Plagiarism Checker

This Python script is designed to check for potential plagiarism in a text file by comparing it with content found online using the Google Custom Search API.

Modules Used

1. requests: Handles HTTP requests to interact with web APIs and fetch webpage content.
2. BeautifulSoup (from bs4): Parses HTML content to extract text from specific HTML tags.
3. nltk (Natural Language Toolkit): Processes and analyzes text data, including tokenization, stopword removal, stemming, part-of-speech tagging, and synonym finding.

Installation and Setup

Step 1: Install Required Libraries

Make sure to install the necessary libraries before running the script:

```
pip install requests beautifulsoup4 nltk
```

Additionally, download the necessary NLTK resources:

```
import nltk
```

```
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('averaged_perceptron_tagger')
nltk.download('wordnet')
```

Step 2: Google Custom Search API Setup

You need to define your API key and Custom Search Engine (CSE) ID:

```
API_KEY = 'YOUR_API_KEY'
CSE_ID = 'YOUR_CSE_ID'
```

Replace 'YOUR_API_KEY' and 'YOUR_CSE_ID' with your actual credentials from Google Cloud.

Script Breakdown

Step 3: Function to Search Google

This function sends a query to the Google Custom Search API:

```
def google_search(query, num_results=10):
    url = f"https://www.googleapis.com/customsearch/v1?q={query}&key={API_KEY}&cx={CSE_ID}&num={num_results}"
    response = requests.get(url)
    if response.status_code == 200:
        return response.json()
    else:
```

```
        print("Error:", response.status_code, response.text)
        return None
```

Step 4: Function to Extract Text from a URL

This function extracts text content from a webpage:

```
def extract_text_from_url(url):
    try:
        response = requests.get(url)
        soup = BeautifulSoup(response.text, 'html.parser')
        paragraphs = soup.find_all('p')
        text = ' '.join([para.get_text() for para in paragraphs])
        return text
    except Exception as e:
        print(f"Error extracting text from {url}: {e}")
        return ""
```

Step 5: Text Preprocessing Function

This function preprocesses text by tokenizing, removing stopwords, stemming, and extracting nouns:

```
def preprocess_text(text):
    sentences = sent_tokenize(text)
    stop_words = set(stopwords.words('english'))
    porter = PorterStemmer()
    concepts = []
```

```
    for sentence in sentences:
        words = word_tokenize(sentence)
        words = [word for word in words if word.isalnum() and word.lower() not in stop_words]
        words = [porter.stem(word) for word in words]
        nouns = [word for (word, pos) in pos_tag(words) if pos.startswith('N')]
        for noun in nouns:
            concepts.append(noun)
            synsets = wordnet.synsets(noun)
            if synsets:
                concepts.append(synsets[0].lemmas()[0].name())
    return concepts
```

Step 6: Function to Check Plagiarism

This function combines all the steps to check for plagiarism in a text file:

```
def check_plagiarism(file_path):
    with open(file_path, 'r', encoding='utf-8') as file:
        input_text = file.read()
    results = google_search(input_text)
    if results is None or 'items' not in results:
        print("No search results found or there was an issue with the API request.")
        return
    urls = [item['link'] for item in results['items']]
    retrieved_texts = []
    for url in urls:
        raw_text = extract_text_from_url(url)
```

```
        if raw_text:
            preprocessed_text = preprocess_text(raw_text)
            retrieved_texts.append(preprocessed_text)
    preprocessed_input_text = preprocess_text(input_text)
    input_set = set(preprocessed_input_text)
    plagiarism_score = 0
    for text in retrieved_texts:
        intersection = input_set.intersection(set(text))
        score = len(intersection) / len(input_set) * 100
        plagiarism_score = max(plagiarism_score, score)
    print(f"Plagiarism Score: {plagiarism_score:.2f}%")
    if plagiarism_score > 70:
        print("Potential plagiarism detected.")
    else:
        print("No plagiarism detected.")
```

Step 7: Example Usage

Specify the file path and run the plagiarism check:

```
file_path = '/content/sample1.txt'  # Ensure the file path is correct
check_plagiarism(file_path)
```

Execution Summary

1. Install and import necessary libraries.
2. Define your API key and Custom Search Engine ID.

3. Implement the function to search Google using the Custom Search API.

4. Implement the function to extract text from URLs.

5. Implement the text preprocessing function.

6. Implement the plagiarism checking function.

7. Run the plagiarism check with a sample text file.