A Baby Search Engine

Author(s): Hareena Chowdary Polavaram
Date: 06 Oct. 23

A Baby Search Engine

Phase 1: INTRODUCTION

In this report, I will document the development of a simple search engine, inspired by the principles of modern web search engines. The goal of this project was to build a mini search engine capable of indexing webpages, processing user queries, and returning relevant search results

Phase 2: SPECIFICATION

A simple search engine for finding websites that contain specific words. The search engine initially creates an inverted index from a database of webpages. The inverted index connects words to groups of URLs that use those words. The search engine then uses the inverted index to find websites that correlate to the user's query text.

The following are the goals that we should follow in order to write a code:

- Tokenization
- Creating forward and inverted index
- Search engine basics
- Practice developing high-performance solution.

2.1: Assignment Description

The primary objective of this assignment was to develop a search engine with the following functionalities:

- Reads a database file containing webpage URLs and their content.
- Builds an inverted index to map words to webpage URLs.
- Processes user queries, including single terms, compound queries, and optional modifiers.
- Presents search results to the user in a user-friendly manner.

Phase 3: DESIGN

3.1: Data Structures

We used Python dictionaries and sets extensively to efficiently organise and process data. Our primary data structures included:

- A forward index that stores URLs and unique words for each webpage.
- An inverted index for mapping words to webpage URLs

3.2: Key Functions

1. def cleanToken(token):

This function cleans and normalizes tokens found in webpages.

Input: A string token.

Output: A cleaned and processed string token.

Action:

a. Removes all the punctuation from the token.
b. Verifies whether the token has any alphanumeric characters. If not, gives a blank string.
c. Converts the token to lowercase.
d. Returns the token.

2. def buildInvertedIndex(docs):

This function creates an inverted index from the forward index**.**

Input: A dictionary mapping URLs to sets of unique words.

Output: A dictionary mapping words to sets of URLs.

Action:

a. Creates a new index.
b. Iterates over each token and checks if it’s in a new index as a key.
    i. If not, it adds it as a key.
c. Adds url to its respective token.
d. Returns inverted index.

3. def findQueryMatches(index, query):

This function processes user queries and returns relevant search results.

Input: A dictionary mapping URLs to sets of unique words.

Output: A dictionary mapping words to sets of URLs.

Action:

a. Splits the query into tokens.
b. Iterates over the tokens in the query.
    i. If the token is an operator, then it sets the operator variable.
    ii. Otherwise, the token is a search token.
        1. Normalizes the search token.
        2. If the search token is in the inverted index, then it adds it to the result set.
c. Returns the results set.

4. def readDocs(dbfile):

This function reads the database file and populates the forward index.

Input: The path to a database file(string).

Output: A dictionary mapping URLs to sets of unique words.

Action:

a. Reads the data from the text file.
b. Iterates over the data.

        i. If the sentence starts with http, it adds the URL of a new document.
            1. For the existing document, it adds it to the dictionary with key as url and unique tokens list as value.
        ii. If the sentence starts with ‘ [‘ or ends with ‘] ‘ it adds the data of the page.
        iii. Otherwise, it adds the data of the current page.
    c. Adds the data of the last page.
    d. Returns the dictionary of each page.

5. def mySearchEngine(dbfile):

This function serves as the main entry point of the search engine.

Input: The path to a database file (string).

Output: Displays matching URLs for user queries in a command-line interface.

Action:

a. Reads the data from the text file.
b. Creates Inverted Index.
c. Displays the number of unique tokens and indexed pages in the input file.
d. If the user provides input as:
    i. Query - returns the output.
    ii. Enter/Return - Stops the execution.

Phase 4: RISK ANALYSIS

The code is a simple search engine. No specific risks were identified in this project. However, potential issues might include:

- Incorrect data parsing from the input file.
- Performance concerns with large datasets.

Phase 5: VERIFICATION

I performed thorough testing on the code, including unit tests for each function and manual validation using sampleWebsiteData.txt files. All functions satisfied the expected output.

