Introduction to Natural Language Processing

Natural Language Processing (NLP) is a field of artificial intelligence (AI) that focuses on the interaction between computers and human languages. It enables computers to process, analyze, and understand large amounts of natural language data, such as text or speech, in a way that is useful for a variety of applications.

NLP involves a combination of **linguistics** and **computer science**, using techniques from **machine learning** (ML), **deep learning** (DL), and **statistical models** to extract meaningful information from language. NLP allows machines to understand, interpret, and even generate human language, improving the way we interact with technology.

Key Applications of NLP

- **Contextual Advertisements**: Serving ads based on the content of a webpage.
- **Email Clients**: Features like spam filtering and smart reply.
- **Social Media**: Controlling content for display, identifying harmful content.
- **Search Engines**: Enhancing search accuracy by understanding user queries.
- **Chatbots**: Providing conversational interactions with users.
- **Caption Generation**: Automatically generating descriptions for images.
- **Text Summarization**: Condensing long texts into shorter summaries.
- **Auto Correct**: Correcting typing errors in real-time.
- **Market Intelligence**: Analyzing trends and sentiments in market data.

Important NLP Terminologies

- **Document**: A collection of many words, such as an article or a book.
- **Vocabulary**: The set of unique words present in a document or corpus.
- **Token**: The basic unit of text, often a word or punctuation mark.
- **Corpus**: A collection of documents.
- **Context**: The surrounding words or tokens that give meaning to a specific word in a document.
- **Vector Embedding**: A numerical representation of words or text, often used in machine learning models to analyze language.

Working of Natural Language Processing (NLP)

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/Preparation

Preprocessing is crucial to clean and prepare the raw text data for analysis. Common preprocessing steps include:

- **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, removing emojis, removing Html tags, expanding contractions, and handling special characters.
- **Tokenization**: Splitting text into smaller units like words or sentences.

3. Text Representation

- **One Hot Encoding**
- **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.

4. Feature Extraction

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

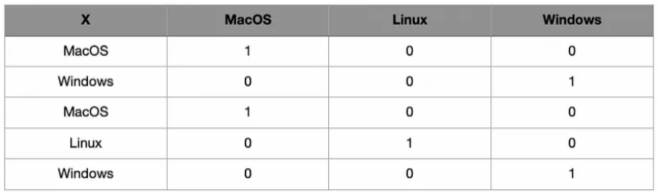
5. Modelling

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

- **Supervised Learning**: Using labeled 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 modeling (e.g., Latent Dirichlet Allocation) on unlabeled data.

Prominent Techniques of Embedding

1. One Hot Encoding



X	MacOS	Linux	Windows
MacOS	1	0	0
Windows	0	0	1
MacOS	1	0	0
Linux	0	1	0
Windows	0	0	1

2. Bag of Words (BoW)

The **Bag of Words (BoW)** model is a fundamental technique in Natural Language Processing (NLP) used for text representation. It simplifies the textual data by converting it into a numerical format that machine learning models can easily process. The BoW model treats a document or sentence as a collection (or "bag") of individual words, disregarding grammar and word order. Each word's frequency is counted, and this information is used to represent the document in a structured way.

In BoW, each unique word in a text corpus becomes a feature in a dataset. The text data is transformed into a matrix of word frequencies, where each row corresponds to a document, and each column corresponds to a unique word in the corpus. The value at each position in the matrix is the frequency of that word in the document.

Steps Involved in the Bag of Words Model

1. **Tokenization**: Splitting the text into individual words or tokens.
2. **Vocabulary Creation**: Identifying all the unique words (vocabulary) from the entire corpus.
3. **Vectorization**: Converting each document into a vector where each element represents the frequency of a unique word in that document.

Practical Example of Bag of Words

Consider a small dataset consisting of three short sentences:

1. "I love programming in Python"
2. "Python is great for data science"
3. "I enjoy learning Python and data science"

Step 1: Tokenization

The first step is to break each sentence into individual words (tokens).

- Sentence 1: ["I", "love", "programming", "in", "Python"]
- Sentence 2: ["Python", "is", "great", "for", "data", "science"]
- Sentence 3: ["I", "enjoy", "learning", "Python", "and", "data", "science"]

Step 2: Vocabulary Creation

The vocabulary consists of all unique words from the dataset. In this case, the vocabulary is:

["I", "love", "programming", "in", "Python", "is", "great", "for", "data", "science", "enjoy", "learning", "and"]

Step 3: Vectorization (Creating the Word Frequency Matrix)

Now, we convert each sentence into a vector where each element represents the frequency of a word from the vocabulary in that sentence. Here's how each sentence would be represented:

Sentence \ Vocabulary	I	love	programming	in	Python	is	great	for	data
Sentence 1	1	1	1	1	1	0	0	0	0
Sentence 2	0	0	0	0	1	1	1	1	1
Sentence 3	1	0	0	0	1	0	0	0	1

Advantages of Bag of Words

- **Simplicity**: The BoW model is easy to understand and implement.

