Data Mining Assignment 1

Zihao Xu zihxu@kth.se

Minchong Li mincli@kth.se

Date: November 8, 2023

1 Short Summary of our Solutions

1.1 Introduction

This homework is aimed at implementing the stages of finding textually similar documents based on Jaccard similarity using the shingling, minHashing, LSH (locality-sensitive hashing) techniques and corresponding algorithms.

The homework is implemented with Python.

1.2 Dataset Construction

We use 2 datasets for the assignment.

The first dataset is part of the text from the davisWiki corpus for this assignment. We selected 10 files for this assignment, which were retrieved based on the following keywords respectively: ”zombie”, ”restaurant”, ”computer”.

Another dataset is a series of texts related generated by ChatGPT. These texts are generated by telling ChatGPT to express the same concept in different ways. These texts are more semantically related.

Dataset can be download from here.

1.3 Shingling

The Shingling part takes text as input and returns k-shingles (where we set = 5) as a Set. In this part, in order to minimize conflicts, we use built-in Python hash functions for mapping, and the

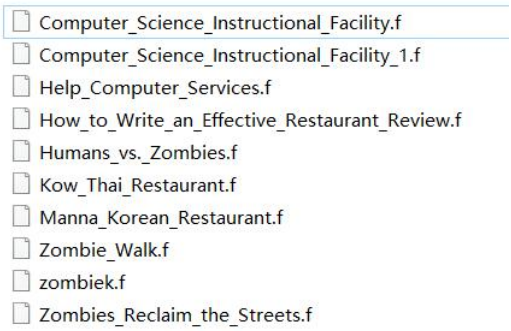

Figure 1: Selected files from davisWiki corpus

mapped results are further processed to reduce the sparsity of the results. Detailed implementation can be seen in `shingling.py`

1.4 Compare Sets

We calculate the Jaccard similarity of different sets of k-shingles. Detailed implementation can be seen in `compareSets.py`

1.5 minHashing

The minHashing part generates a vector (length defined by the user) by applying the min-hashing algorithm with a group of random hash functions. Detailed implementation can be seen in `minHashing.py`

1.6 Compare Signatures

This part compares signatures similarity to estimate the Jaccard similarity between two signatures representing their original documents.Detailed implementation can be seen in `compareSignatures.py`

1.7 LSH (locality-sensitive hashing)

The LSH part takes as input the signature matrix M representing documents, a similarity threshold t, and the number of bands. This function effectively generates pairs of documents with a similarity greater than t and returns them in a set format. The function identifies candidate pairs

by partitioning the hashed signatures into bands and treats two documents as candidates in case of collisions. Detailed implementation can be seen in `lsh.py`

2 How to Build and Run

Firstly, download the dataset and unzip it into the assignment1 root.

Secondly, install the requirements:

```
pip install -r requirements.txt
```

Then, use the following command to run the program:

```
cd ./src/
python main.py ../corpus --k 5 --n 500 -m 1000000007 -t 0.01 -b 5
python main.py ../corpus_test --k 5 --n 500 -m 1000000007 -t 0.1 -b 20
```

Arguments explanation:

- required positional argument: path of corpus.
- `--k`: Length of a single shingle (k-shingle)
- `--n`: Number of hash functions in signature
- `-m / --mod`: Maximum number of hash
- `-t / --threshold`: Threshold of being similar
- `-b / --band`: Band width in LSH

3 Results

Results of davisWiki corpus:

