - Example: A regex pattern can capture words and punctuation marks separately.
4. **Byte Pair Encoding (BPE)**
   - A subword tokenization technique that splits rare words into more frequent subwords.
   - Example: "unhappiness" → ["un", "happiness"]
   - Commonly used in **neural machine translation** and large language models.

Topic 7 : STEMMING

Stemming is the process of reducing words to their root or base form by removing suffixes and prefixes. It helps in text normalization, making different forms of a word comparable in NLP tasks.





Why is Stemming Important?

- **Reduces word variations**: Helps group similar words together (e.g., "running," "runs," and "ran" → "run").
- **Improves search results**: Search engines use stemming to match different forms of a word.
- **Speeds up NLP tasks**: Reducing words to their root form decreases computational complexity.

Examples of Stemming

Word	**Stemmed Form**
Running	Run
Happily	Happi
Studies	Studi
Flying	Fli

Note: Stemming may not always return valid words (e.g., "happily" → "happi"), which is a limitation.

Types of Stemming Algorithms

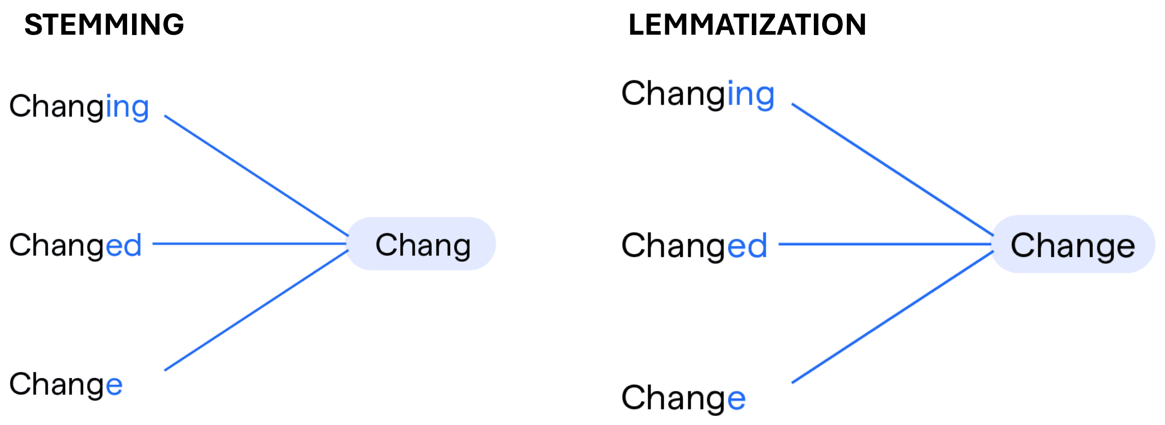
1. **Porter's Stemmer**
   - One of the most popular stemming algorithms.
   - Uses a set of **rules** to remove common suffixes.
   - Example: "crying" → "cri", "flies" → "fli".
2. **Lancaster Stemmer**
   - A more aggressive version of Porter's stemmer.
   - Removes larger chunks of words, sometimes over-stemming.
   - Example: "running" → "run", "happiness" → "happy".
3. **Snowball Stemmer (Porter2 Stemmer)**
   - An improved version of Porter's stemmer with better language handling.
   - Supports multiple languages.
   - Example: "running" → "run", "arguing" → "argu".
4. **Regex-Based Stemmer**
   - Uses **regular expressions** to remove suffixes.
   - Example: Removing "ing", "ed", or "ly" from words.

Limitations of Stemming

- **Over-Stemming**: Reduces words too much, making them hard to understand.
  - Example: "university" → "univers".
- **Under-Stemming**: Does not stem enough, leaving similar words ungrouped.
  - Example: "running" and "runner" remain different.
- **Not Language-Specific**: Most stemming algorithms work best for **English** and may not handle other languages effectively.

Topic 8 : LEMMATIZATION

Lemmatization is the process of reducing a word to its base or dictionary form (called a *lemma*) using linguistic rules. Unlike stemming, lemmatization ensures that the root word is a valid word.