Phase 6: CODING

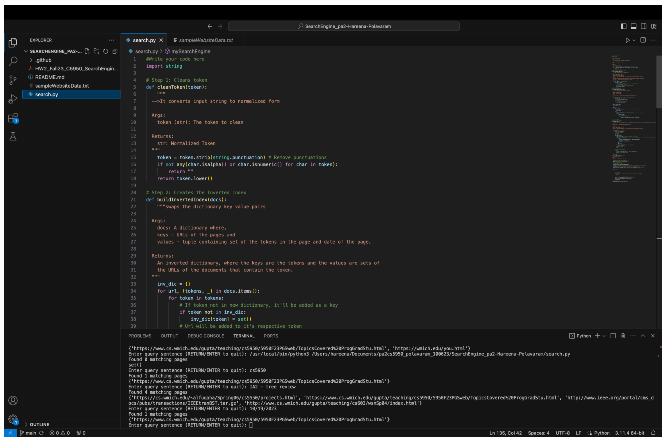
This program was implemented in Python. The main parts of the code are as follows:

- Parsing the input database file.
- Processing tokens and URLs.
- Building the inverted index.
- Handling user queries and displaying results.

The code of this program is included in the zip file.

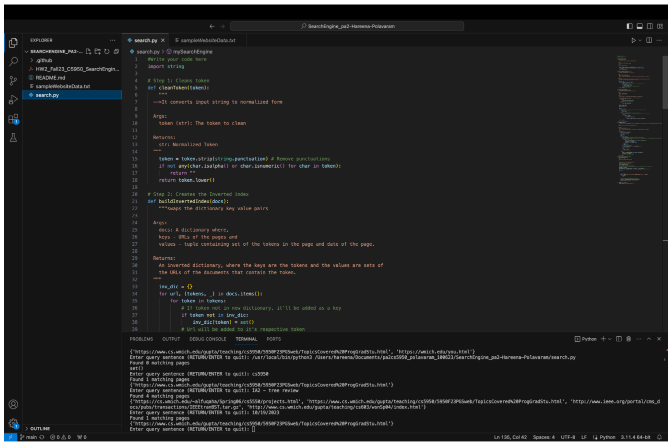
Phase 7: TESTING

7.1 Input & Output Specifications:



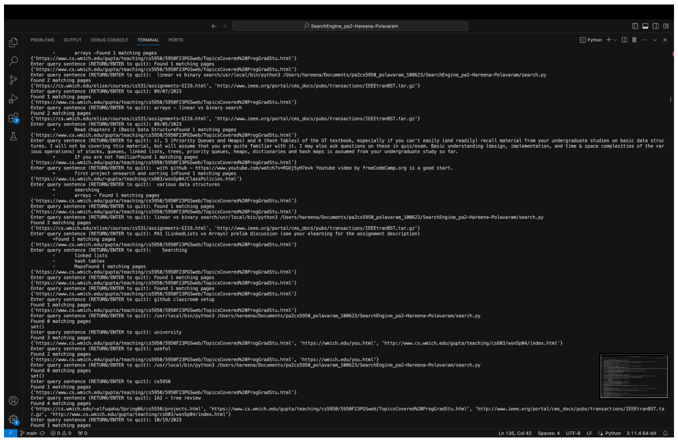
```python
#Write your code here
import string

# Step 1: Cleans token
def cleanToken(token):
    """
  -->It converts input string to normalized form

  Args:
    token (str): The token to clean

  Returns:
    str: Normalized Token
  """
    token = token.strip(string.punctuation) # Remove punctuations
    if not any(char.isalpha() or char.isnumeric() for char in token):
        return ""
    return token.lower()

# Step 2: Creates the Inverted index
def buildInvertedIndex(docs):
    """swaps the dictionary key value pairs

  Args:
    docs: A dictionary where,
    keys - URLs of the pages and
    values - tuple containing set of the tokens in the page and date of the page.

  Returns:
    An inverted dictionary, where the keys are the tokens and the values are sets of
    the URLs of the documents that contain the token.
  """
    inv_dic = {}
    for url, (tokens, _) in docs.items():
        for token in tokens:
            # If token not in new dictionary, it'll be added as a key
            if token not in inv_dic:
                inv_dic[token] = set()
            # Url will be added to it's respective token
```

```
{'https://www.cs.wmich.edu/gupta/teaching/cs5950/5950F23PGSweb/TopicsCovered%20ProgGradStu.html', 'https://wmich.edu/you.html'}
Enter query sentence (RETURN/ENTER to quit): /usr/local/bin/python3 /Users/hareena/Documents/pa2cs5950_polavaram_100623/SearchEngine_pa2-Hareena-Polavaram/search.py
Found 0 matching pages
set()
Enter query sentence (RETURN/ENTER to quit): cs5950
Found 1 matching pages
{'https://www.cs.wmich.edu/gupta/teaching/cs5950/5950F23PGSweb/TopicsCovered%20ProgGradStu.html'}
Enter query sentence (RETURN/ENTER to quit): IA2 - tree review
Found 4 matching pages
{'https://cs.wmich.edu/~alfuqaha/Spring06/cs5550/projects.html', 'https://www.cs.wmich.edu/gupta/teaching/cs5950/5950F23PGSweb/TopicsCovered%20ProgGradStu.html', 'http://www.ieee.org/portal/cms_d
ocs/pubs/transactions/IEEEtranBST.tar.gz', 'http://www.cs.wmich.edu/gupta/teaching/cs603/wsnSp04/index.html'}
Enter query sentence (RETURN/ENTER to quit): 10/19/2023
Found 1 matching pages
{'https://www.cs.wmich.edu/gupta/teaching/cs5950/5950F23PGSweb/TopicsCovered%20ProgGradStu.html'}
Enter query sentence (RETURN/ENTER to quit): 
```



```python
#Write your code here
import string

# Step 1: Cleans token
def cleanToken(token):
