Practical-7

Objective: To perform document preprocessing techniques such as Tokenization, POS Tagging, Stop Words Removal, Stemming, and Lemmatization on a sample document. Additionally, create a document representation using Term Frequency (TF) and Inverse Document Frequency (IDF).

Experiment 1: Document Preprocessing

Requirements:

- Python programming environment
- Libraries: nltk, spacy, sklearn

Steps:

1. **Load Sample Document**
    - Provide a sample document as input.
2. **Tokenization**
    - Split the document into individual words or sentences.
    - Use nltk.word_tokenize() or spacy tokenizer.
3. **POS Tagging**
    - Assign Part-of-Speech (POS) tags to words using nltk.pos_tag() or spacy.
4. **Stop Words Removal**
    - Remove commonly used words that do not contribute to meaning.
    - Use nltk.corpus.stopwords or spacy.
5. **Stemming**
    - Reduce words to their root forms using PorterStemmer or LancasterStemmer.
6. **Lemmatization**
    - Convert words to their base form using WordNetLemmatizer or spacy lemmatizer.

TF-IDF Calculation

Requirements:

- Python programming environment
- Libraries: sklearn

Steps:

1. **Load Preprocessed Document**
2. **Compute Term Frequency (TF)**
    - Count occurrences of each word in the document.
3. **Compute Inverse Document Frequency (IDF)**
    - Calculate IDF using sklearn.feature_extraction.text.TfidfVectorizer.
4. **Compute TF-IDF Representation**
    - Multiply TF and IDF values to get TF-IDF scores.
5. **Display TF-IDF Matrix**
    - Show the TF-IDF scores for words in the document.

Expected Outcome:

- A preprocessed document with tokens, POS tags, and cleaned words.
- Computed TF-IDF values representing document importance.

Conclusion:

- Successfully preprocessed text data using NLP techniques.
- Represented text as a numerical TF-IDF matrix.