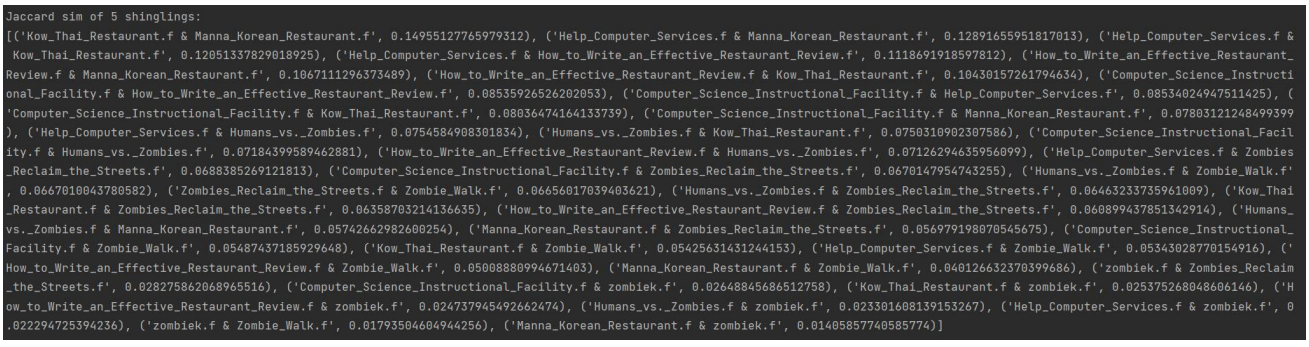

Figure 2: Result of davisWiki corpus

Results of texts generated by ChatGPT:

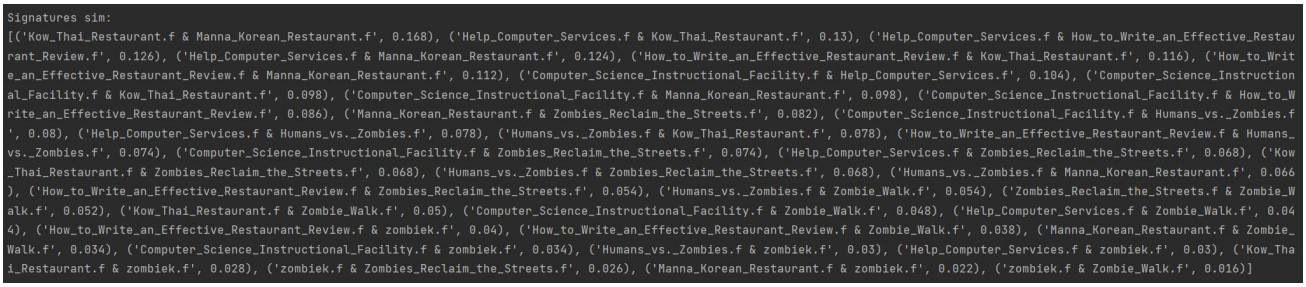

Figure 3: Result of davisWiki corpus

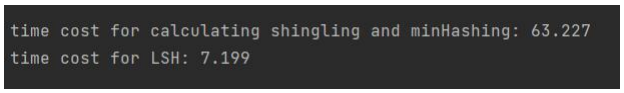

Figure 4: Time cost of davisWiki corpus

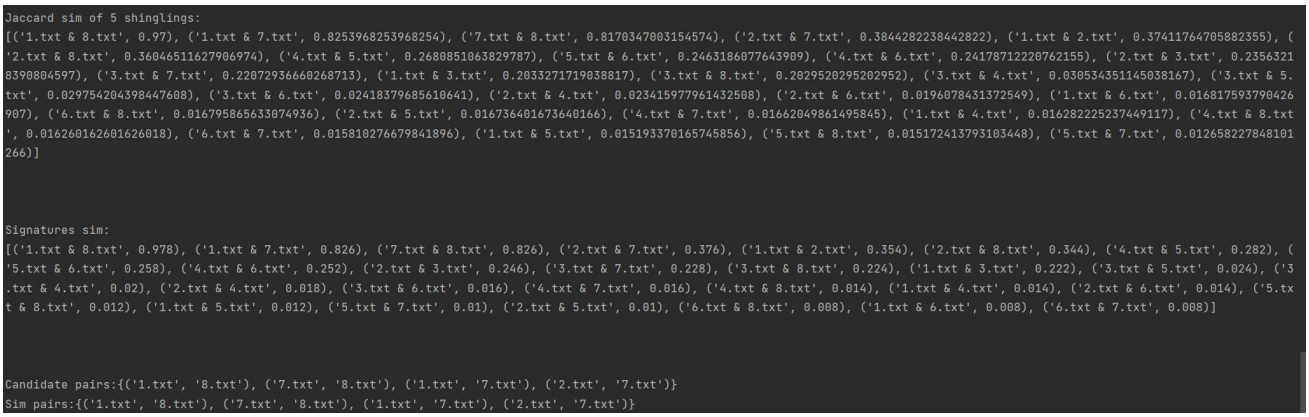

Figure 5: Result of ChatGPT generated texts

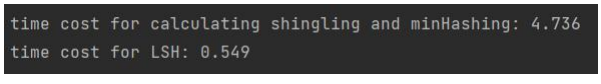

Figure 6: Time cost of ChatGPT generated texts