    """
  -->It converts input string to normalized form

  Args:
    token (str): The token to clean

  Returns:
    str: Normalized Token
  """
    token = token.strip(string.punctuation) # Remove punctuations
    if not any(char.isalpha() or char.isnumeric() for char in token):
        return ""
    return token.lower()

# Step 2: Creates the Inverted index
def buildInvertedIndex(docs):
    """swaps the dictionary key value pairs

  Args:
    docs: A dictionary where,
    keys - URLs of the pages and
    values - tuple containing set of the tokens in the page and date of the page.

  Returns:
    An inverted dictionary, where the keys are the tokens and the values are sets of
    the URLs of the documents that contain the token.
  """
    inv_dic = {}
    for url, (tokens, _) in docs.items():
        for token in tokens:
            # If token not in new dictionary, it'll be added as a key
            if token not in inv_dic:
                inv_dic[token] = set()
            # Url will be added to it's respective token
```

```
{'https://www.cs.wmich.edu/gupta/teaching/cs5950/5950F23PGSweb/TopicsCovered%20ProgGradStu.html', 'https://wmich.edu/you.html'}
Enter query sentence (RETURN/ENTER to quit): /usr/local/bin/python3 /Users/hareena/Documents/pa2cs5950_polavaram_100623/SearchEngine_pa2-Hareena-Polavaram/search.py
Found 0 matching pages
set()
Enter query sentence (RETURN/ENTER to quit): cs5950
Found 1 matching pages
{'https://www.cs.wmich.edu/gupta/teaching/cs5950/5950F23PGSweb/TopicsCovered%20ProgGradStu.html'}
Enter query sentence (RETURN/ENTER to quit): IA2 - tree review
Found 4 matching pages
{'https://cs.wmich.edu/~alfuqaha/Spring06/cs5550/projects.html', 'https://www.cs.wmich.edu/gupta/teaching/cs5950/5950F23PGSweb/TopicsCovered%20ProgGradStu.html', 'http://www.ieee.org/portal/cms_d
ocs/pubs/transactions/IEEEtranBST.tar.gz', 'http://www.cs.wmich.edu/gupta/teaching/cs603/wsnSp04/index.html'}
Enter query sentence (RETURN/ENTER to quit): 10/19/2023
Found 1 matching pages
{'https://www.cs.wmich.edu/gupta/teaching/cs5950/5950F23PGSweb/TopicsCovered%20ProgGradStu.html'}
Enter query sentence (RETURN/ENTER to quit): 
```



```
        ◦       arrays -Found 1 matching pages
{'https://www.cs.wmich.edu/gupta/teaching/cs5950/5950F23PGSweb/TopicsCovered%20ProgGradStu.html'}
Enter query sentence (RETURN/ENTER to quit): Found 1 matching pages
{'https://www.cs.wmich.edu/gupta/teaching/cs5950/5950F23PGSweb/TopicsCovered%20ProgGradStu.html'}
Enter query sentence (RETURN/ENTER to quit):  linear vs binary search/usr/local/bin/python3 /Users/hareena/Documents/pa2cs5950_polavaram_100623/SearchEngine_pa2-Hareena-Polavaram/search.py
Found 2 matching pages
{'https://cs.wmich.edu/elise/courses/cs531/assignments-SI19.html', 'http://www.ieee.org/portal/cms_docs/pubs/transactions/IEEEtranBST.tar.gz'}
