Ahmed Allam

900214493

Analysis and Design of Algorithms Lab

Project Report

Dr. Reem Haweel

13 May 2023

Introduction

This project presents a unique and innovative search engine, designed to provide an efficient and user-friendly way to search for information. The search engine is built using advanced algorithms and data structures, and it offers a unique blend of functionality and performance.

One of the key highlights of my search engine is its dual-mode operation. It allows users to perform searches in both a console mode and a browser mode. This dual-mode functionality provides flexibility to the users, enabling them to choose the mode that best suits their needs and comfort.

In the console mode, users can perform searches directly from the command line, which is ideal for quick searches and for users who prefer a more traditional, text-based interface. On the other hand, the browser mode offers a more interactive and visually appealing interface, with a beautifully designed UI that enhances the user experience.

The search engine is built on a robust indexing and ranking system. It uses an efficient indexing algorithm to build a web graph and associate keywords with URLs. The ranking algorithm is based on the PageRank algorithm, which takes into account the structure of the web graph and the importance of each page, providing a more accurate ranking of the URLs.

Furthermore, it also keeps track of the number of impressions and clicks for each URL, which are used to calculate the Click-Through Rate (CTR), providing another dimension to the ranking of the search results.

In the following sections, we will delve deeper into the pseudo-code of the indexing and ranking algorithms, their time and space complexity analysis, the main data structures used, and the design tradeoffs made during the development.

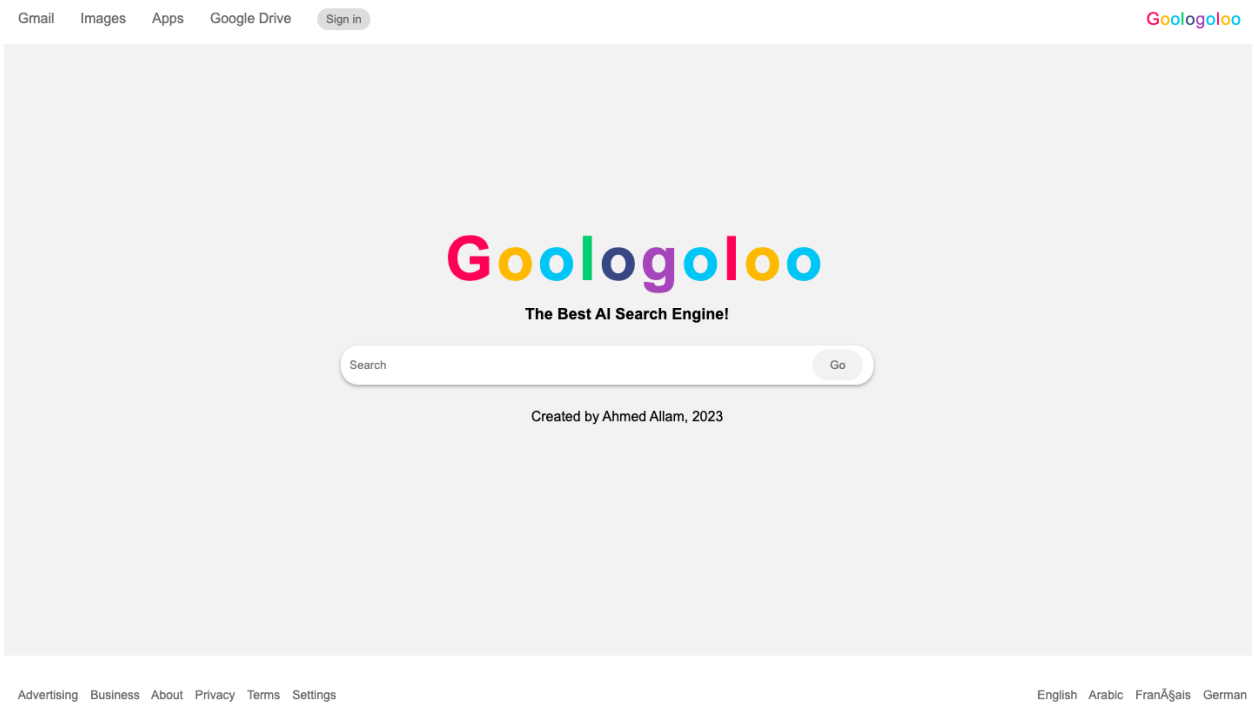

Figure 1. Preview of the browser mode of the search engine. It is designed to be similar to the design of Google, but better :)