- **Efficiency**: It converts text into a numerical format that machine learning models can easily process.

Limitations of Bag of Words

- **Word Order Ignorance**: BoW disregards the order of words, which might result in losing meaningful relationships between words.
- **Sparse Matrix**: As the vocabulary grows, the matrix becomes sparse (many zeros), leading to inefficient storage and computation.

Q3.(Bag of Words & Similarity)

You are provided with the following five sentences.

Sentences:

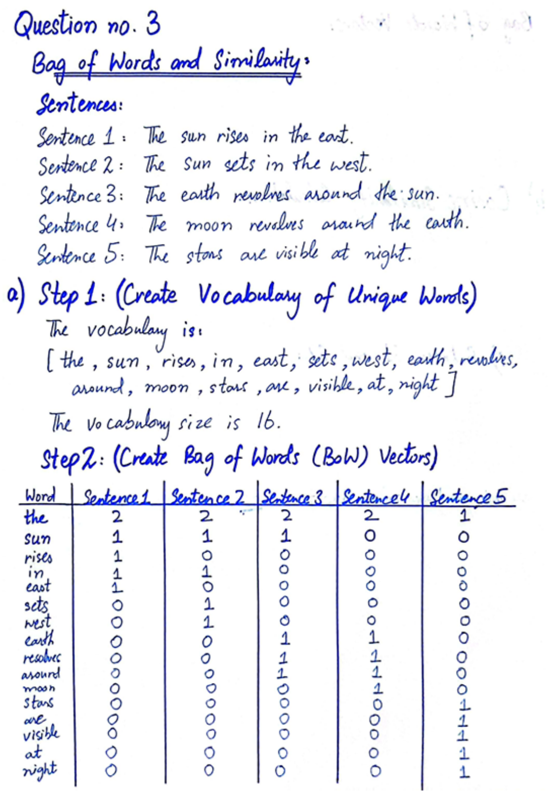
Sentence 1: The sun rises in the east.

Sentence 2: The sun sets in the west.

Sentence 3: The earth revolves around the sun.

Sentence 4: The moon revolves around the earth.

Sentence 5: The stars are visible at night.



Question no. 3

Bag of Words and Similarity:

Sentences:

Sentence 1: The sun rises in the east.

Sentence 2: The sun sets in the west.

Sentence 3: The earth revolves around the sun.

Sentence 4: The moon revolves around the earth.

Sentence 5: The stars are visible at night.

a) Step 1: (Create Vocabulary of Unique Words)

The vocabulary is:

[the, sun, rises, in, east, sets, west, earth, revolves, around, moon, stars, are, visible, at, night]

The vocabulary size is 16.

Step 2: (Create Bag of Words (BoW) Vectors)

Word	Sentence 1	Sentence 2	Sentence 3	Sentence 4	Sentence 5
the	2	2	2	2	1
sun	1	1	1	0	0
rises	1	0	0	0	0
in	1	1	0	0	0
east	1	0	0	0	0
sets	0	1	0	0	0
west	0	1	0	0	0
earth	0	0	1	1	0
revolves	0	0	1	1	0
around	0	0	1	1	0
moon	0	0	0	1	0
stars	0	0	0	0	1
are	0	0	0	0	1
visible	0	0	0	0	1
at	0	0	0	0	1
night	0	0	0	0	1

3. Term Frequency-Inverse Document Frequency (TF-IDF)

Term Frequency-Inverse Document Frequency (TF-IDF) is an advanced technique in Natural Language Processing (NLP) used to represent text data. It extends the basic **Bag of Words (BoW)** model by not just considering the frequency of words in a document but also taking into account how common or rare a word is across multiple documents in a corpus. This helps to reduce the weight of commonly occurring words like "the," "is," or "and" while giving more importance to rarer, more informative words.

Key Concepts in TF-IDF

1. **Term Frequency (TF)**: Measures how frequently a term (word) occurs in a document.
2. **Inverse Document Frequency (IDF)**: Measures how important a word is by checking how rare it is across the entire corpus.
3. **TF-IDF Score**: A combination of TF and IDF, giving more weight to words that are frequent in a document but rare in the overall corpus.

