<table>
<tr><td colspan="2">Practical Assignment No. 5</td></tr>
<tr><td>Title:</td><td>Text preprocessing</td></tr>
<tr><td>Problem Statement:</td><td>Build sentiment analysis model using techniques like Bag-of Words, TF-IDF, or word embedding (Word2Vec, GloVe).</td></tr>
<tr><td>Objective:</td><td>To develop applications for text processing using natural language processing techniques.</td></tr>
<tr><td>Outcome:</td><td>CO606.2: Develop text processing application using natural language processing techniques.</td></tr>
<tr><td>Software or Hardware Requirements:</td><td>Anaconda/Java/GCC</td></tr>
<tr><td>Theory:</td><td>Sentiment Analysis:

Sentiment Analysis (also known as Opinion Mining) is a natural language processing (NLP) technique used to determine the emotional tone behind a body of text.
It helps in identifying whether the expressed opinion in a text is positive, negative, or neutral.

It is widely used in:
Social media monitoring
Product and movie reviews
Customer feedback systems
Market analysis and brand monitoring
The process involves:
Text preprocessing
Feature extraction using Bag-of-Words, TF-IDF, or word embeddings
Model training and classification using machine learning algorithms like Naïve Bayes, Logistic Regression, or SVM</td></tr>
</table>

Steps in Sentiment Analysis

1. Data Collection:
   Text data is collected from sources such as Twitter, movie reviews, or product feedback.

2. Text Preprocessing:
   Cleaning text by removing special characters, punctuation, stop-words, and performing tokenization, stemming, and lemmatization.

3. Feature Extraction:
   Converting text into numerical form using one of the following methods:

   - Bag-of-Words

   - TF-IDF

   - Word Embedding (Word2Vec / GloVe)

4. Model Building:
   Training a machine learning classifier on the vectorized data.

5. Performance Evaluation:
   Evaluating the model using metrics such as accuracy, precision, recall, and F1-score.

Feature Extraction Techniques

(a) Bag-of-Words (BoW) Model

Definition:

Bag-of-Words represents a document as a collection (or "bag") of individual words, disregarding grammar and word order but keeping word frequency.

Process:

1. Tokenize the text into words

2. Build a vocabulary of all unique words

3. Represent each document as a vector showing the frequency of each word

Example:

Sentence	Bag-of-Words Representation
"I love Python"	[0, 1, 1, 0]
"Python is great"	[0, 0, 1, 1]

Vocabulary: ['I', 'love', 'Python', 'great']

Advantages:

- Simple and easy to implement
- Works well for small and medium-sized datasets

Disadvantages:

- Ignores context and word order
- Large feature space with sparse vectors

Python Example:

```
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer()
X = vectorizer.fit_transform(corpus)
```

(b) TF-IDF (Term Frequency - Inverse Document Frequency)

Definition:

TF-IDF improves upon BoW by reducing the importance of frequently occurring words (like "the", "is") and giving higher weight to rarer, more informative words.

Formula:

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

Where,

- TF(t, d) = Number of times term *t* appears in document *d*

	• IDF(t) = log (Total number of documents / Number of documents containing *t*)
	
	Advantages:
	• Captures importance of words
	• Reduces impact of common terms
	
	Disadvantages:
	• Still ignores word order and context
	• Produces sparse matrices for large datasets
	
	Python Example:
	`from sklearn.feature_extraction.text import TfidfVectorizer`
	`vectorizer = TfidfVectorizer()`
	`X = vectorizer.fit_transform(corpus)`
	
	(c) Word Embeddings
	
	Definition:
	Word Embedding is a dense vector representation of words that captures their semantic relationships and context.
	Unlike BoW or TF-IDF, it understands that similar words have similar meanings.
	
	Common Embedding Techniques:
	1. Word2Vec
	◦ Developed by Google.
	◦ Based on Neural Networks using either CBOW (Continuous Bag of Words) or Skip-Gram model.
	◦ Captures contextual relationships between words.
	
	Example:
	The model learns that *"king"* – *"man"* + *"woman"* ≈ *"queen"*.

Python Example:

```
from gensim.models import Word2Vec
model = Word2Vec(sentences, vector_size=100, window=5, min_count=1, sg=0)
```

2. **GloVe (Global Vectors for Word Representation)**
   - Developed by Stanford NLP Group.
   - Combines the benefits of global matrix factorization and local context window methods.
   - Learns word meanings based on co-occurrence statistics.

Mathematical Idea:
Words appearing together frequently have similar vector representations.

Example:

```
from gensim.models import KeyedVectors
model = KeyedVectors.load_word2vec_format('glove.6B.100d.txt', binary=False)
```

Advantages:

- Captures semantic meaning and context
- Words with similar meanings have closer vector representations
- Suitable for deep learning models

Disadvantages:

- Requires large data and training time
- More complex compared to BoW or TF-IDF

Model Building

	Once text is converted into numerical form using one of the above techniques, various machine learning algorithms can be applied: ● Naïve Bayes Classifier ● Logistic Regression ● Support Vector Machine (SVM) ● Random Forest ● Deep Learning models (e.g., LSTM, CNN) The trained model predicts sentiment labels like Positive, Negative, or Neutral. **Evaluation Metrics** Performance of the sentiment analysis model is evaluated using: ● Accuracy = (TP + TN) / (TP + TN + FP + FN) ● Precision = TP / (TP + FP) ● Recall = TP / (TP + FN) ● F1-Score = 2 × (Precision × Recall) / (Precision + Recall) ● Confusion Matrix to visualize performance **Applications** ● Product review and movie review analysis ● Social media sentiment monitoring ● Customer service and chatbot systems ● Financial market prediction based on news sentiment
Input/Datasets/Test Cases:	Dataset- Name of the Dataset: Description of the Dataset: Dataset Characteristics:

	Subject Area: Associated Tasks: Feature Type: # Instances: # Features:
Results:	Execute code to use any one standard dataset and any NLP technique for sentiment analysis. Take a print of this code with output for submission as part of results.
Analysis and conclusion:	Write your own analysis of output and conclusion(Minimum 1 statement of Analysis, Minimum 1 Statement Conclusion)
References:	Reference /Links(min Any 2, include dataset ref.). Write references in IEEE format.