Pseudo-code for Indexing and Ranking Algorithms

Indexing Algorithm

```
Function build_webgraph(filename):
    open the file
    for each line in the file:
        split the line into two urls
        trim the urls
        add the second url to the list of outbound links of the first url

Function add_keywords(filename):
    open the file
    for each line in the file:
        split the line into a url and keywords
        trim the url and keywords
        add the url to the list of urls for each keyword

Function add_impressions_count(filename):
    open the file
    for each line in the file:
        split the line into a url and count
        trim the url
        set the impressions count for the url
```

```
Function add_clicks_count(filename):
    open the file
    for each line in the file:
        split the line into a url and count
        trim the url
        set the clicks count for the url
```

Ranking Algorithm

```
Function calculatePageRank(url):
    if url already in the pageRanks:
        return pageRank of url

    set dampingFactor, initialPageRank, minDifference and convergence flag

    set initialPageRank for all pages in the graph

    while PageRank has not converged:
        create a newPageRanks map
        for each page in the graph:
            calculate sum of PageRank of outbound links divided by the number of their own outbound links
            calculate newPageRank
```

```
            update newPageRanks for the page
            if difference between newPageRank and old PageRank is greater than minDifference:
                set convergence flag to false

        replace old PageRanks with newPageRanks

    return PageRank of the url




Function search(query):
    split query into words
    for each word in the query:
        if the word is "AND":
            find urls that contain both the words around "AND"
            create results for those urls and add to the results list
        else if the word is "OR":
            find urls that contain either of the words around "OR"
            create results for those urls and add to the results list
        else if the word starts with a quote:
            find the closing quote
            if found:
                create a phrase from the quoted words
                find urls that contain all the words in the phrase
                create results for those urls and add to the results list
```

```
    else:
        find urls that contain the word
        create results for those urls and add to the results list

sort the results

return results
```

