Data Mining

Assignment - 1

Lakshmi Srinidh Pachabotla

Group – 56

11 November 2024

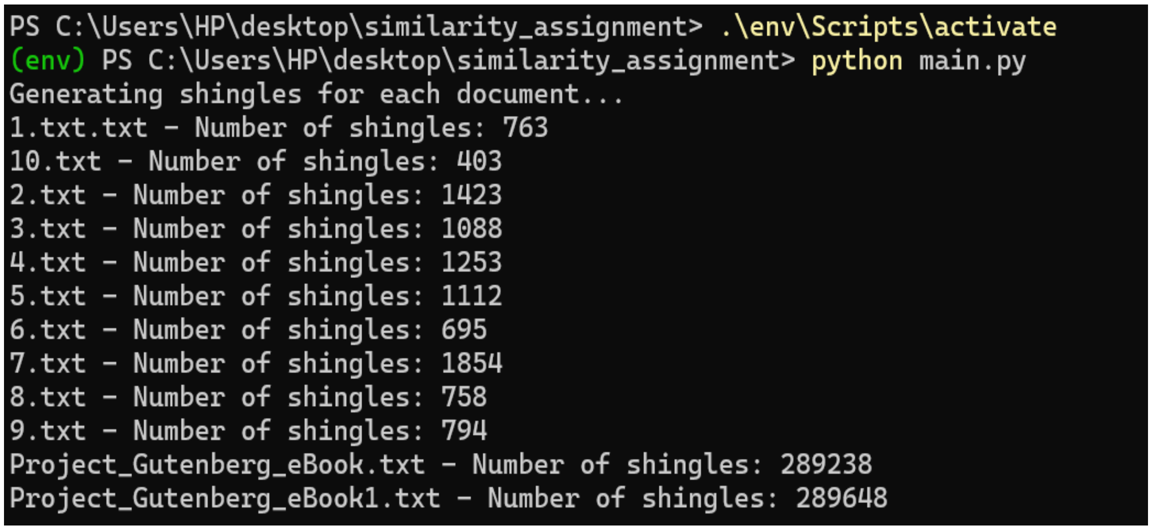
1. Introduction

The purpose of this assignment is to implement a system for identifying textually similar documents using three key techniques: shingling, MinHashing, and Locality-Sensitive Hashing (LSH). Together, these techniques provide a scalable solution for similarity detection, which is valuable for tasks such as duplicate detection, clustering, and content recommendation. This project aims to efficiently detect pairs of documents that are similar based on a threshold, allowing us to handle large text datasets and discover textual similarities across documents. This report outlines the solution, implementation details, results, and reflections on the challenges encountered.

2. Solution

The solution is built on three main techniques: Shingling, MinHashing, and Locality-Sensitive Hashing (LSG). These steps collectively allow us to efficiently determine similar documents in a dataset, particularly for large collections. The output obtained from the solution provides valuable insights into the effectiveness of each technique and demonstrates the functionality of the implemented components. Below is the detailed information provided output for context.

a. **Shingling:** The shingling stage converts each document into a set of unique, fixed-length sequences of characters. This set representation of documents captures their content structure and context, enabling similarity comparisons. Document is divided into singles of length K equal to 10 characters. Shingles are hashed to produce unique individuals, resulting in a set of hashed values that represent each document. The output list the number of shingles generated for each document, indicating variations in document lengths. For example: 1.txt.txt has 763 shingles, while Project_Gutenberg_eBook.txt has a much larger set with 289,238 shingles. These differences reflect the varying lengths and content density of the documents in the dataset. Provides a foundational step that transforms documents into comparable sets. The output shows that each document is now represented by a unique set of shingles, setting the stage for similarity comparisons through Jaccard similarity and min hashing.

