Data Mining
Laboratory 1

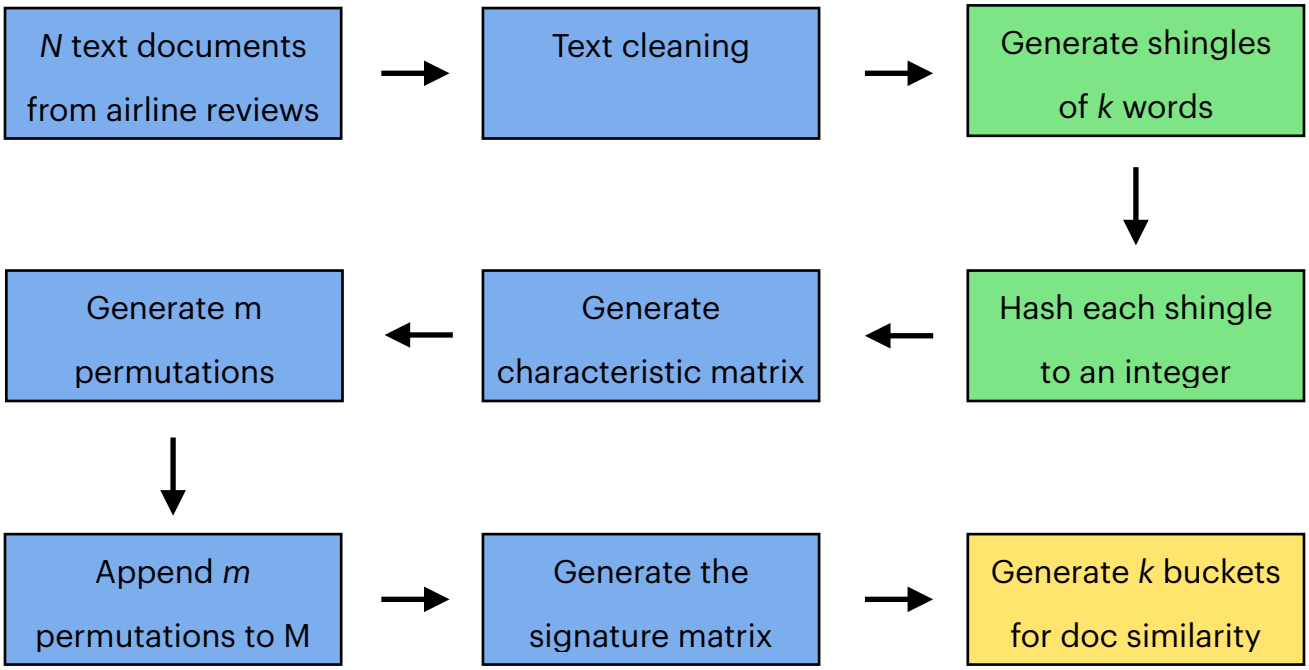
Finding Similar Items

Juan Álvarez Fernández del Vallado & Zhaopeng Tao
9 November 2020

1. Solution proposed

We propose a solution for finding similar documents in large text corpuses. Our current solution is based on Python and PySpark.

The following image shows the pipeline of the solution we propose.



The data used for this laboratory was airline reviews from a free dataset. From this dataset, we selected 15 documents to test our implementation. Moreover, a similar analysis is done in a different Python module called *main2* where we do the same analysis but to a shorter corpus composed of three sentences each sentence representing a document.

The *Text Cleaning* step makes sure all the words are lower and no punctuation, dashes, brackets or special symbols appear in the processed text.

