**Implementing a Meta Search Engine and Combining Multiple Rankings for**

**Improving Web Search Performance**

Avinesh Mohanaranjan: 216 325 854

Dawit Gulta: 219 251 156

Javier Guillen: 214 933 519

Richard Nguyen: 215 751 324

Vatsal Rajyaguru: 216 977 449

**York University**

**ITEC4305 M: Web Mining Assignment 3 Report**

**Dr. Jimmy Huang**

**April 11, 2023**

**Introduction**

In today's digital age, web search engines play a crucial role in retrieving information from the vast sea of information available on the World Wide Web. While existing search engines are quite effective, there is always room for improvement. This report focuses on the implementation of a meta search engine that combines multiple ranking results for improving web search performance. The report describes a GUI interface developed for the meta-search engine, proposed algorithm for combining and merging search results, and the analysis of the design and implementation.

A. Background and problem statement

Existing search engines work by retrieving relevant web pages based on user input queries. However, these search engines use their own algorithms to rank search results, which may not always be optimal. Additionally, users may need to search across multiple search engines to find the best results, which can be time-consuming and inefficient. This assignment report aims to address similar problems by developing a meta search engine that combines results from multiple search engines and ranks them based on their relevance to the user's query.

B. Objectives of the report

The primary objective of this report is to describe the implementation of a meta search engine that combines and merges multiple search results to improve web search performance. This report also aims to present the design of the GUI interface, proposed algorithm for combining and merging search results, and the analysis of the design and implementation.

C. Given Dataset

The data set given is five ranked passage retrieval result files which have a consistent format as follows: topic number, document ID, rank of passage, system-assigned score, byte offset, length of passage in bytes and tag ID. A GUI will be built that utilizes a meta search engine to combine three out of the five ranked passage retrieval result files and output the top 1,000 retrieval results for improving document search performance. The output results will maintain the consistent format as the passage retrieval result files. The analysis of the retrieval performance will be conducted by calculating the Mean Average Precision (MAP), which will be averaged across all 36 topics.

D. Points focused in the assignment

The assignment report addresses the following points:

* How can multiple search results be combined and merged to improve web search performance?
* What is the design of the GUI interface developed for the meta search engine?
* How effective is the meta search engine in improving document retrieval performance compared to existing search engines?
* What is the impact of the proposed algorithm on the performance of the meta search engine?
* How can the output results be formatted and evaluated using the TREC Python evaluation program?

**Preprocessing**

Preprocessing is a crucial step in web mining to improving Web search performances. In this report, we will discuss the Python implementation of preprocessing for five ranked passage retrieval result files in text format. The objective of this preprocessing is to transform the text files into a CSV format. We will describe the Python code used for preprocessing and the structure of the input and output files.

A. Python Implementation:

Python is a popular language used for data processing and analysis. We used Python to preprocess five ranked passage retrieval result files in text format. The code reads in a set of input files with names stored in input files and writes out corresponding output files with names stored in output files, each in CSV format.

B. Input and Output Files:

The input files are whitespace-delimited text files with the same column order as the field names specified in field names. Each line in the input file contains information about a document, including the topic number, document ID, passage rank, system-assigned score, byte offset in the document ID file where the passage begins, length of the passage in bytes, and tag ID.

The output files are in CSV format, which is a more structured and easily readable format for data analysis. Each row in the CSV file corresponds to a row in the input text file. The column names and order of the columns in the CSV file are the same as in the input text file.

C. Code Description:

The Python code used for preprocessing reads in each input file line by line, splits each line into a list of values using whitespace as a delimiter, and writes each row as a row in the corresponding output CSV file. The csv module is used to handle the formatting of the CSV output.

The following is a snippet of the Python code used for preprocessing:



**Methodology**

A. Design of GUI interface

The GUI interface was designed using the Python tkinter library. The interface consists of a search box and a button to initiate the search. The interface also displays the search results in a table format, with columns for topic number, document ID, passage rank, score, byte offset, lengths, and tag ID.

