ID2222 Data Mining Homework 1

GIOVANNI MANFREDI SEBASTIANO MENEGHIN
gioman | meneghin @kth.se

13th November 2023

1 Introduction

Our project consists of the full five points presented in the project description, including the optional point 5. We have create a set of classes able to perform Shingling, MinHashing and LSH on the provided dataset.

As first step we enter the data from a .zip file present in our project folder and then we process those data. Then the selected data, according to some parameter that can be inserted by the users, pass through the procedure described above and the final results is the full list of similar documents, according to their signature similarity.

The timing results are presented in the section 4, where the *project_executor.py* is a reduced to a *example.py*, to show how the provided code can be scalable and how it could work on big or small datasets.

2 Data extraction and processing

The dataset used in this project is the Persaude Corpus 2.0, available on Kaggle. This comprises over 25,000 argumentative essays produced by 6th-12th grade students in the United States for 15 prompts on two writing tasks. The file can be easily downloaded within a .zip folder, together with its precedent version, Persuade Corpus 1.0. In this project, the default value for the number of essays taken by this document is 100, which also implies that the same 100 first essays are extracted, to make the result of the program more reproducible.

The data present in the essay are with diverse white spaces, accents, special characters, new line indicators and in a mix of lowercase and uppercase. So, the file data_processor.py provides classes and method to normalize and clean those data. Indeed, an object *DataProcessor()* is instantiated anytime the main program is called, and then its processing methods are called. However, if it's needed, they can easily deactivated by changing some boolean values in the code, that have the same name of the operations performed on the data.

3 Algorithm explanation

After the data are processed, the real algorithms of this homework are taken in consideration. As first step, the k-shingles, where k=10 by default, are created starting from each essay. Each shingles is hashed, with all its 10 characters, and is associated to a specific number from 0 to $2\hat{3}2$. Then, all the hashed shingles creates a single global dictionary of shingles

Pair of essays here can be confronted thanks to the sets of shingles that represent each of them.

The characteristic matrix is created, containing ones in cell represented by the intersection of a global shingle and an essay, meaning that the essay contains that specific shingle. MinHashing is then performed, with a default of signature numbers = 100. When all the hash functions are computed, the

```
(/home/sebastiano/miniconda3/envs/dm/hw1) sebastiano@Sebastiano-Meneghin:/mnt/c/Developer/University/DM/dm-2023-manfredi-meneghin/homework1$ time python3 example.py

################## SIMILARITY BETWEEN TWO ESSAYS ##################
The similarity between the two essays is: 0.009392121054004696

################## SIGN SIMILARITY BETWEEN FIVE ESSAYS ##################
The similarity between the signature of essay 1 and essay 2 is: 0.108
The similarity between the signature of essay 1 and essay 3 is: 0.206
The similarity between the signature of essay 1 and essay 4 is: 0.17
The similarity between the signature of essay 1 and essay 5 is: 0.19
The similarity between the signature of essay 2 and essay 3 is: 0.094
The similarity between the signature of essay 2 and essay 4 is: 0.064
The similarity between the signature of essay 2 and essay 5 is: 0.096
The similarity between the signature of essay 3 and essay 4 is: 0.142
The similarity between the signature of essay 3 and essay 5 is: 0.126
The similarity between the signature of essay 4 and essay 5 is: 0.098


################## SIMILAR PAIRS ##################
Couples of similar document are: []


real    0m2.547s
user    0m2.317s
sys     0m0.567s
```

Figure 1: Timings and results for example.py

```
(/home/sebastiano/miniconda3/envs/dm/hw1) sebastiano@Sebastiano-Meneghin:/mnt/c/Developer/University/DM/dm-2023-manfredi-meneghin/homework1$ time python3 project_exe
cutor.py
Dataset file: persuade_2.0_.zip
Number of essays: 100
Shingles length: 10
Number of signature: 100
Number of bands: 20
Threshold: 0.8
similar documents: []

real    0m14.896s
user    0m14.387s
sys     0m0.746s
(/home/sebastiano/miniconda3/envs/dm/hw1) sebastiano@Sebastiano-Meneghin:/mnt/c/Developer/University/DM/dm-2023-manfredi-meneghin/homework1$
```

Figure 2: Timings and results for project_executor.py

min-hash is found for each of the essay, for each of the signature, and then the signature matrix is created.

Here pair of essays can be confronted on their signature similarity.

Lastly, LSH algorithms are applied to the so far processed data. The data are divided in a default of 20 bands, of 5 rows each. Then they are hashed in different buckets. As final results, similar documents are created.

4 Results

As said before, there are two runnable files that have been used for the project, present into the project folder with the name of *project_executor.py* and *example.py*. Both have been tested on a WSL subsystem built on Windows 11 running on a laptop with 16GB RAM and CPU Intel i7 series 12.

The smaller project, evaluating 2 essays and then 5 essays with a more difficult tasks, containing the full procedure of *project_executor.py* performed as shown in Figure 1

Instead for *project_executor.py*, containing 100 essays as input data and evaluating the similarity on all of them, the results have shown, as expected, how the time needed increases with the increasing amount of data provided in the project. In this code, less print operations are present, since the only goal is to show which are the similar essay. The results are presented in Figure 2.

It has to be noticed that the shown results do not find any similar documents, since the documents chosen as data set are loosely similar. However, lowering the threshold t, suggested by the task description as t=0.8, to a lower level, such as t=0.2, pairs of less-similar documents can be found.

5 How to run

In order to run the project, you must have installed some packages in your own python environment. What is needed can be found in *requirements.text*, that can actually used to install the through the pip command. We have used Python 3.10.6 for this specific tasks and we suggest to use Python 3.8.0 or an higher version, since some commands of some primitives and libraries have been changed from older versions' ones. The full project can easily be run via terminal using the command *python3 project_executor.py* and the timings can be found, if you are running it on Linux, using *time python3*

project_executor.py Since the dataset is not provided, you are also asked to download it from the source provided as described in section 2. Once you have downloaded the .zip folder and placed it in a folder called *datasets*, your program will recognize the provided files and then process them.