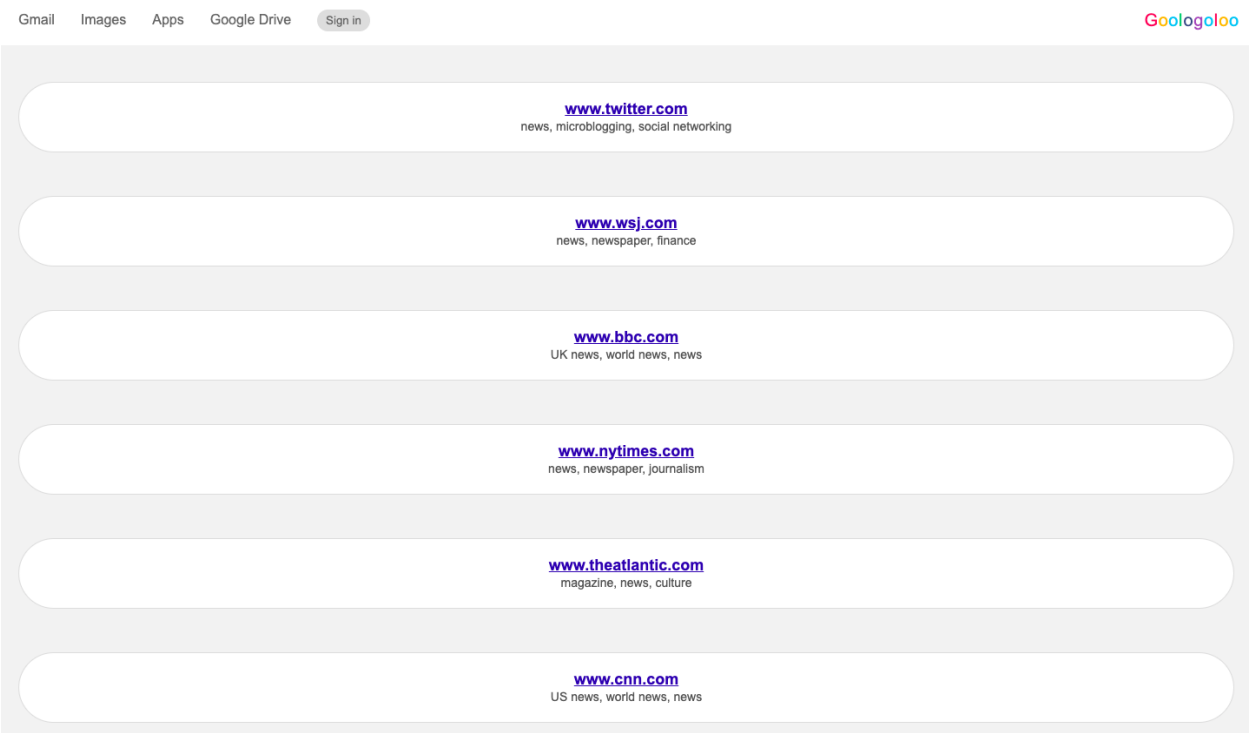

Figure 2. Test case for the engine. When given a query to search for “News”, it iterates over the list of real-world websites - given in webgraph.txt - to search for the desired word, leading to results like Twitter, BBC, CNN, etc. while also giving the ability to click on them and be redirected to the real website, just like what a real professional search engine would do.

Time and Space Complexity Analysis

Indexing Algorithm

The time complexity of the indexing algorithm is O(n), where n is the total number of lines across all input files. This is because each line in each file is processed exactly once.

The space complexity of the indexing algorithm is also O(n), where n is the total number of unique URLs across all input files. This is because each unique URL is stored in the graph, keywords, impressions, and clicks data structures.

Ranking Algorithm

The time complexity of the PageRank calculation is O(m * n), where m is the number of iterations required for the PageRank values to converge, and n is the number of URLs in the graph. This is because in each iteration, the PageRank value for each URL is updated based on the PageRank values of its outbound links.

The space complexity of the ranking algorithm is O(n), where n is the number of URLs in the graph. This is because the PageRank value for each URL is stored in the pageRanks data structure.

Main Data Structures Used

The main data structures used in the algorithm are:

unordered_map<string, vector<string>> *graph: This data structure stores the web graph , where each URL is mapped to a list of its outbound links.

unordered_map<string, vector<string>> *keywords: This data structure stores the keywords associated with each URL. Each keyword is mapped to a list of URLs that contain that keyword.

unordered_map<string, int> *impressions: This data structure stores the number of impressions for each URL. Each URL is mapped to its number of impressions.

unordered_map<string, int> *clicks: This data structure stores the number of clicks for each URL. Each URL is mapped to its number of clicks.

unordered_map<string, float> pageRanks: This data structure stores the PageRank value for each URL. Each URL is mapped to its PageRank value.

Design Tradeoffs and Justifications

Use of unordered_map: The use of unordered_map for storing the graph, keywords, impressions, and clicks allows for efficient O(1) average time complexity for search, insert, and delete operations. The tradeoff is that unordered_map uses more memory than other data structures like map (which uses a balanced binary search tree internally), but the benefit of faster operations justifies this tradeoff.

PageRank calculation: The PageRank algorithm is used to rank the URLs, which takes into account the structure of the web graph and the importance of each page. The tradeoff is that the PageRank calculation can be computationally expensive, especially for large graphs, but it provides a more accurate ranking of the URLs than simpler methods.

Updating of impressions and clicks: The impressions and clicks counts for each URL are updated in the corresponding files whenever they are incremented. The tradeoff is that this requires additional I/O operations, but it ensures that the data is always up-to-date and persistent, even if the program is terminated.

Conclusion

In conclusion, this search engine project stands out due to its unique blend of advanced algorithms, user-friendly interface, and dual-mode operation. The project successfully leverages the power of the PageRank algorithm and efficient data structures to deliver accurate and relevant search results.

The distinctive feature of this project is its dual-mode operation, offering both console and browser modes, catering to the preferences of different users. The browser mode, with its beautifully designed UI, enhances the user experience, making Goologolo not just a search engine, but a visually appealing platform for information retrieval.