M1: Milestone 1

Search Engine Project

Collaborators

UCI IDs
61929470
43345488
69889948
14002250

Report

From Generated report.txt (From Code)

Number of indexed documents:	1988
Number of unique tokens:	11977
Total size of the index on disk (KB):	81289.0576171875

```
1    Number of indexed documents: 1988
2    Number of unique tokens: 11977
3    Total size of the index on disk (KB): 81289.0576171875
```

```python
import os
import re
from bs4 import BeautifulSoup
import json
from collections import defaultdict  #for create_inverted_index
from nltk.stem import PorterStemmer

inverted_index = defaultdict(list)  #tdefault dict will handle missing keys if necessary

stemmer = PorterStemmer()  #Initialize Porter Stemmer

#Used to load the content from a specified directory
def load_content(directory):
    documents = []  #Empty list to store the documents' data
    for subdirectory in os.listdir(directory):
        subdirectory_path = os.path.join(directory, subdirectory)
        if os.path.isdir(subdirectory_path):
            for fileName in os.listdir(subdirectory_path):  #Loops through the directory
                if fileName.endswith('.json'):
                    filePath = os.path.join(subdirectory_path, fileName)  #Combines directory and fileName to make a full path
                    with open(filePath, 'r', encoding='utf-8') as file:  #Encodes the file in utf-8 and reads the file
                        try:
                            data = json.load(file)
                            #Extracts all the data in the JSON file
                            url = data.get("url")
                            content = data.get("content")
                            encoding = data.get("encoding")
                            #Adds it to the documents list
                            documents.append({"filename": fileName, "url": url, "content": content, "encoding": encoding})
                        except json.JSONDecodeError:
                            print(f"Error decoding JSON in file {fileName}")
    print(f"Loaded {len(documents)} documents")  #Add this line to verify loaded documents
    return documents

#Clean HTML content using BeautifulSoup
def clean_html(raw_html):
    if raw_html:
        soup = BeautifulSoup(raw_html, "html.parser")
        return soup.get_text()
    return ""

#Tokenize and stem text
def tokenize_text_and_stem(text):
    #Clean the text to exclude HTML tags
    clean_text = clean_html(text)

    #Find sequences of alphabet characters that are at least 2 characters long
    raw_tokens = re.findall(r'[a-zA-Z]{2,}', clean_text.lower())

    #Stem tokens using Porter Stemmer
    stemmed_tokens = [stemmer.stem(token) for token in raw_tokens]
    print(f"Tokens: {stemmed_tokens}")  #Add this line to verify tokens
    return stemmed_tokens


#Create the inverted index
def create_inverted_index(documents):

    for document in documents:  #Iterate through documents list
        file_name = document['filename']  #Pull from load_content using type filename and type content
        content = document['content']
        if content:
            print(f"Document: {document['filename']}, Content length: {len(content)}")  #Verify content length

        term_frequency = defaultdict(int)  #Will also handle missing keys

        #Tokenize and stem the content
        tokens = tokenize_text_and_stem(content)
        for token in tokens:  #Iterate through the tokens
            term_frequency[token] += 1  #Increment term frequency of a found token per loop

        #Generate a posting using the term frequency, (Technically incomplete, since this only uses the "tf" in the tf-idf score)
        for token, frequency in term_frequency.items():
            posting = {
                'document': file_name,
                'term_frequency': frequency
            }
            inverted_index[token].append(posting)  #Add posting to the inv index

    return inverted_index

#Save the inverted index to a JSON file
def saveInvertedIndex(file_path):
    with open(file_path, 'w', encoding='utf-8') as file:
        json.dump(inverted_index, file, indent=4)
```

```python
def save_report(file_path, numDocuments, numUniqueTokens, indexSizeKB):
    with open(file_path, 'w', encoding='utf-8') as file:
        file.write(f"Number of indexed documents: {numDocuments}\n")
        file.write(f"Number of unique tokens: {numUniqueTokens}\n")
        file.write(f"Total size of the index on disk (KB): {indexSizeKB}\n")

#Main function
def main():
    directory = "./ANALYST"
    documents = load_content(directory)
    inverted_index = create_inverted_index(documents)
    saveInvertedIndex("inverted_index.json")

    #Print data for the report
    num_documents = len(documents)
    num_unique_tokens = len(inverted_index)
    index_size_kb = os.path.getsize("inverted_index.json") / 1024

    print(f"Number of documents: {num_documents}")
    print(f"Number of unique tokens: {num_unique_tokens}")
    print(f"Index size (KB): {index_size_kb}")
    #Save the report to a text file
    save_report("report.txt", num_documents, num_unique_tokens, index_size_kb)

if __name__ == "__main__":
    main()
```