Examples of Lemmatization

Word	**Lemma**
Running	Run
Better	Good
Studies	Study
Mice	Mouse

Difference Between Stemming and Lemmatization

Feature	**Stemming**	**Lemmatization**
Approach	Removes suffixes/prefixes	Uses language rules & dictionary
Output	May not be a real word	Always a valid word
Example	"Caring" → "Car"	"Caring" → "Care"
Accuracy	Lower	Higher
Speed	Faster	Slower

How Lemmatization Works?

1. **Identifies the word's part of speech (POS)** (e.g., noun, verb).
2. **Looks up the word in a dictionary** for its lemma.
3. **Returns the correct base form** (e.g., "better" → "good").

Types of Lemmatization

1. **Dictionary-Based Lemmatization**: Uses predefined word lists to find the lemma.
2. **Rule-Based Lemmatization**: Uses grammar rules to determine the lemma.
3. **POS-Aware Lemmatization**: Considers the part of speech before lemmatizing (e.g., "saw" as a noun stays "saw", but as a verb becomes "see").

Challenges in Lemmatization

- **Context Sensitivity**: "Left" could mean past tense of "leave" or a direction.
- **Computational Cost**: Slower than stemming due to dictionary lookups.
- **Language-Specific Rules**: Lemmatization must be adapted for different languages.

Topic 9 FEATURE EXTRACTION

Feature extraction is the process of converting raw text into numerical representations that machine learning models can process. Since computers cannot directly understand text, NLP techniques extract meaningful features from text data for analysis.

Why is Feature Extraction Important?

- Helps machine learning models understand text.
- Converts unstructured data (text) into structured numerical data.
- Improves text classification, sentiment analysis, and chatbot development.

Types of Feature Extraction Techniques

1. Term Frequency (TF)
2. Inverse Document Frequency (IDF)
3. Part-of-Speech (POS) tagging
4. Named Entity Recognition (NER)
5. N-grams

Topic 10 : Term Frequency (TF) in NLP

What is Term Frequency (TF)?

Term Frequency (TF) measures how frequently a word appears in a document relative to the total number of words in that document. It helps determine the importance of a word in a given text.

Formula for TF

$$TF(w) = \frac{\text{Number of times word } w \text{ appears in the document}}{\text{Total number of words in the document}}$$

Example of TF Calculation

Consider a document:

"Natural Language Processing (NLP) is a field of AI. NLP helps machines understand human language."

Word	Frequency	TF Calculation
NLP	2	2 / 12 = 0.167
language	2	2 / 12 = 0.167
human	1	1 / 12 = 0.083
AI	1	1 / 12 = 0.083

Why is TF Important?

- Highlights frequently used words in a document.
- Helps in text analysis and classification.
- Used in TF-IDF (Term Frequency-Inverse Document Frequency) to rank words in search engines.

Limitations of TF

- Common words (e.g., "the", "is", "a") may have high TF but add little meaning.
- TF alone does not consider how important a word is in a larger collection of documents (handled by IDF).

Topic 11 : Inverse Document Frequency (IDF) in NLP

What is IDF?

Inverse Document Frequency (IDF) measures how important a word is across multiple documents. It helps reduce the impact of common words (like "the", "is", "and") and highlights rare but important words.

Formula for IDF

$$IDF(w) = \log\left(\frac{N}{DF(w)}\right)$$

.

Where:

- N = Total number of documents
- $DF(w)$ = Number of documents containing the word w

How IDF Works?

- If a word appears in **many documents**, its IDF value is **low** (not important).
- If a word appears in **few documents**, its IDF value is **high** (important).

Example of IDF Calculation

Consider **3 documents**:

1. **Doc 1**: "NLP is a subfield of AI."
2. **Doc 2**: "AI and NLP help machines understand human language."
3. **Doc 3**: "Deep learning is useful for NLP."

Word	DF (No. of Docs)	IDF Calculation
NLP	3	log(3/3) = 0.0
AI	2	log(3/2) = 0.176
learning	1	log(3/1) = 0.477

- NLP" appears in all 3 documents, so IDF = 0 (not unique).
- "Learning" appears in only one document, so it has a higher IDF (more unique).

Why is IDF Important?

- Helps filter out common words in large datasets.
- Improves text search by ranking rare, meaningful words higher.
- Used in TF-IDF (combining TF & IDF) to improve document ranking.