The step *Generate shingles of k words* is implemented in a separate Python module that is imported to the *main and main2* module. This module is called *Shingling.py* and it extracts the shingles from a document given a shingling length and the integer hash of each shingle. Afterwards, the *Hash each shingle to an integer* step, which is done in the Shingling class, we

define a set comparator class called *CompareSets.py*, which is responsible of calculating the Jaccard similarity between two sets of hashes. This class mainly reads all the document's hashes and compares them using the following formula:

$$Jaccard(A, B) = \frac{|A \bigcap B|}{|A \bigcup B|}$$

In this formula, A and B are integer lists, each of their values representing the hash value of a shingle. It is important to note that the documents selected as A and B are the *first* document from the corpus and the rest of the documents, that is why this is done inside a loop iterating over the 15 documents.

The *Generate characteristic Matrix* consists of creating an empty matrix of 15 columns (one for each document) and as many rows as total unique shingles are. Afterwards, we fill the matrix with ones and zeros, one if a shingle is in a column.

The next step consists of generating permutations. In the code, the number of row permutations generated are defined as *signature_num* and in the pipeline above as *m*. For each permutation there is a hash function with random coefficients that generated the values of the permutation. It's important to mention that each permutation does *not* guarantee that there will not be a column permutation collision (i.e. it can happen that permutation number n has two rows indicating the same row number) but we can consider this is very rare. In the *main2.py* module this problem was solved resetting the random hashing coefficients that generated the permutations each time a collision appeared but this method is not scalable to large text corpuses, like in *main.py*. The next step of the pipeline consists simply of appending these permutations to the characteristic matrix.

The next step (*Generate the Signature Matrix*) is about performing the permutations previously obtained in the characteristic matrix (the one with 1s

and 0s) and generating the signatures the first and lowest one column position. This step is done iterating through the signature matrix.

The last step is the bonus point. It consists of performing the LSH algorithm to the signature matrix. For this, we partition the LSH matrix into *b* groups of rows (each partition will have *r*) rows. The values of *b* and *r* are chosen to get a similarity threshold of ~0.8. For each column in band b, we hash it and append the value to a bucket (there are *k* buckets) defined. We repeat this step for each column from the band we are. The idea behind is those documents similar will go to the same buckets and this will indicate us that those documents are similar. For this we store each bucket in a dictionary as a key and as a value we use an array for every band which will contain the documents similar to that band in that array.

2. Execute the code

To run the code you can do the actions explained in the *README.md* file. Install *python3, numpy, mmh3 and pyspark*. Clone the repository and change directory to the home repository folder. Run the main class and the program will start:

```
python3 main.py
```

The data is already in the repository as well as the default values, which are indicated as comments in the code and are also indicated in the next section.

3. Default values

The default values selected for this laboratory will be explained in the following table:

Variable name	Variable value	Brief explanation	Value
Number of documents	n	Number of documents to compare from dataset	15
Shingle size	s	Number of words grouped together	3
Number of permutations	signature_num	Number of new permutations to generate	50
Number of bands	b	Number of bands to divide matrix for LSH	6
Number of rows	r	Number of rows conatained in each band for LSH	8
Number of buckets	k	Number of buckets to generate for LSH	100

Important to mention is that the values of *b* and *r* already define the similarity threshold, which is ~0.8. Another thing is that this values apply *only* for the *main.py* module and not for the *main2.py*.

4. Results

This section will overview the results obtained for this laboratory.

The Jaccard Similarity between the first document and the rest of the documents before doing LSH (just using the characteristic matrix) for the main class:

```
Similarity between the 2 sets is 1.0
Similarity between the 2 sets is 0.0
Similarity between the 2 sets is 0.0
Similarity between the 2 sets is 0.0
Similarity between the 2 sets is 0.0
Similarity between the 2 sets is 0.0
Similarity between the 2 sets is 0.0
Similarity between the 2 sets is 0.0
Similarity between the 2 sets is 0.0
Similarity between the 2 sets is 0.0
Similarity between the 2 sets is 0.0
Similarity between the 2 sets is 0.0
Similarity between the 2 sets is 0.0
Similarity between the 2 sets is 0.0
```

Indeed this values are not good since there is only one non-zero value and it doesn't really mean anything because that is the similarity between document zero with document zero. We believe this happens because the chosen dataset is not really similar. Another hypothesis is that the shingle size may be too big (s=3) because if you lower it to s=2 the results get a little better, but we are not convinced about changing the shingling size.

