Artificial Intelligence and Data Science Department
Natural Language Processing / Odd Sem 2023-24 / Experiment 2

Name : Yash Sarang	Class/Roll No. : D16AD / 47	Grade :

Title of Experiment :
Tokenization and Filtration & Script Validation text preprocessing techniques for any given text:

Problem Statement :
Design a comprehensive text preprocessing pipeline that incorporates tokenization, filtration, and script validation techniques to enhance the quality and usability of given text data. The goal is to prepare the text for downstream NLP tasks by effectively handling issues related to language scripts, removing irrelevant information, and converting the text into a suitable format for analysis.

Description / Theory :
Text preprocessing is a crucial step in NLP that involves preparing raw text data for analysis by removing noise, irrelevant information, and standardizing the format.
The techniques we'll use are:

1. Tokenization: Breaking down the text into individual words or tokens. This simplifies the data and allows for further analysis. Techniques include word-based tokenization and subword tokenization (e.g., Byte-Pair Encoding).

2. Filtration: Removing stopwords (commonly occurring words with little informational value), punctuation, and other non-essential elements from the text.

3. Script Validation: Ensuring that the text is in the desired script or language. This can involve detecting and handling multilingual text or ensuring that text is in a specific script (e.g., Latin characters for English).

Tokenization
Tokenization is a fundamental text preprocessing technique that involves breaking down a given text into smaller units called tokens. These tokens can be words, subwords, or characters, depending on the level of granularity required. Tokenization serves to simplify the text data, making it more manageable for subsequent NLP tasks. For instance, the sentence "I love to read books" would be tokenized into individual words: ["I", "love", "to", "read", "books"]. Tokenization aids in feature extraction, text analysis, and model training by providing a structured representation of the text.

Filtration & Script Validation
Filtration encompasses various sub-steps aimed at refining the text further. This includes removing stopwords (common words like "and," "the," "is" with little informational value), punctuation, and special characters. Filtration also involves lowercasing all tokens to standardize the text. Script validation is the process of ensuring that the text is in the desired script or language. In a multilingual context, script validation helps identify and manage mixed-script text. For instance, when processing a text containing both English and non-Latin characters, script validation ensures that the text adheres to

the desired script (e.g., Latin) and handles the non-conforming characters appropriately. These techniques collectively cleanse the text from irrelevant elements, improving the quality and consistency of data for subsequent NLP analysis, such as sentiment analysis or topic modeling.

Flowchart :

Text preprocessing steps are:

1. Tokenization
2. Lower casing
3. Stop words removal
4. Stemming
5. Lemmatization

Program:

```
import string
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
import nltk
nltk.download('punkt')
nltk.download('stopwords')

def tokenize_text(text):
    tokens = word_tokenize(text)  # Tokenize into words
    return tokens



def filter_tokens(tokens):
    filtered_tokens = []
    stop_words = set(stopwords.words("english"))  # Load English stopwords

    for token in tokens:
        token = token.lower()  # Convert token to lowercase
        if token not in stop_words and token not in string.punctuation:
            filtered_tokens.append(token)

    return filtered_tokens

# Example text
text = "I love to read books. मुझे पढ़ना पसंद है."
tokens = tokenize_text(text)
```

```
filtered_tokens = filter_tokens(tokens)


print("Original Tokens:", tokens)
print("Filtered Tokens:", filtered_tokens)
```

Output:

```
Original Tokens: ['I', 'love', 'to', 'read', 'books', '.', 'मुझे', 'पढ़ना', 'पसंद', 'है', '.']
Filtered Tokens: ['love', 'read', 'books', 'मुझे', 'पढ़ना', 'पसंद', 'है']
```

Results and Discussions :

Text preprocessing plays a pivotal role in Natural Language Processing (NLP) by refining raw textual data to ensure it's structured, coherent, and amenable to effective analysis. Through techniques like tokenization, stemming, stopword removal, and spell correction, preprocessing enhances the quality of data, reduces noise, and standardizes formats, thereby facilitating accurate feature extraction and model training. By mitigating the challenges of varying text formats, languages, and noise, preprocessing enables NLP models to uncover meaningful patterns, improve classification and sentiment analysis, and yield more precise insights from unstructured text data.

text preprocessing insights are as follows::

- Effectiveness: Evaluate how each technique contributes to cleaning and enhancing the text data. Compare the input and preprocessed text to highlight improvements.
- Tokenization Variants: Compare the results and challenges of different tokenization methods, such as word-based vs. subword-based tokenization.
- Filtration Impact: Discuss the impact of stopwords and punctuation removal on the text's readability and analysis potential. Highlight any challenges in dealing with specific cases.
- Script Validation Challenges: Address the difficulties and strategies encountered in script validation, especially when dealing with multilingual or mixed-script text.
- Usability: Reflect on the usability of the preprocessed text for various NLP tasks. How does this preprocessing enhance the accuracy and efficiency of subsequent analysis?
- Scalability: Consider the scalability of the preprocessing pipeline for larger datasets and suggest ways to optimize its performance.

******