Topic 12 : Modelling Using TF-IDF in NLP

What is TF-IDF?

TF-IDF (Term Frequency-Inverse Document Frequency) is a numerical statistic that helps convert text into a format that machine learning models can process. It assigns a weight to each word based on its frequency in a document (TF) and its uniqueness across multiple documents (IDF).

Formula for TF-IDF

$$TF\text{-}IDF(w) = TF(w) \times IDF(w)$$

Where:

- TF (Term Frequency): Measures how often a word appears in a document.
- IDF (Inverse Document Frequency): Reduces the weight of common words across documents.
- The term frequency (TF) is a measure of how frequently a term appears in a document. It is calculated by dividing the number of times a term appears in a document by the total number of words in the document. The resulting value is a number between 0 and 1.
- The inverse document frequency (IDF) is a measure of how important a term is across all documents in the corpus. It is calculated by taking the logarithm of the total number of documents in the corpus divided by the number of documents in which the term appears. The resulting value is a number greater than or equal to 0.

The TF-IDF score is calculated by multiplying the term frequency and the inverse document frequency. The higher the TF-IDF score, the more important the term is in the document.

TF-IDF = TF * IDFTF-IDF = TF * log(N/DF)

Where:

- TF is the term frequency of a word in a document
- N is the total number of documents in the corpus
- DF is the document frequency of a word in the corpus (i.e., the number of documents that contain the word)

Suppose we have a corpus of five documents

Doc1: The quick brown fox jumps over the lazy dog

Doc2: The lazy dog likes to sleep all day

Doc3: The brown fox prefers to eat cheese

Doc4: The red fox jumps over the brown fox

Doc5: The brown dog chases the fox

Now, let’s say we want to calculate the TF-IDF scores for the word “fox” in each of these documents.

Step 1: Calculate the term frequency (TF)

The term frequency (TF) is the number of times the word appears in the document. We can calculate the TF for the word “fox” in each document as follows:

TF = (Number of times word appears in the document) / (Total number of words in the document)

Doc1: 1 / 9

Doc2: 0 / 8

Doc3: 1 / 7

Doc4: 2 / 8

Doc5: 1 / 6

Step 2: Calculate the document frequency (DF):

The document frequency (DF) is the number of documents in the corpus that contain the word. We can calculate the DF for the word “fox” as follows:

DF = 4 (Doc1, Doc3, Doc4 and Doc5)

Step 3: Calculate the inverse document frequency (IDF):

The inverse document frequency (IDF) is a measure of how rare the word is across the corpus. It is calculated as the logarithm of the total number of documents in the corpus divided by the document frequency. In our case, we have:

IDF = log(5/4) = 0.2231

Step 4: Calculate the TF-IDF score:

The TF-IDF score for the word “fox” in each document can now be calculated using the following formula:

TF-IDF = TF * IDF

Doc1: 1/9 * 0.2231 = 0.0247

Doc2: 0/8 * 0.2231 = 0

Doc3: 1/7 * 0.2231 = 0.0318

Doc4: 2/8 * 0.2231 = 0.0557

Doc5: 1/6 * 0.2231 = 0.0372

Therefore, the TF-IDF score for the word “fox” is highest in Doc4 indicating that this word is relatively important in this document compared to the rest of the corpus. On the other hand, the TF-IDF score is zero in Doc2, indicating that the word “fox” is not relevant in this document.

Advantages of TF-IDF

Some of the advantages of using TF-IDF include:

