plagiarism checker;

Step 1: Install and Import Required Libraries

python
```
!pip install requests beautifulsoup4 nltk import nltk
nltk.download('punkt') nltk.download('stopwords')
nltk.download('averaged_perceptron_tagger') nltk.download('wordnet')
import requests from bs4 import BeautifulSoup from nltk.tokenize
import sent_tokenize, word_tokenize from nltk.corpus import stopwords
from nltk.stem import PorterStemmer from nltk import pos_tag from
nltk.corpus import wordnet
```

This part ensures you have the necessary libraries and resources for web scraping and text processing.

Step 2: Google Custom Search API Setup

Define the API key and Custom Search Engine ID:

```
API_KEY = 'AIzaSyB-Li_QVGqqCA5zvn1PjQy7fbEzA8U_ltw'
CSE_ID = '76626311804d2468f'
```

Replace these with your own API key and CSE ID from Google Cloud.

Step 3: Function to Search Google

Create a function to search Google using the Custom Search API:

```
def google_search(query, num_results=10):
    url =
f"https://www.googleapis.com/customsearch/v1?q={query}&key={API_KEY}&c
x={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
```

This function makes a GET request to the Google Custom Search API and returns the JSON response if successful.

Step 4: Function to Extract Text from a URL

Define a function to extract 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 ""
```

This function fetches the content of a webpage and extracts text from `<p>` tags.

Step 5: Text Preprocessing Function

Create a function to preprocess text:

python

```
def preprocess_text(text):

    sentences = sent_tokenize(text)
```

```python
    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
```

This function tokenizes the text, removes stopwords, stems the words, and extracts nouns to find related concepts using WordNet.

Step 6: Function to Check Plagiarism

Combine everything into a function to check for plagiarism:

Python

```python
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.")
```

This function reads the input file, searches for related content online, extracts and preprocesses the text from the results, and calculates a plagiarism score based on the overlap of concepts.

Step 7: Example Usage

Finally, specify the file path and call the plagiarism check function:

python

Copy code

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

check_plagiarism(file_path)
```

Ensure the file path is correct and the file is available.

Execution Summary

1. Install and import necessary libraries.
2. Define 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.

By following these steps, you can check for potential plagiarism in a text file by comparing it with content found online using Google Custom Search API.