We repeated the same experiment for the signatures. This consisted in comparing two columns form the signature matrix and estimating the similarity between them as a fraction of components which agree. This time we compared document number 4 instead of number one (as done previously):

```
0.0
0.0
0.030927835051546393
0.0
1.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
```

Clearly these results are a bit better but still this shows that the documents are not similar between each other.

Finally, we will present the results from applying the LSH algorithm. Since we used bands for this parts, there will be as many set of results as bands. To interpret this, the arrays grouped together represent those documents which are in the same bucket for that particular band.

```
dict_values([[0, 9, 13], [1], [2], [3], [4, 6], [5], [7], [8], [10], [11], [12], [14]])
dict_values([[0, 9, 13], [1], [2], [3], [4, 6], [5], [7], [8], [10], [11], [12], [14]])
dict_values([[0, 9, 13], [1], [2], [3], [4, 6], [5], [7], [8], [10], [11], [12], [14]])
dict_values([[0, 9, 13], [1], [2], [3], [4, 6], [5], [7], [8], [10], [11], [12], [14]])
dict_values([[0, 9, 13], [1], [2], [3], [4, 6], [5], [7], [8], [10], [11], [12], [14]])
dict_values([[0, 9, 13], [1], [2], [3], [4, 6], [5], [7], [8], [10], [11], [12], [14]])
dict_values([[0, 9, 13], [1], [2], [3], [4, 6], [5], [7], [8], [10], [11], [12], [14]])
dict_values([[0, 9, 13], [1], [2], [3], [4, 6], [5], [7], [8], [10], [11], [12], [14]])
dict_values([[0, 9, 13], [1], [2], [3], [4, 6], [5], [7], [8], [10], [11], [12], [14]])
dict_values([[0, 9, 13], [1], [2], [3], [4, 6], [5], [7], [8], [10], [11], [12], [14]])
dict_values([[0, 9, 13], [1], [2], [3], [4, 6], [5], [7], [8], [10], [11], [12], [14]])
dict_values([[0, 9, 13], [1], [2], [3], [4, 6], [5], [7], [8], [10], [11], [12], [14]])
dict_values([[0, 9, 13], [1], [2], [3], [4, 6], [5], [7], [8], [10], [11], [12], [14]])
dict_values([[0, 9, 13], [1], [2], [3], [4, 6], [5], [7], [8], [10], [11], [12], [14]])
dict_values([[0, 9, 13], [1], [2], [3], [4, 6], [5], [7], [8], [10], [11], [12], [14]])
dict_values([[0], [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14]])
dict_values([[0], [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14]])
dict_values([[0], [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14]])
dict_values([[0], [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14]])
dict_values([[0], [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14]])
dict_values([[0], [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14]])
dict_values([[0], [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14]])
dict_values([[0], [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14]])
dict_values([[0], [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14]])
dict_values([[0], [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14]])
dict_values([[0], [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14]])
dict_values([[0], [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14]])
```

```
dict_values([[0], [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14]])
dict_values([[0], [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14]])
dict_values([[0], [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14]])
dict_values([[0, 8], [1], [2], [3], [4], [5], [6], [7], [9], [10], [11], [12], [13], [14]])
dict_values([[0, 8], [1], [2], [3], [4], [5], [6], [7], [9], [10], [11], [12], [13], [14]])
dict_values([[0, 8], [1], [2], [3], [4], [5], [6], [7], [9], [10], [11], [12], [13], [14]])
dict_values([[0, 8], [1], [2], [3], [4], [5], [6], [7], [9], [10], [11], [12], [13], [14]])
dict_values([[0, 8], [1], [2], [3], [4], [5], [6], [7], [9], [10], [11], [12], [13], [14]])
dict_values([[0, 8], [1], [2], [3], [4], [5], [6], [7], [9], [10], [11], [12], [13], [14]])
dict_values([[0, 8], [1], [2], [3], [4], [5], [6], [7], [9], [10], [11], [12], [13], [14]])
dict_values([[0, 8], [1], [2], [3], [4], [5], [6], [7], [9], [10], [11], [12], [13], [14]])
dict_values([[0, 8], [1], [2], [3], [4], [5], [6], [7], [9], [10], [11], [12], [13], [14]])
dict_values([[0, 8], [1], [2], [3], [4], [5], [6], [7], [9], [10], [11], [12], [13], [14]])
dict_values([[0, 8], [1], [2], [3], [4], [5], [6], [7], [9], [10], [11], [12], [13], [14]])
dict_values([[0, 8], [1], [2], [3], [4], [5], [6], [7], [9], [10], [11], [12], [13], [14]])
dict_values([[0, 8], [1], [2], [3], [4], [5], [6], [7], [9], [10], [11], [12], [13], [14]])
dict_values([[0, 8], [1], [2], [3], [4], [5], [6], [7], [9], [10], [11], [12], [13], [14]])
dict_values([[0, 8], [1], [2], [3], [4], [5], [6], [7], [9], [10], [11], [12], [13], [14]])
dict_values([[0, 2], [1], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14]])
dict_values([[0, 2], [1], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14]])
dict_values([[0, 2], [1], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14]])