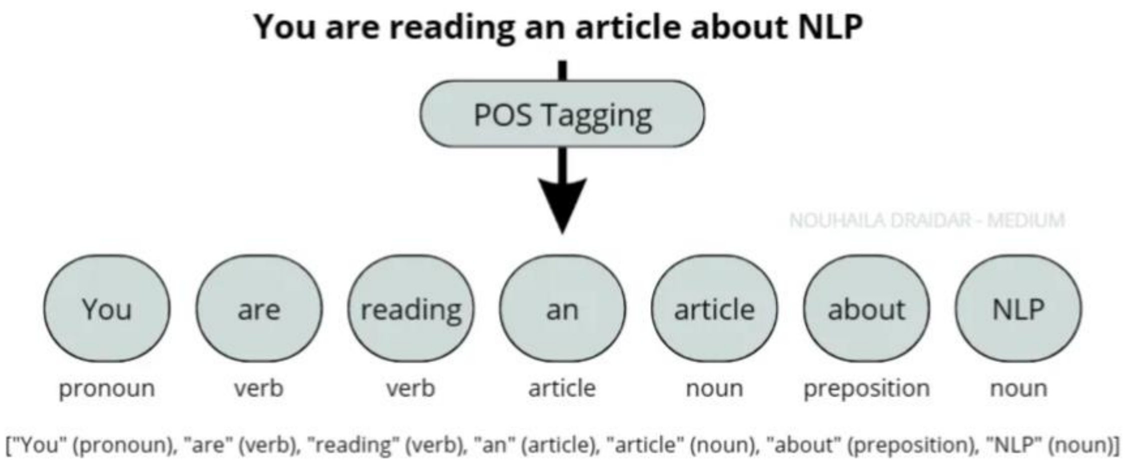
1. Measures relevance: TF-IDF measures the importance of a term in a document, based on the frequency of the term in the document and the inverse document frequency (IDF) of the term across the entire corpus. This helps to identify which terms are most relevant to a particular document.
2. Handles large text corpora: TF-IDF is scalable and can be used with large text corpora, making it suitable for processing and analyzing large amounts of text data.
3. Handles stop words: TF-IDF automatically down-weights common words that occur frequently in the text corpus (stop words) that do not carry much meaning or importance, making it a more accurate measure of term importance.
4. Can be used for various applications: TF-IDF can be used for various natural language processing tasks, such as text classification, information retrieval, and document clustering.
5. Interpretable: The scores generated by TF-IDF are easy to interpret and understand, as they represent the importance of a term in a document relative to its importance across the entire corpus.
6. Works well with different languages: TF-IDF can be used with different languages and character encodings, making it a versatile technique for processing multilingual text data.

Limitations of TF-IDF

- Ignores the context
- Assumes independence
- Vocabulary size
- Sensitivity to stopwords

Topic 13 : PARTS OF SPEECH TAGGING

Parts of Speech (POS) Tagging is the process of assigning grammatical labels (such as noun, verb, adjective) to words in a sentence. It helps machines understand the structure and meaning of text.


You are reading an article about NLP
POS Tagging
NOUHAILA DRAIDAR - MEDIUM
You
pronoun
are
verb
reading
verb
an
article
article
noun
about
preposition
NLP
noun
["You" (pronoun), "are" (verb), "reading" (verb), "an" (article), "article" (noun), "about" (preposition), "NLP" (noun)]

Example of POS Tagging

Sentence:

"The quick brown fox jumps over the lazy dog."

Word	**POS Tag**
The	Determiner (DT)
quick	Adjective (JJ)
brown	Adjective (JJ)
fox	Noun (NN)
jumps	Verb (VBZ)
over	Preposition (IN)
the	Determiner (DT)
lazy	Adjective (JJ)
dog	Noun (NN)

Why is POS Tagging Important?

- Text Processing : Helps in lemmatization (reducing words to root form).
- Speech Recognition : Identifies word meanings based on context.
- Machine Translation: Improves grammatical accuracy in translations.
- Named Entity Recognition (NER) : Differentiates names from common words.

Types of POS Tagging Methods

1. Rule-Based POS Tagging

- Uses predefined grammatical rules to tag words.
- Example: If a word ends in "-ing", it is likely a verb.

2. Stochastic (Probabilistic) POS Tagging

- Uses statistical models like Hidden Markov Models (HMM) and Naïve Bayes.
- Assigns the most likely POS tag based on probabilities.