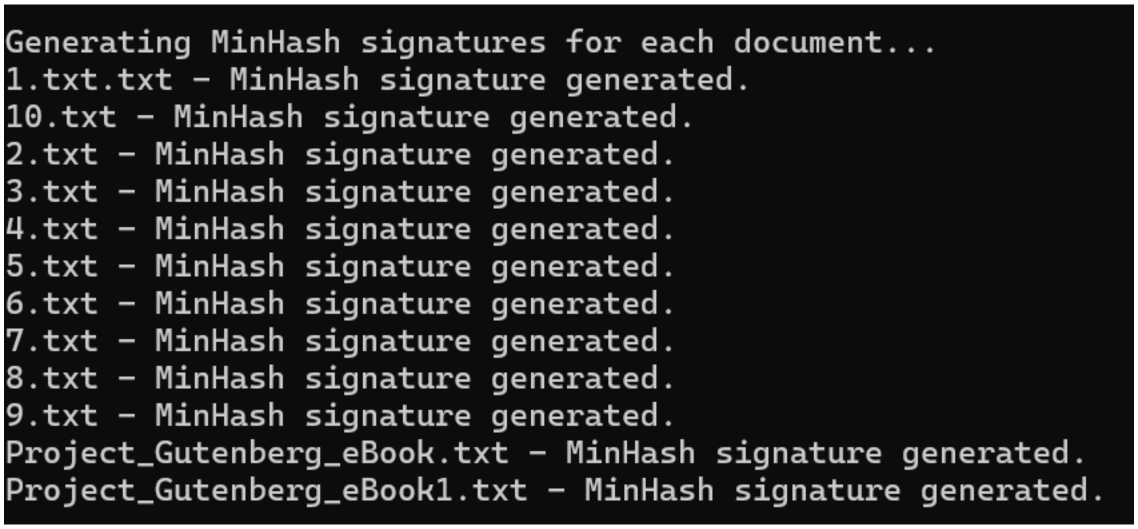


```
PS C:\Users\HP\desktop\similarity_assignment> .\env\Scripts\activate
(env) PS C:\Users\HP\desktop\similarity_assignment> python main.py
Generating shingles for each document...
1.txt.txt - Number of shingles: 763
10.txt - Number of shingles: 403
2.txt - Number of shingles: 1423
3.txt - Number of shingles: 1088
4.txt - Number of shingles: 1253
5.txt - Number of shingles: 1112
6.txt - Number of shingles: 695
7.txt - Number of shingles: 1854
8.txt - Number of shingles: 758
9.txt - Number of shingles: 794
Project_Gutenberg_eBook.txt - Number of shingles: 289238
Project_Gutenberg_eBook1.txt - Number of shingles: 289648
```

b. **Jaccard Similarity:** Jaccard similarity is used to compare the. Single sets between each document pair, providing a direct measure of overlap between the sets. Jaccard similarity between two documents is calculated as the size of intersection divided by the size of the union of their shingle sets. High Jaccard similarity values indicate the documents share significant content. Document pairs show low Jaccard similarity, indicating minimal content overlap. For example, `Jaccard Similarity between 1.txt.txt and 10.txt: 0.0000` and `Jaccard Similarity between 2.txt and 3.txt: 0.0187`. Notably, the pair `Project_Gutenberg_eBook.txt` and

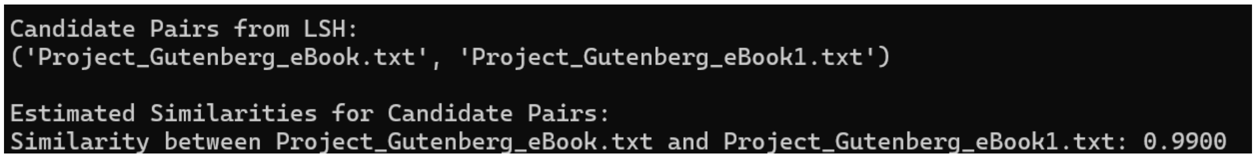
`Project_Gutenberg_eBook1.txt` has a very high Jaccard similarity of 0.9970, suggesting almost identical content. Similarity scores for most document pairs reflect minimal overlap, consistent with the expected diversity in content across documents. The high similarities scores suggest these documents are near duplicates or contain high similar content. This pair is a good candidate. For min hash-based approximation and LSH detection.