dict_values([[0, 2], [1], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14]])
dict_values([[0, 2], [1], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14]])
dict_values([[0, 2], [1], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14]])
dict_values([[0, 2], [1], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14]])
dict_values([[0, 2], [1], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14]])
dict_values([[0, 2], [1], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14]])
dict_values([[0, 2], [1], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14]])
dict_values([[0, 2], [1], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14]])
dict_values([[0, 2], [1], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14]])
dict_values([[0, 2], [1], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14]])
dict_values([[0, 2], [1], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14]])
dict_values([[0, 2], [1], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14]])
dict_values([[0], [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14]])
dict_values([[0], [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14]])
dict_values([[0], [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14]])
dict_values([[0], [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14]])
dict_values([[0], [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14]])
dict_values([[0], [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14]])
dict_values([[0], [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14]])
dict_values([[0], [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14]])
dict_values([[0], [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14]])
dict_values([[0], [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14]])
dict_values([[0], [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14]])
dict_values([[0], [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14]])
dict_values([[0], [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14]])
dict_values([[0], [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14]])
dict_values([[0], [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14]])
dict_values([[0], [1, 11], [2], [3], [4], [5], [6], [7, 14], [8, 10], [9], [12], [13]])
dict_values([[0], [1, 11], [2], [3], [4], [5], [6], [7, 14], [8, 10], [9], [12], [13]])
dict_values([[0], [1, 11], [2], [3], [4], [5], [6], [7, 14], [8, 10], [9], [12], [13]])
dict_values([[0], [1, 11], [2], [3], [4], [5], [6], [7, 14], [8, 10], [9], [12], [13]])
dict_values([[0], [1, 11], [2], [3], [4], [5], [6], [7, 14], [8, 10], [9], [12], [13]])
dict_values([[0], [1, 11], [2], [3], [4], [5], [6], [7, 14], [8, 10], [9], [12], [13]])
dict_values([[0], [1, 11], [2], [3], [4], [5], [6], [7, 14], [8, 10], [9], [12], [13]])
dict_values([[0], [1, 11], [2], [3], [4], [5], [6], [7, 14], [8, 10], [9], [12], [13]])
dict_values([[0], [1, 11], [2], [3], [4], [5], [6], [7, 14], [8, 10], [9], [12], [13]])
dict_values([[0], [1, 11], [2], [3], [4], [5], [6], [7, 14], [8, 10], [9], [12], [13]])
dict_values([[0], [1, 11], [2], [3], [4], [5], [6], [7, 14], [8, 10], [9], [12], [13]])
dict_values([[0], [1, 11], [2], [3], [4], [5], [6], [7, 14], [8, 10], [9], [12], [13]])
dict_values([[0], [1, 11], [2], [3], [4], [5], [6], [7, 14], [8, 10], [9], [12], [13]])
dict_values([[0], [1, 11], [2], [3], [4], [5], [6], [7, 14], [8, 10], [9], [12], [13]])
dict_values([[0], [1, 11], [2], [3], [4], [5], [6], [7, 14], [8, 10], [9], [12], [13]])
dict_values([[0, 8], [1], [2], [3], [4], [5], [6], [7], [9], [10], [11], [12], [13], [14]])
dict_values([[0, 8], [1], [2], [3], [4], [5], [6], [7], [9], [10], [11], [12], [13], [14]])
dict_values([[0, 8], [1], [2], [3], [4], [5], [6], [7], [9], [10], [11], [12], [13], [14]])
dict_values([[0, 8], [1], [2], [3], [4], [5], [6], [7], [9], [10], [11], [12], [13], [14]])
dict_values([[0, 8], [1], [2], [3], [4], [5], [6], [7], [9], [10], [11], [12], [13], [14]])
dict_values([[0, 8], [1], [2], [3], [4], [5], [6], [7], [9], [10], [11], [12], [13], [14]])
dict_values([[0, 8], [1], [2], [3], [4], [5], [6], [7], [9], [10], [11], [12], [13], [14]])
dict_values([[0, 8], [1], [2], [3], [4], [5], [6], [7], [9], [10], [11], [12], [13], [14]])
dict_values([[0, 8], [1], [2], [3], [4], [5], [6], [7], [9], [10], [11], [12], [13], [14]])
dict_values([[0, 8], [1], [2], [3], [4], [5], [6], [7], [9], [10], [11], [12], [13], [14]])
dict_values([[0, 8], [1], [2], [3], [4], [5], [6], [7], [9], [10], [11], [12], [13], [14]])
dict_values([[0, 8], [1], [2], [3], [4], [5], [6], [7], [9], [10], [11], [12], [13], [14]])
dict_values([[0, 8], [1], [2], [3], [4], [5], [6], [7], [9], [10], [11], [12], [13], [14]])
dict_values([[0, 8], [1], [2], [3], [4], [5], [6], [7], [9], [10], [11], [12], [13], [14]])
dict_values([[0, 8], [1], [2], [3], [4], [5], [6], [7], [9], [10], [11], [12], [13], [14]])
```