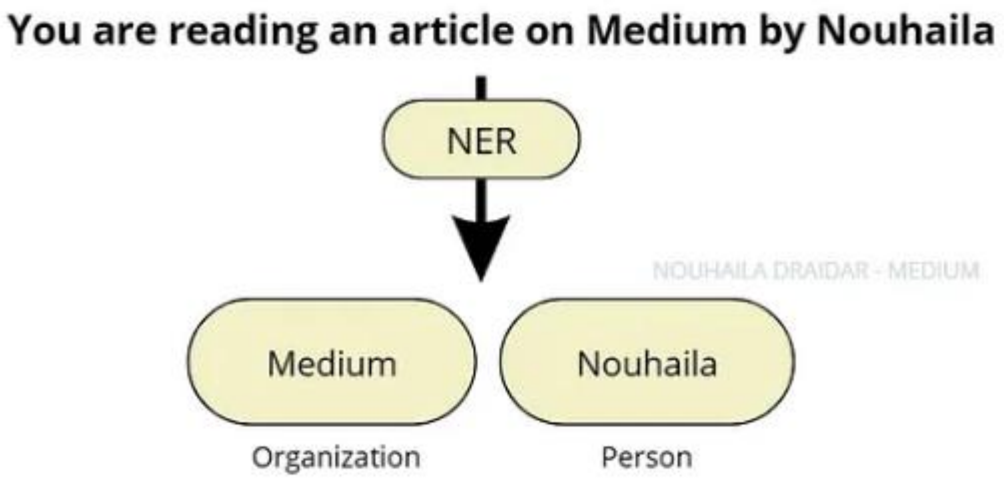
3. Machine Learning-Based POS Tagging

- Uses Supervised Learning (like Decision Trees, CRFs).
- Trained on labeled datasets such as the Penn Treebank.
- Example models: NLTK, spaCy, Stanford POS Tagger.

Topic 14 : NAMED ENTITY RECOGNITION

Named Entity Recognition (NER) is an NLP technique used to identify and classify named entities in text into predefined categories such as:

- **Person names** (e.g., "Albert Einstein")
- **Locations** (e.g., "New York", "India")
- **Organizations** (e.g., "Google", "NASA")
- **Dates & Times** (e.g., "January 1, 2025")
- **Monetary values** (e.g., "$100", "50 Euros")
- **Other specific entities** (e.g., "COVID-19", "Python programming")


You are reading an article on Medium by Nouhaila
NER
NOUHAILA DRAIDAR - MEDIUM
Medium
Nouhaila
Organization
Person

Example of NER

Sentence:

"Elon Musk is the CEO of Tesla, which is headquartered in California."

Entity	**Category**
Elon Musk	PERSON
Tesla	ORGANIZATION
California	LOCATION

Why is NER Important?

Information Extraction	– Extracts key entities from large texts.
Search Engines	– Improves query accuracy by recognizing entities.
Chatbots & Virtual Assistants	– Helps understand user queries better.
Medical NLP	– Identifies diseases, drugs, and patient names.
Finance & News Analysis	– Detects company names and stock symbols.

Methods for NER

1. Rule-Based Approach

- Uses predefined patterns (e.g., capitalized words as names).
- Example: If a word starts with “Dr.”, it’s likely a person’s name.

2. Statistical & Machine Learning-Based Approach

- Uses Hidden Markov Models (HMMs), CRFs, and Neural Networks.

- Requires a large labeled dataset (like CoNLL-2003).

3. Deep Learning-Based Approach

- Uses Transformer models (BERT, spaCy, NLTK, etc.) for better accuracy.
- Fine-tuned on domain-specific data for healthcare, finance, etc.

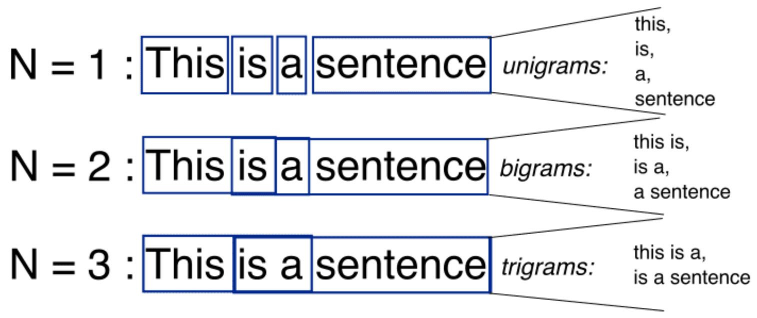
Topic 15 : N-grams

What are N-grams?

N-grams are continuous sequences of N words from a given text. They help analyze word patterns and relationships in Natural Language Processing (NLP).