c. **MinHashing:** Matching creates a compact fixed size signature for each document, approximating the Jaccard similarity between documents without requiring full set comparison. This approach enhances efficiency, especially for large data sets. For each document, MinHashing generates a signature by applying multiple hash functions to the single set and selecting the minimum value for each function. A MinHash signature with 100 hash values is generated for each document. The output indicates that each document's MinHash signature was successfully generated. Signature based similarity results reveal patterns consistent with Jaccard similarity. For example, the signature similarity between `Project_Gutenberg_eBook.txt` and `Project_Gutenberg_eBook1.txt` is `0.9900`, closely matching their Jaccard similarity. Other pairs have lower Minhash signature similarities, reflecting the low jaccard similarities seen in the initial comparisons. Dashing effectively approximates Jaccard similarity, allowing the solution to operate efficiently with fixed length signatures. The high signature similarity between `Project_Gutenberg_eBook.txt` and `Project_Gutenberg_eBook1.txt` reinforces the strong content overlap identified in the jaccard similarity step making this pair a likely candidate for LSH detection.



```
Generating MinHash signatures for each document...
1.txt.txt - MinHash signature generated.
10.txt - MinHash signature generated.
2.txt - MinHash signature generated.
3.txt - MinHash signature generated.
4.txt - MinHash signature generated.
5.txt - MinHash signature generated.
6.txt - MinHash signature generated.
7.txt - MinHash signature generated.
8.txt - MinHash signature generated.
9.txt - MinHash signature generated.
Project_Gutenberg_eBook.txt - MinHash signature generated.
Project_Gutenberg_eBook1.txt - MinHash signature generated.
```

d. **Locality-Sensitive Hashing (LSH):** Search for other optimises the similarity detection process by identifying candidate pairs that are likely to be similar. By using banded men hash signatures, LSH groups similar documents into the same buckets, reducing unnecessary comparisons. Mahesh Signature is divided into bands (5 bands of 20 rows each), Each band hash to create a banned specific bucket. Documents that share a bucket in at least one band are considered candidate pairs for detailed similarity comparison. Report shows that only one candidate pair was identified by LSH: ('Project_Gutenberg_eBook.txt', 'Project_Gutenberg_eBook1.txt'). Estimated similarity for this pair is 0.9900, matching its minhash similarity and indicating a very high level of similarity between these documents. LHS successfully filtered out pairs with low similarity, focusing on the most similar pair. This result demonstrates that LSH works as intended, identifying only highly similar document pairs as candidates. Given the lower similarity across most documents, this outcome alliance with expectations as only near duplicate documents like ('Project_Gutenberg_eBook.txt', 'Project_Gutenberg_eBook1.txt') qualify as candidate.



```
Candidate Pairs from LSH:
('Project_Gutenberg_eBook.txt', 'Project_Gutenberg_eBook1.txt')

Estimated Similarities for Candidate Pairs:
Similarity between Project_Gutenberg_eBook.txt and Project_Gutenberg_eBook1.txt: 0.9900
```

3. Instructions on How to Build and Run:

- Setup Environment: Ensure Python is installed. Create and activate a virtual environment.
    - ✓ python -m venv env
    - ✓ .\env\Scripts\activate
- Install Dependencies: Install required packages.
    - ✓ pip install -r requirements.txt
- Run the program: To run the main program and generate similarity results.
    - ✓ python main.py

4. Conclusion:

This assignment successfully demonstrated the implementation of document imilarity detection using Shingling, MinHashing and Locality-Sensitive Hashing (LSH). Through these techniques, it efficiently identified similar documents in a dataset by first capturing unique content characteristics, then approximating similarity with MinHash signatures, and finally filtering candidate pairs with LSH. The total time taken for each stage across all documents was as follows: **Shingling** took around **1 second**, **MinHashing** was the most time-consuming at **30 seconds**, and **Locality-Sensitive Hashing (LSH)** required **1 second**. Altogether, the complete processing time for the dataset was **32 seconds**, demonstrating the solution's efficiency and scalability for similarity detection across multiple documents. The solution effectively reduced computational costs by limiting direct comparisons to only likely similar pairs, as evidenced by the high similarity detected between near duplicate documents. Overall, this approach provides a scalable framework for similarity detection applicable to large text data sets where efficient processing is essential.