Plagiarism Detection System using NLP

Overview:

The goal is to compare two documents and identify whether one document is plagiarized from the other. We will use NLP techniques such as text preprocessing (tokenization, stopword removal), vectorization (TF-IDF), and cosine similarity to measure document similarity.

Theory

1. Tokenization:

Tokenization refers to splitting text into smaller units such as words or phrases. This allows us to work with individual components of the text, which are easier to analyze.

2. Stopwords Removal:

Stopwords are common words like "the," "and," "is," etc., that don't carry much meaning and can be ignored in the text analysis process.

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

TF-IDF is a statistical measure used to evaluate how important a word is within a document relative to all other documents in a corpus. It combines two components:

- **TF (Term Frequency)**: Measures how frequently a word appears in a document.
- **IDF (Inverse Document Frequency)**: Measures how important the word is by considering how often it appears across all documents. The higher the TF-IDF value, the more important the word is for that document.

4. Cosine Similarity:

Cosine similarity measures the cosine of the angle between two vectors. In our case, these vectors represent the document's TF-IDF values. A cosine similarity close to 1 indicates high similarity, and a value close to 0 indicates low similarity.

Procedure:

1. **Preprocess the Text:**
   - Convert text to lowercase to avoid case-sensitive issues.
   - Remove special characters and numbers that are not useful for similarity comparison.
   - Remove stopwords from the text to focus on meaningful words.
   - Tokenize the text into words.
2. **Vectorize the Text:**
   - Use the TF-IDF method to convert the text into a numerical format (vector).
   - Calculate the TF-IDF score for each word in the document.
3. **Measure Similarity:**
   - Calculate the cosine similarity between two documents.

- The higher the cosine similarity score, the more likely one document is plagiarized from the other.

4. **Plagiarism Detection:**
   - Set a threshold for similarity (e.g., 0.7). If the cosine similarity between two documents exceeds this threshold, it is considered plagiarized.

Code Implementation

```
import re

import nltk

from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.metrics.pairwise import cosine_similarity

from nltk.corpus import stopwords

# Download the stopwords from NLTK

nltk.download('stopwords')

# Preprocessing the text: Removing special characters, stopwords, and lowercasing

def preprocess_text(text):

    # Convert to lowercase

    text = text.lower()

    # Remove special characters and numbers

    text = re.sub(r'[^a-zA-Z\s]', '', text)

    # Tokenize and remove stopwords

    stop_words = set(stopwords.words('english'))

    words = text.split()

    filtered_words = [word for word in words if word not in stop_words]

    return ' '.join(filtered_words)

# Sample documents

doc1 = """

Plagiarism is the act of taking someone else's work or ideas and presenting them as your own.

It can occur in many forms, such as copying text, stealing ideas, or even paraphrasing without giving credit.
```

```
"""

doc2 = """

Plagiarism is the act of using someone else's intellectual property without permission or acknowledgment.

It can happen through copying text, stealing ideas, or paraphrasing the work without giving credit.

"""

# Preprocess the documents

doc1_cleaned = preprocess_text(doc1)

doc2_cleaned = preprocess_text(doc2)

# Vectorize the cleaned text using TF-IDF

vectorizer = TfidfVectorizer()

tfidf_matrix = vectorizer.fit_transform([doc1_cleaned, doc2_cleaned])

# Calculate cosine similarity between the two documents

cosine_sim = cosine_similarity(tfidf_matrix[0:1], tfidf_matrix[1:2])

# Display the similarity score

print(f"Cosine Similarity Score: {cosine_sim[0][0]:.4f}")

# Set a threshold for plagiarism detection (e.g., 0.7)

threshold = 0.7

if cosine_sim[0][0] >= threshold:

   print("Plagiarism Detected!")

else:

   print("No Plagiarism Detected.")
```

<u>Output</u> :

```
Cosine Similarity Score: 0.5386
No Plagiarism Detected.
```

Explanation of the Code:

1. **Preprocessing:**
   - **Lowercasing**: The text is converted to lowercase to eliminate case-sensitivity.

   - **Removing special characters**: We use regular expressions to remove any characters that are not letters or spaces, ensuring that punctuation and numbers don't interfere with the text analysis.
   - **Stopwords Removal**: We use the `stopwords` corpus from the NLTK library to remove common English stopwords from the text.
   - **Tokenization**: The text is split into words, and we filter out stopwords to focus on meaningful words.
2. **Vectorization (TF-IDF)**:
   - We use the `TfidfVectorizer` from the `sklearn` library to convert the cleaned text into a matrix of TF-IDF features. Each document is represented as a vector of TF-IDF values for each word.
3. **Cosine Similarity Calculation**:
   - The `cosine_similarity` function computes the cosine similarity between the two vectors (representing the two documents). The value ranges from 0 (no similarity) to 1 (high similarity).
4. **Plagiarism Detection**:
   - A threshold of `0.7` is set. If the cosine similarity score is higher than this threshold, we consider the documents to be similar enough to be considered plagiarized.

Additional Enhancements:

1. **Multiple Document Comparison**:
   - Extend the system to compare a single document against a corpus of documents (e.g., a database of academic papers or articles).
2. **More Advanced NLP Preprocessing**:
   - Use stemming or lemmatization to reduce words to their root forms (e.g., "running" → "run").
3. **Enhanced Similarity Measures**:
   - Experiment with other similarity measures like Jaccard similarity or Jaro-Winkler distance for more accurate plagiarism detection.

Conclusion:

This mini project demonstrates how to build a basic plagiarism detection system using NLP techniques. By utilizing text preprocessing, TF-IDF vectorization, and cosine similarity, we can compare the similarity between two documents to detect potential plagiarism. This approach can be extended to handle larger datasets or more complex plagiarism detection challenges.