Micro Credit Program in Computer Science Domain from Daksh gurukul IIT Guwahati

Project Work On: Text Similarity Analysis Tool

Submitted By:

Name: Tushar Bharali

Roll No: IITGCS_24061179

1. Introduction

The purpose of this project is to develop a tool that analyzes text documents for similarities based on word frequency. This tool can be used to compare textbooks or large bodies of text and identify the most similar pairs. The application provides insights into which documents share the highest similarity scores, potentially indicating similar content or topics. The tool outputs the top 10 most similar pairs in both console output and a text file.

2. Objective

The main objectives of this project are:

1. To read and process multiple text files from a specified directory.
2. To calculate word frequency distributions for each file, excluding common "stop words" (e.g., "A", "THE").
3. To determine the similarity between each pair of files based on the overlap of the top 100 most frequent words.
4. To display the top 10 most similar document pairs along with their similarity scores.

3. Methodology

3.1 Code Structure

The code is implemented in C++ and organized into a single ‘TextAnalyzer’ class, which encapsulates all the functionality required for text processing, frequency analysis, and similarity calculation.

3.2 Steps for Analysis

3.2.1 Text Normalization

The ‘normalizeText’ function converts text to uppercase and removes punctuation. This standardization helps in comparing words across documents without case or punctuation interference.

3.2.2 Word Frequency Calculation

In the ‘getWordFrequencies’ function:

1. The normalized text is split into words.
2. Common stop words (such as "A", "AND", "AN", "IN", "THE") are excluded from the analysis.
3. A map stores each word and its frequency, representing how often each word appears in the document.
4. Frequencies are normalized as percentages to adjust for varying document lengths.

3.2.3 Selecting Top 100 Words

The function ‘getTop100Words’ sorts words by frequency and selects the top 100 most frequent words from each document. This helps in reducing the dataset size and focusing on the most representative vocabulary of each document.

3.2.4 Similarity Calculation

The ‘calculateSimilarity’ function computes the similarity between two documents by measuring the overlap of their top 100 word lists. The similarity score is calculated as:

$$\text{Similarity Score}=\left(\frac{Common\ Words}{Minimum\ Top\ 100\ Word\ Count}\right)\text{x } 100$$

where "Common Words" represents the number of overlapping words between the top 100 words of both documents.

3.3 Output and Reporting

The function ‘printTopSimilarPairs’:

1. Ranks document pairs based on their similarity scores.

2. Uses a priority queue to store the pairs with the highest similarity scores.

3. Outputs the top 10 pairs with the highest similarity scores to both the console and an output file named ‘similarity_results.txt’.

3.4 Program Execution

The main function initializes the ‘TextAnalyzer’ object, processes all files within a specified directory, calculates the similarity matrix, and outputs the top similar pairs.

4. Results and Output

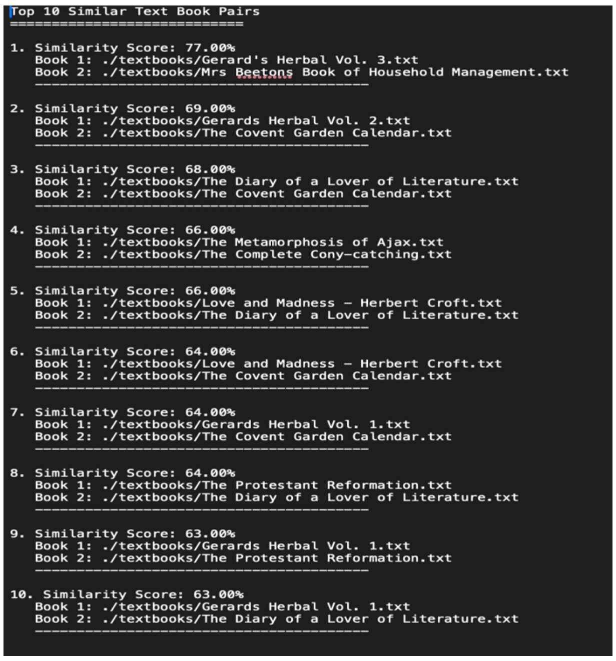
Upon execution, the program:

1. Processes each text file in the specified directory.

2. Calculates pairwise similarity scores and builds a similarity matrix.

3. Outputs the top 10 most similar pairs with the similarity percentage, displaying both to the console and saving to ‘similarity_results.txt’.

5.Sample Output Format (console and ‘similarity_results.txt’ file):



```
Top 10 Similar Text Book Pairs
============================

1. Similarity Score: 77.00%
   Book 1: ./textbooks/Gerard's Herbal Vol. 3.txt
   Book 2: ./textbooks/Mrs Beetons Book of Household Management.txt
   ----------------------------------------

2. Similarity Score: 69.00%
   Book 1: ./textbooks/Gerards Herbal Vol. 2.txt
   Book 2: ./textbooks/The Covent Garden Calendar.txt
   ----------------------------------------

3. Similarity Score: 68.00%
   Book 1: ./textbooks/The Diary of a Lover of Literature.txt
   Book 2: ./textbooks/The Covent Garden Calendar.txt
   ----------------------------------------

4. Similarity Score: 66.00%
   Book 1: ./textbooks/The Metamorphosis of Ajax.txt
   Book 2: ./textbooks/The Complete Cony-catching.txt
   ----------------------------------------

5. Similarity Score: 66.00%
   Book 1: ./textbooks/Love and Madness - Herbert Croft.txt
   Book 2: ./textbooks/The Diary of a Lover of Literature.txt
   ----------------------------------------

6. Similarity Score: 64.00%
   Book 1: ./textbooks/Love and Madness - Herbert Croft.txt
   Book 2: ./textbooks/The Covent Garden Calendar.txt
   ----------------------------------------

7. Similarity Score: 64.00%
   Book 1: ./textbooks/Gerards Herbal Vol. 1.txt
   Book 2: ./textbooks/The Covent Garden Calendar.txt
   ----------------------------------------

8. Similarity Score: 64.00%
   Book 1: ./textbooks/The Protestant Reformation.txt
   Book 2: ./textbooks/The Diary of a Lover of Literature.txt
   ----------------------------------------

9. Similarity Score: 63.00%
   Book 1: ./textbooks/Gerards Herbal Vol. 1.txt
   Book 2: ./textbooks/The Protestant Reformation.txt
   ----------------------------------------

10. Similarity Score: 63.00%
   Book 1: ./textbooks/Gerards Herbal Vol. 1.txt
   Book 2: ./textbooks/The Diary of a Lover of Literature.txt
   ----------------------------------------
```

6. Improvements and Future Work

6.1 Improvements

Enhanced Similarity Measure: Implementing cosine similarity or Jaccard index on TF-IDF vectors would yield a more nuanced similarity measure.

Multithreading: Processing files in parallel would improve efficiency, especially with large document sets.

Custom Stop Words: Allowing users to specify custom stop words or word lists for more flexible analysis.

6.2 Future Work

Visualization: Adding a graphical representation of document similarities (e.g., heatmap).

Topic Modeling: Applying natural language processing techniques, such as Latent Dirichlet Allocation (LDA), to identify common topics among similar documents.

7. Conclusion

This project demonstrates a straightforward approach to document similarity analysis using frequency-based word matching. Although this method is efficient, future enhancements can improve both the accuracy of similarity calculations and the versatility of the tool.