Now we will output the results for the *main2.py* module to compare them. The data used was the following:

The sky is blue and the sun is bright.

The sun in the sky is bright.

We can see sun is bright, the sky is blue.

These are the three documents used. Now we will output the results from the Jaccard similarity for the characteristic matrix and a shingle size 3:

```
Similarity between the 2 sets is 1.0
Similarity between the 2 sets is 0.1
Similarity between the 2 sets is 0.25
```

Now the signature vector similarity estimation:

```
1.0
0.0
0.09090909090909091
```

And finally the LSH:

```
dict_values([[0], [1], [2]])
dict_values([[0], [1], [2]])
dict_values([[0], [1], [2]])
dict_values([[0], [1], [2]])
dict_values([[0], [1], [2]])
dict_values([[0], [1], [2]])
dict_values([[0], [1], [2]])
dict_values([[0], [1], [2]])
dict_values([[0], [1], [2]])
dict_values([[0, 1], [2]])
dict_values([[0, 1], [2]])
dict_values([[0, 1], [2]])
dict_values([[0], [1], [2]])
dict_values([[0], [1], [2]])
dict_values([[0], [1], [2]])
dict_values([[0], [1], [2]])
dict_values([[0], [1], [2]])
dict_values([[0], [1], [2]])
```

In this case, the values for *b* and *r* chosen make a similarity threshold of 0.1. Otherwise, if we use a higher similarity threshold there will not be any match.