TF-IDF Formula

- **Term Frequency (TF)**:

$$TF(t,d) = \frac{\text{Number of times term } t \text{ appears in document } d}{\text{Total number of terms in document } d}$$

TF measures how often a word appears in a particular document. Higher frequency gives a higher score.

- **Inverse Document Frequency (IDF)**:

$$IDF(t) = \log\left(\frac{\text{Total number of documents}}{\text{Number of documents containing the term } t}\right)$$

IDF helps identify rare words. If a word appears in many documents, its IDF score will be lower (since it's not unique).

- **TF-IDF**:

$$TF\text{-}IDF(t,d) = TF(t,d) \times IDF(t)$$

This score reflects the importance of a word in a document relative to the entire corpus. Words with high TF-IDF are considered highly relevant in their respective documents.

Practical Example of TF-IDF

Let's consider a small dataset of three documents:

1. **Document 1**: "Python is great for data science"
2. **Document 2**: "I love programming in Python"
3. **Document 3**: "Data science is fascinating"

Step 1: Calculate Term Frequency (TF)

Word	TF in Doc 1	TF in Doc 2	TF in Doc 3
Python	1/6	1/5	0/4
is	1/6	0/5	1/4
great	1/6	0/5	0/4
for	1/6	0/5	0/4
data	1/6	0/5	1/4
science	1/6	0/5	1/4
I	0/6	1/5	0/4
love	0/6	1/5	0/4
programming	0/6	1/5	0/4
fascinating	0/6	0/5	1/4

Step 2: Calculate Inverse Document Frequency (IDF)

Word	Document Frequency	IDF
Python	2/3	0.176
is	2/3	0.176
great	1/3	0.477
for	1/3	0.477
data	2/3	0.176
science	2/3	0.176
I	1/3	0.477
love	1/3	0.477
programming	1/3	0.477
fascinating	1/3	0.477

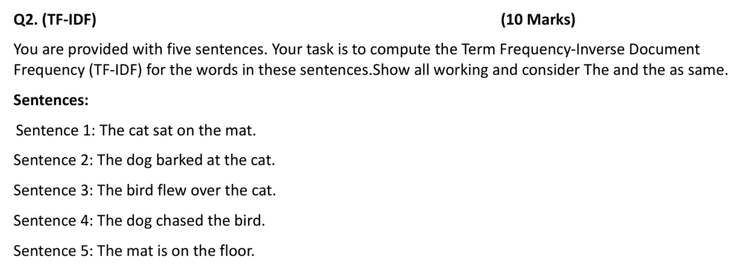
Step 3: Calculate TF-IDF

Word	TF-IDF in Doc 1	TF-IDF in Doc 2	TF-IDF in Doc 3
Python	0.029	0.035	0
is	0.029	0	0.044
great	0.079	0	0
for	0.079	0	0

data	0.029	0	0.044
science	0.029	0	0.044
I	0	0.095	0
love	0	0.095	0
programming	0	0.095	0
fascinating	0	0	0.119

Analysis

- Words like "great," "for," and "fascinating" have higher TF-IDF scores because they occur less frequently across the entire corpus, making them more distinctive for their respective documents.
- Common words like "is" and "Python" have lower TF-IDF scores because they appear in multiple documents, reducing their significance.



Q2. (TF-IDF) **(10 Marks)**

You are provided with five sentences. Your task is to compute the Term Frequency-Inverse Document Frequency (TF-IDF) for the words in these sentences.Show all working and consider The and the as same.

Sentences:

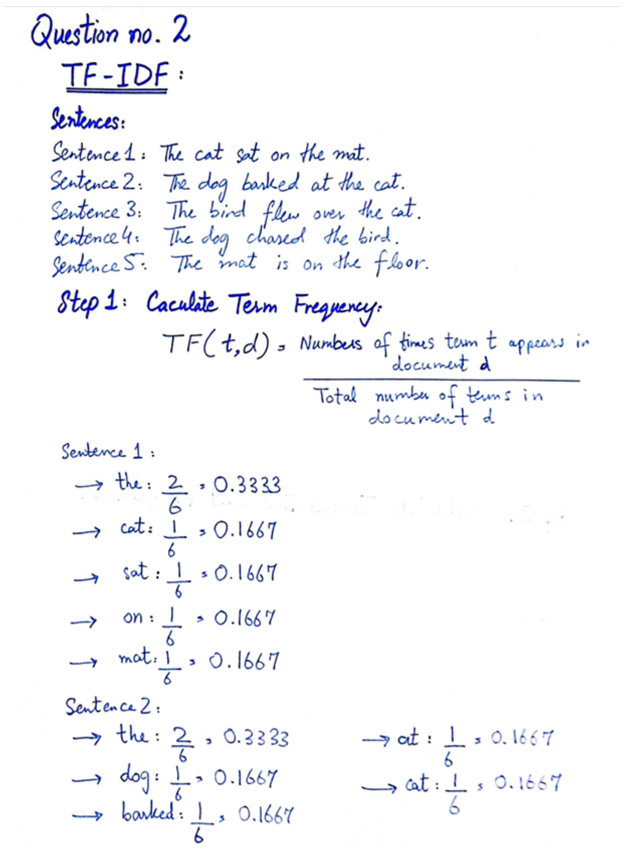
Sentence 1: The cat sat on the mat.

