Creating a Simple Search Engine Using Python

Introduction

Information retrieval is a crucial task in the age of big data, where users face significant challenges in finding relevant information amidst a vast amount of available data. This project aims to build a simple search engine using artificial intelligence techniques such as **TF-IDF** and **Cosine Similarity** to retrieve relevant articles based on user queries.

Aspects of Artificial Intelligence in the Project

1. **Data Extraction**:
    - We employed **Web Scraping** techniques to extract documents from the internet. This requires the ability to analyze textual content and gather it from multiple sources.
2. **Text Processing**:
    - The textual data is cleaned to remove noise (such as unnecessary symbols and formatting), making it easier for subsequent analysis.
3. **Text Analysis Using TF-IDF**:
    - **TF-IDF (Term Frequency-Inverse Document Frequency)** is a popular method for converting texts into numerical representations.
        - **TF**: Measures how often a word appears in a document.
        - **IDF**: Reflects the importance of the word in a set of documents. If a word is common across many documents, it gets a lower weight.
4. **Calculating Similarity Using Cosine Similarity**:
    - After converting the texts into numerical representations, we use **Cosine Similarity** to calculate how similar the user query is to the documents.
    - The angle between the text vectors is computed, and the smaller the angle, the greater the similarity.

Steps of the Process

1. **Extracting Documents**:
    - We used the `BeautifulSoup` library to scrape content from the web. Users can change the link to collect content from other websites.
2. **Cleaning the Data**:
    - The extracted texts are cleaned to remove unnecessary words and standardize formatting.
3. **Creating a TF-IDF Matrix**:
    - The `scikit-learn` library is utilized to create a TF-IDF matrix, where each document is represented as a set of numerical values.
4. **Converting the Query into a Vector**:
    - When a query is inputted, it is transformed into a numerical representation using the same TF-IDF process.
5. **Calculating Similarity**:
    - Using `Cosine Similarity`, we calculate the similarity between the query and the articles, and then sort the articles based on their similarity.
6. **Displaying Results**:
    - The articles with the highest similarity are displayed, assisting users in accessing the most relevant information.

Mathematics Behind the Project

- **TF-IDF**:

$$\text{TF}(t, d) = \frac{\text{Number of times term } t \text{ appears in document } d}{\text{Total number of words in document } d}$$

$$\text{IDF}(t) = \log\left(\frac{N}{\text{Number of documents containing } t}\right)$$

$$\text{TF-IDF}(t, d) = \text{TF}(t, d) \times \text{IDF}(t)$$

- **Cosine Similarity**:

$$\text{Cosine Similarity}(A, B) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

Conclusion

This project serves as an example of how artificial intelligence techniques can be utilized in information retrieval effectively. By integrating data extraction, text analysis, and similarity calculation, we provide a solution that helps users find information accurately and quickly.