Enter query sentence (RETURN/ENTER to quit): 09/07/2023
Found 1 matching pages
{'https://www.cs.wmich.edu/gupta/teaching/cs5950/5950F23PGSweb/TopicsCovered%20ProgGradStu.html'}
Enter query sentence (RETURN/ENTER to quit): arrays - linear vs binary search
Found 2 matching pages
{'https://cs.wmich.edu/elise/courses/cs531/assignments-SI19.html', 'http://www.ieee.org/portal/cms_docs/pubs/transactions/IEEEtranBST.tar.gz'}
Enter query sentence (RETURN/ENTER to quit): 09/05/2023
        •       Read chapters 2 (Basic Data StructureFound 1 matching pages
{'https://www.cs.wmich.edu/gupta/teaching/cs5950/5950F23PGSweb/TopicsCovered%20ProgGradStu.html'}
Enter query sentence (RETURN/ENTER to quit): s), 5 (Priority Queues and Heaps) and 6 (Hash Tables) of the GT textbook, especially if you can't easily (and readily) recall material from your undergraduate studies on basic data struc
tures. I will not be covering this material, but will assume that you are quite familiar with it. I may also ask questions on these in quiz/exam. Basic understanding (design, implementation, and time & space complexities of the var
ious operations) of stacks, queues, linked lists, trees, priority queues, heaps, dictionaries and hash maps is assumed from your undergraduate study so far.
        •       If you are not familiarFound 1 matching pages
{'https://www.cs.wmich.edu/gupta/teaching/cs5950/5950F23PGSweb/TopicsCovered%20ProgGradStu.html'}
Enter query sentence (RETURN/ENTER to quit):  with github - https://www.youtube.com/watch?v=RGOj5yH7evk Youtube video by freeCodeCamp.org is a good start.
        •       first project onsearch and sorting inFound 1 matching pages
{'https://www.cs.wmich.edu/~gupta/teaching/cs603/wsnSp04/ClassPolicies.html'}
Enter query sentence (RETURN/ENTER to quit):  various data structures
        •       searching
        ◦       arrays - Found 1 matching pages
{'https://www.cs.wmich.edu/gupta/teaching/cs5950/5950F23PGSweb/TopicsCovered%20ProgGradStu.html'}
Enter query sentence (RETURN/ENTER to quit): Found 1 matching pages
{'https://www.cs.wmich.edu/gupta/teaching/cs5950/5950F23PGSweb/TopicsCovered%20ProgGradStu.html'}
Enter query sentence (RETURN/ENTER to quit): linear vs binary search/usr/local/bin/python3 /Users/hareena/Documents/pa2cs5950_polavaram_100623/SearchEngine_pa2-Hareena-Polavaram/search.py
Found 2 matching pages
{'https://cs.wmich.edu/elise/courses/cs531/assignments-SI19.html', 'http://www.ieee.org/portal/cms_docs/pubs/transactions/IEEEtranBST.tar.gz'}
Enter query sentence (RETURN/ENTER to quit): PA1 (LinkedLists vs Arrays) prelim discussion (see your elearning for the assignment description)
        •Found 1 matching pages
{'https://www.cs.wmich.edu/gupta/teaching/cs5950/5950F23PGSweb/TopicsCovered%20ProgGradStu.html'}