Sentence 2: The dog barked at the cat.

Sentence 3: The bird flew over the cat.

Sentence 4: The dog chased the bird.

Sentence 5: The mat is on the floor.



Question no. 2

TF-IDF :

Sentences:

Sentence 1: The cat sat on the mat.
Sentence 2: The dog barked at the cat.
Sentence 3: The bird flew over the cat.
sentence 4: The dog chased the bird.
Sentence 5: The mat is on the floor.

Step 1: Caculate Term Frequency:

$$TF(t,d) = \frac{\text{Numbers of times term } t \text{ appears in document } d}{\text{Total number of terms in document } d}$$

Sentence 1 :

$\rightarrow$ the : $\frac{2}{6} = 0.3333$

$\rightarrow$ cat : $\frac{1}{6} = 0.1667$

$\rightarrow$ sat : $\frac{1}{6} = 0.1667$

$\rightarrow$ on : $\frac{1}{6} = 0.1667$

$\rightarrow$ mat : $\frac{1}{6} = 0.1667$

Sentence 2 :

$\rightarrow$ the : $\frac{2}{6} = 0.3333$

$\rightarrow$ dog : $\frac{1}{6} = 0.1667$

$\rightarrow$ barked : $\frac{1}{6} = 0.1667$

$\rightarrow$ at : $\frac{1}{6} = 0.1667$

$\rightarrow$ cat : $\frac{1}{6} = 0.1667$

Sentence 3:

$\rightarrow$ the : $\frac{2}{6} = 0.3333$

$\rightarrow$ bird : $\frac{1}{6} = 0.1667$

$\rightarrow$ flew : $\frac{1}{6} = 0.1667$

$\rightarrow$ over : $\frac{1}{6} = 0.1667$

$\rightarrow$ cat : $\frac{1}{6} = 0.1667$

Sentence 4:

$\rightarrow$ the: $\frac{2}{5} = 0.4$

$\rightarrow$ dog: $\frac{1}{5} = 0.2$

$\rightarrow$ chased : $\frac{1}{5} = 0.2$

$\rightarrow$ bird : $\frac{1}{5} = 0.2$

Sentence 5:

$\rightarrow$ the: $\frac{2}{6} = 0.3333$

$\rightarrow$ mat : $\frac{1}{6} = 0.1667$

$\rightarrow$ is : $\frac{1}{6} = 0.1667$

$\rightarrow$ on : $\frac{1}{6} = 0.1667$

$\rightarrow$ floor : $\frac{1}{6} = 0.1667$

Step 2: Calculate Inverse Document Frequency:

$$IDF(t) = \log\left(\frac{\text{Total number of documents}}{\text{Number of documents containing } t}\right)$$

