Data Mining Homework 1: Finding Similar Items, Textually Similar Documents

Massimo Perini
Samuel Leonardo Gracio

November 2018

1 Short explanation of the program

Our program is coded in python, mainly using the library pyspark. First of all, we have one python file made to clean the data set : in input, it takes the different data sets that we've chosen for the lab, according to what was on the website. This code just put the different files in the right format: a Spark SQL table with one row for each document and different columns that corresponds to different fields like the path of the document, the title, the abstract.... Then,

there is the main file. At the beginning, you have the initialization of the parameters, which is choosing the different files you want to use and the number of elements you want to work with. Then you have the main cell that contains all the different classes and functions. Finally, you have the "test" cell where you can call all the different functions in order to test the program, divided for each step of the homework.

2 Instructions to test the program

Our program has several functions :

- def shingle(text, shingle_length): which creates the shingles of a text. Its parameter are the source text and the length of the shingles.

- The shingling class that creates the objects shingles and which is the main class. Its parameters are the documents and the number of shingles, which is 10 by default. In this class you have all the others functions :

- def multiple_shingle(self, field = "Abstract", merged = False): which hash the document and return you for each document multiple [(document_id, hash),...] values in an RDD.

-def compare_docs(self, rdd1, rdd2, merged = False) which is made to compare two documents. As input, you have the two documents and the function will return "float(common_items)/float(n_hashes)" which is basically counting the total number of hashes and counting how many they have in common.

-def min_hashing(self, n_func, merged = False): that computes different hashes function and return the minimum of them. In input, you put the number of hashing functions.

- def compare_signatures(self, rdd1, rdd2): that computes and compares two signatures of two different documents as input.

- def spark_lsh(self, band_size): which implements the LSH technique. As input, you have the band_size which is the number of bands you want. This will compute the threshold t that we use to compare the documents according to the LSH technique.

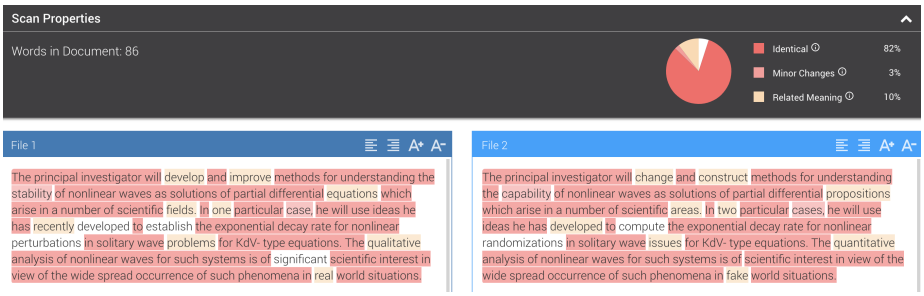
All the others functions are more mathematical or useful functions that are called only to compute some mathematical operations during the program.

3 Results

We used our programm on the Abstract dataset available on https://archive.ics.uci.edu/ml/datasets/NSF+Rese 2003. We used 8 for the shingle length which is a good compromise between long and short texts.

In order to test it, we did try it with two texts, one random text picked in the dataset and then, one modified text which was almost the same text but with some different words.

Here is how different the text are, according to copyleaks.com :



So we've tested our program on it and these are the results :

```
('Exact for same document', 1.0)
('Exact rate for modified document', 0.5466284074605452)
('Approximation for same document', 1.0)
('Approximation for the modified document', 0.535)
```

What we can see there is that our program is able to know if two documents are exactly the same. For the modified document, the program gives some good results because its still more than 50% similar. This is due to the shingle length we've choosen.

For the difference between the exact computation and the approximation using only the signatures, this gives almost the same results which was what we expected.