Enter query sentence (RETURN/ENTER to quit):    Searching
        ◦       linked lists
        ◦       hash tables
        ◦       MapsFound 1 matching pages
{'https://www.cs.wmich.edu/gupta/teaching/cs5950/5950F23PGSweb/TopicsCovered%20ProgGradStu.html'}
Enter query sentence (RETURN/ENTER to quit): Found 1 matching pages
{'https://www.cs.wmich.edu/gupta/teaching/cs5950/5950F23PGSweb/TopicsCovered%20ProgGradStu.html'}
Enter query sentence (RETURN/ENTER to quit): Found 1 matching pages
{'https://www.cs.wmich.edu/gupta/teaching/cs5950/5950F23PGSweb/TopicsCovered%20ProgGradStu.html'}
Enter query sentence (RETURN/ENTER to quit): github classroom setup
Found 1 matching pages
{'https://www.cs.wmich.edu/gupta/teaching/cs5950/5950F23PGSweb/TopicsCovered%20ProgGradStu.html'}
Enter query sentence (RETURN/ENTER to quit): /usr/local/bin/python3 /Users/hareena/Documents/pa2cs5950_polavaram_100623/SearchEngine_pa2-Hareena-Polavaram/search.py
Found 0 matching pages
set()
Enter query sentence (RETURN/ENTER to quit): university
Found 3 matching pages
{'https://www.cs.wmich.edu/gupta/teaching/cs5950/5950F23PGSweb/TopicsCovered%20ProgGradStu.html', 'https://wmich.edu/you.html', 'http://www.cs.wmich.edu/gupta/teaching/cs603/wsnSp04/index.html'}
Enter query sentence (RETURN/ENTER to quit): useful
Found 2 matching pages
{'https://www.cs.wmich.edu/gupta/teaching/cs5950/5950F23PGSweb/TopicsCovered%20ProgGradStu.html', 'https://wmich.edu/you.html'}
Enter query sentence (RETURN/ENTER to quit): /usr/local/bin/python3 /Users/hareena/Documents/pa2cs5950_polavaram_100623/SearchEngine_pa2-Hareena-Polavaram/search.py
Found 0 matching pages
set()
Enter query sentence (RETURN/ENTER to quit): cs5950
Found 1 matching pages
{'https://www.cs.wmich.edu/gupta/teaching/cs5950/5950F23PGSweb/TopicsCovered%20ProgGradStu.html'}
Enter query sentence (RETURN/ENTER to quit): IA2 - tree review
Found 4 matching pages
{'https://cs.wmich.edu/~alfuqaha/Spring06/cs5550/projects.html', 'https://www.cs.wmich.edu/gupta/teaching/cs5950/5950F23PGSweb/TopicsCovered%20ProgGradStu.html', 'http://www.ieee.org/portal/cms_docs/pubs/transactions/IEEEtranBST.ta
r.gz', 'http://www.cs.wmich.edu/gupta/teaching/cs603/wsnSp04/index.html'}
Enter query sentence (RETURN/ENTER to quit): 10/19/2023
Found 1 matching pages
```

Phase 8: REFINING THE PROGRAM

Based on the test results, we refined the code, addressing any identified issues, optimizing performance, and enhancing error handling.

Phase 9: PRODUCTION

A zip file including the source files and output of the program have been included.

The final implementation was successfully deployed as a Python program named mySearchEngine. This program enables users to perform searches based on web content indexed in the inverted index.

Phase 10: MAINTAINANCE

This application can be futher improved.