$\rightarrow$ **the:** $\log\left(\frac{5}{5}\right) = 0$

$\rightarrow$ cat: $\log\left(\frac{5}{3}\right) = 0.2218$

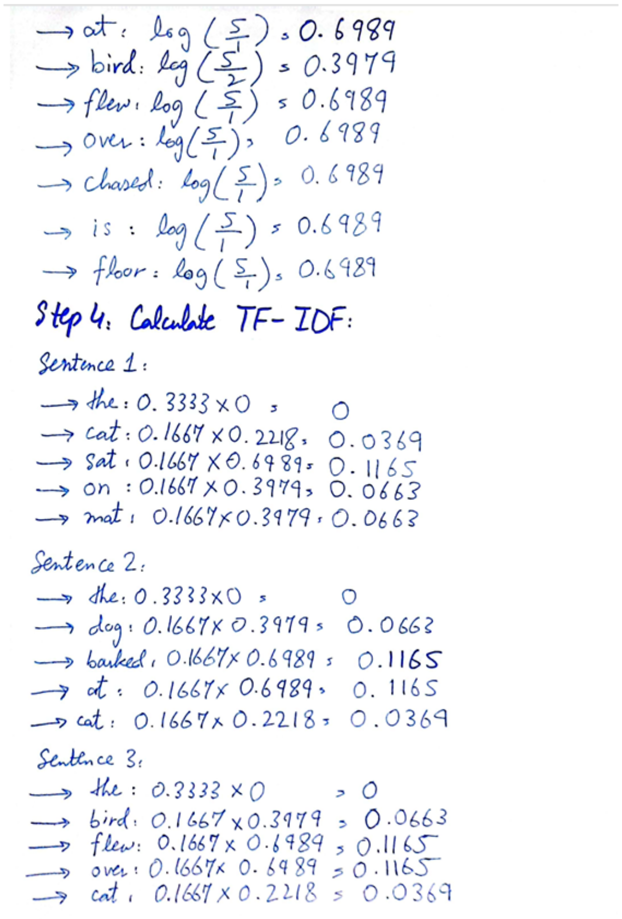
$\rightarrow$ sat: $\log\left(\frac{5}{1}\right) = 0.6989$

$\rightarrow$ on: $\log\left(\frac{5}{2}\right) = 0.3979$

$\rightarrow$ mat: $\log\left(\frac{5}{2}\right) = 0.3979$

$\rightarrow$ dog: $\log\left(\frac{5}{2}\right) = 0.3979$

$\rightarrow$ barked: $\log\left(\frac{5}{1}\right) = 0.6989$



→ at : $\log\left(\frac{5}{1}\right) = 0.6989$

→ bird : $\log\left(\frac{5}{2}\right) = 0.3979$

→ flew : $\log\left(\frac{5}{1}\right) = 0.6989$

→ over : $\log\left(\frac{5}{1}\right) = 0.6989$

→ chased : $\log\left(\frac{5}{1}\right) = 0.6989$

→ is : $\log\left(\frac{5}{1}\right) = 0.6989$

→ floor : $\log\left(\frac{5}{1}\right) = 0.6989$

Step 4: Calculate TF-IDF:

Sentence 1:

→ the : $0.3333 \times 0 = 0$

→ cat : $0.1667 \times 0.2218 = 0.0369$

→ sat : $0.1667 \times 0.6989 = 0.1165$

→ on : $0.1667 \times 0.3979 = 0.0663$

→ mat : $0.1667 \times 0.3979 = 0.0663$

Sentence 2:

→ the : $0.3333 \times 0 = 0$

→ dog : $0.1667 \times 0.3979 = 0.0663$

→ barked : $0.1667 \times 0.6989 = 0.1165$

→ at : $0.1667 \times 0.6989 = 0.1165$

→ cat : $0.1667 \times 0.2218 = 0.0369$

Sentence 3:

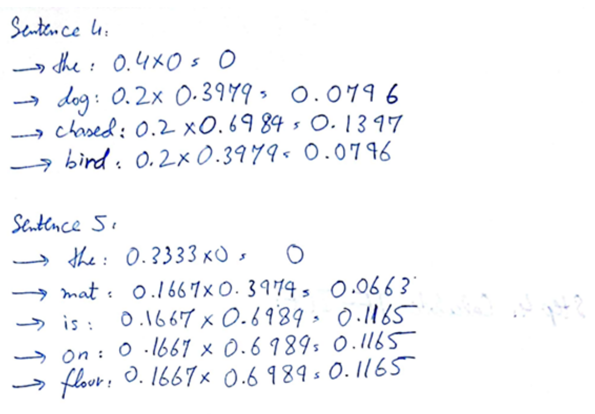
→ the : $0.3333 \times 0 = 0$

→ bird : $0.1667 \times 0.3979 = 0.0663$

→ flew : $0.1667 \times 0.6989 = 0.1165$

→ over : $0.1667 \times 0.6989 = 0.1165$

→ cat : $0.1667 \times 0.2218 = 0.0369$



Sentence 4:

→ the : $0.4 \times 0 = 0$

→ dog : $0.2 \times 0.3979 = 0.0796$

→ chased : $0.2 \times 0.6984 = 0.1397$

→ bird : $0.2 \times 0.3979 = 0.0796$

Sentence 5:

→ the : $0.3333 \times 0 = 0$

→ mat : $0.1667 \times 0.3979 = 0.0663$

→ is : $0.1667 \times 0.6989 = 0.1165$

→ on : $0.1667 \times 0.6989 = 0.1165$

→ floor : $0.1667 \times 0.6989 = 0.1165$