Data Mining: Similar Objects

Braulio Grana Gutiérrez Adrián Ramírez del Río

November 20, 2016

1 Description

For this assignment we had to implement the stages of finding textually similar objects based on Jaccard similarity. In order to achieve this, we implemented shingling, minhashing and, as a bonus task, locality-sensitive hashing. Our solution was implemented using Python3.

1.1 Shingling

In order to compute the shingles for a document we created a class named *Shingling* which receives the shigles'0 size in the constructor and defined a method called *transform* that iterates over the document creating the shingles using the following formula:

$$shingle = doc[i : i + shingle\_size]$$

where i has a step of 1. Then, we compute their hash values using python3 integrated hash function.

Finally, we implemented a function *compare_shingles* that returns the Jaccard similarity between two shingles.

1.2 Minhashing

We implemented a class called *MinHashing* that takes one parameter in its constructor, the number of hash functions to be applied to the shingles. The hash functions generated are as follows:

$$hash = (a \cdot shingle + b) \ mod \ c$$

where a and b are random integer, *shingle* is a hashed shingle and c is a large prime number, in our case $c = 4294967311$.

A method called *transform* applies each generated hash function to each shingle and then selects the lowest hash for each shingle which becomes the signature of the document.

Additionally, we implemented a *compare_signatures* function that calculates the percentage in which two document signatures are similar.

1.3 Locality-Sensitive Hashing

For this part we implemented a class named *LSH* that generates candidate pairs of signatures that are possibly similar and should be compared. This class' constructor takes two parameters: the list of signatures, and the threshold at which we considerate a pair of signatures as candidates.

Using the threshold we calculate the number of bands in which we will divide the signature matrix by solving the following equations system:

$$signature\_size = r \cdot b$$
$$threshold = (1/b)^{1/r}$$

where *signature_size* and *threshold* are known constants, b is the number of bands and r the number of rows per band.

Once we have the matrix divided in bands, we apply a hashing function (in our case the norm) to every signature part in each band. If the percentage of bands of two signatures that computed to the same hash value (bucket) is greater than the threshold (times the number of bands) we consider the signatures as a candidate pair.

2 Instructions

In order to execute our solution certain libraries need to be installed beforehand. To install them, run the following commands (it is possible that you will need admin access):

```
$ apt-get install python3 python3-pip
$ pip3 install scipy numpy
```

To run the code, simply go to the project folder and run:

```
$ python3 src/main --dir DATASET_DIR \
--n SHINGLE_SIZE \
--k SIGNATURE_SIZE \
--th SIMILARITY_THRESHOLD
```