Types of N-grams

- **Unigram (N=1)** → Single words
- **Bigram (N=2)** → Two-word sequences
- **Trigram (N=3)** → Three-word sequences
- **4-gram, 5-gram, etc.** → Longer sequences



Examples of N-grams

Sentence:

"Natural Language Processing is amazing!"

Type	N-grams
Unigram	Natural, Language, Processing, is, amazing
Bigram	Natural Language, Language Processing, Processing is, is amazing
Trigram	Natural Language Processing, Language Processing is, Processing is amazing

Why Use N-grams in NLP?

Text Prediction – Used in search engines & keyboards (e.g., Google Auto-Suggest).

Speech Recognition – Helps predict the next word in a sentence.

Plagiarism Detection – Identifies repeated phrases in documents.

Sentiment Analysis – Helps detect phrase-based sentiment (e.g., "not good").

Machine Translation – Identifies common word patterns across languages.

Limitations of N-grams

Ignores long-range dependencies – Can't capture meaning across entire sentences. Data Sparsity – Higher N-grams require large datasets for accuracy. Memory Intensive – Large N-grams need more storage & processing power.

What is Smoothing?

Smoothing is a technique used in **Natural Language Processing (NLP) and Probability Estimation** to handle **zero probabilities** in language models. It helps prevent **unseen words or N-grams** from having a probability of zero, improving the robustness of NLP models.

Why is Smoothing Needed?

- When using **N-gram language models**, some word sequences might not appear in the training data.
- If an unseen N-gram has **zero probability**, it can **completely disrupt probability-based text generation or classification**.
- Smoothing assigns **small nonzero probabilities** to unseen N-grams, ensuring better model performance.

Topic 16 : SMOOTHING

Smoothing is a technique used in Natural Language Processing (NLP) and Probability Estimation to handle zero probabilities in language models. It helps prevent unseen words or N-grams from having a probability of zero, improving the robustness of NLP models.

Why is Smoothing Needed?

- When using N-gram language models, some word sequences might not appear in the training data.
- If an unseen N-gram has zero probability, it can completely disrupt probability-based text generation or classification.
- Smoothing assigns small nonzero probabilities to unseen N-grams, ensuring better model performance.

Types of Smoothing Techniques

1. Laplace Smoothing (Add-One Smoothing)

- Adds **1** to every count, ensuring no probability is zero.
- Used in **text classification and N-gram models**.

Formula:

$$P(w_i|w_{i-1}) = \frac{C(w_{i-1}, w_i) + 1}{C(w_{i-1}) + V}$$

where:

- $C(w_{i-1}, w_i)$ = Count of bigram
- $C(w_{i-1})$ = Count of the previous word
- V = Vocabulary size

Example (Bigram Model without Smoothing):

Bigram	Count	Probability
"data science"	3	$\frac{3}{10}$ = 0.3
"science is"	2	$\frac{2}{10}$ = 0.2
"is amazing"	0	$\frac{0}{10}$ = 0.0 (Issue!)

With **Laplace Smoothing**:

Bigram	Count (Add 1)	Probability
"data science"	$3+1$	$\frac{4}{10+V}$
"science is"	$2+1$	$\frac{3}{10+V}$
"is amazing"	$0+1$	$\frac{1}{10+V}$

2. Additive Smoothing (Generalized Laplace Smoothing)

- A generalized version of Laplace Smoothing where **a small number (α)** is added instead of 1.

Formula:

$$P(w_i|w_{i-1}) = \frac{C(w_{i-1}, w_i) + \alpha}{C(w_{i-1}) + \alpha V}$$

where **α (alpha)** is a small number (e.g., 0.01).

◆ **Advantage**: More flexible than Laplace Smoothing.

3. Good-Turing Smoothing

- Adjusts probabilities of unseen N-grams based on **low-frequency words**.
- If a word **appears only once**, its count is adjusted using words appearing **twice, three times, etc.**.
- Used in **speech recognition** and **language modeling**.

Formula:

$$C^* = \frac{(C+1) \times N_{C+1}}{N_C}$$

where:

- C^* = Adjusted count
- N_C = Number of N-grams with count C
- N_{C+1} = Number of N-grams with count $C+1$

◆ **Advantage**: Works better for **very rare words**.

◆ **Disadvantage**: Can be computationally complex.