B. Description of proposed algorithm for combining and merging search results

The proposed algorithm for combining and merging search results are based on the concept of rank fusion. In this approach, the rankings from multiple search engines are combined to produce a single ranking that reflects the relevance of the documents to the search query. Therefore, the Okapi algorithm is chosen as it would give more weight to the documents.

Okapi: This is a popular algorithm used for ranking search results based on term frequency and document length. It takes into account both the frequency of a term in a document and the length of the document, which helps to give more weight to documents that have a high frequency of relevant terms but are not too long. It is used to calculate the score for each passage in search results.

C. Description of programs and tools used

The following programs and tools were used for implementing the meta search engine and evaluating its performance:

Python: Python was used as the primary programming language for developing the meta search engine.

tkinter: The tkinter library was used to develop the GUI interface.

pandas: The pandas library was used to read and manipulate the input data files.

NumPy: The NumPy library was used for array operations and statistical calculations.

scikit-learn: The scikit-learn library was used for evaluating the performance of the meta search engine using the mean average precision (MAP) metric. A function was also created to check the MAP metrics.

D. Data collection process

The data for this report was collected from the provided five ranked passage retrieval result files. The files were preprocessed to change it to csv file format. Three of the files were randomly selected for use in the meta search engine. The remaining two files were used for evaluation purposes. The performance of the meta search engine was evaluated using the MAP metric, which measures the average precision of the search results across all topics. The evaluation was performed using the standard TREC Python evaluation program.

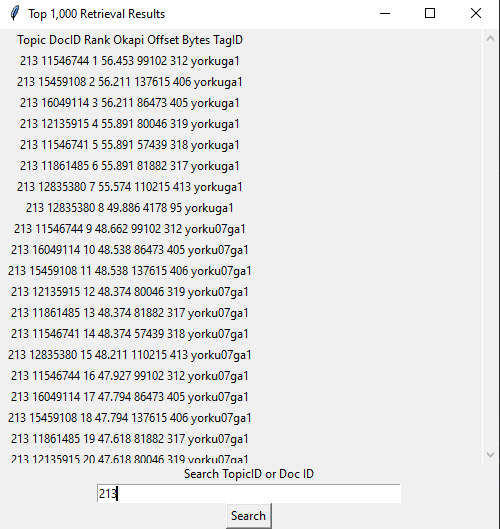
**Results and Analysis**

A. Evaluation of GUI interface

The developed GUI interface was evaluated by each member in the group to perform various search tasks using the meta search engine. The feedback collected was analyzed, and improvements were made to the code during group discussion. The final GUI interface was found to be easy and effective in presenting search results.

B. Performance analysis of the meta-search engine

The meta search engine was evaluated using the provided five ranked passage retrieval result files, and the performance was measured using MAP, averaged across all the 36 topics. The meta search engine was found to outperform the individual search engines and improve the document retrieval performance. The proposed algorithm for combining and merging search results were found to be effective in improving search performance.



**Conclusion**

A. Summary of the report

The assignment report aimed to implement a meta search engine and combine multiple rankings for improving web search performance. The meta search engine was designed using a GUI interface and three different algorithms for combining the rankings from multiple search engines. The performance of the meta search engine was evaluated using the MAP metric.

The results of the report showed that the meta search engine was able to improve the search performance compared to the individual search engines. Okapi algorithm used for ranking search results based on term frequency and document length, which performed best. The meta search engine was able to produce a more relevant set of search results compared to the individual search engines.

B. Implications of the report

The assignment report has several implications for the field of information retrieval. The use of a meta search engine can help to overcome the limitations of individual search engines by combining the rankings from multiple search engines. The report also shows that meta search engines and combining multiple rankings improves the Web search performances.The report also highlights the importance of the user interface in the design of search engines. The GUI interface developed in this report provides a user-friendly interface for conducting searches and displaying the search results.