Data Mining, ID2222 HT201

Homework Assignment 1

Frano Rajic, Ivan Stresec

November 9, 2020

1 Objective

Our objective in this homework was to implement the stages of finding textually similar documents based on Jaccard similarity using shingling, MinHashing, and locality-sensitive hashing (LSH) techniques and corresponding algorithms.

2 Solution

Using Python 3 with some of its builtin functions and the NumPy library, we implemented the five classes described in the homework assignment which each represent some method or part of the standard text document comparison pipeline. For hashing we always used the default Python hashing function.

2.1 `Shingling`

The class `Shingling` constructs k-shingles of a given length k from a given document, computes a hash value for each unique shingle, and represents the document in the form of an ordered set of its hashed k-shingles. We read files line by line using the standard Python Files I/O, removed special characters and any double spaces and separators, and creates hashes of all possible shingles and saves them to a set. During construction the Shingling object is given a set called `universe` which remembers all generated hashed shingles which are later used for the creation of the *characteristic* matrix in the MinHashing method.

2.2 `CompareSets`

The class `CompareSets` computes the Jaccard similarity of two sets of integers - two sets of hashed shingles. Comparisons are made with simple set algebra Python offers (the length of the intersection divided by the length of the union).

2.3 `MinHashing`

Our implementation of the class `MinHashing` builds MinHash signatures in the form of a matrix of with signatures of a given length n from a given array of sets of integers (sets of hashed shingles). The constructor also takes the universe containing all possible hashed shingles in some order which allows the creation of the *characteristic* matrix. We used an implementation of MinHashing using permutations like described in the lectures, the n permutations are randomly generated using the NumPy library and then applied to each row of the *characteristic* matrix.

2.4 `CompareSignatures`

The class `CompareSignatures` estimates the similarity of two integer vectors - MinHash signatures - as a fraction of components, in which they agree.

2.5 `Lsh`

The class `Lsh` implements the LSH technique. Given a collection of MinHash signatures (integer arrays) and a similarity threshold t, the `Lsh` class uses banding and hashing to find all candidate pairs of signatures that agree on at least fraction t of their components which can be retrieved after the object construction. Hashing of bands is done by converting the bands to tuples and using the default Python hash function.

3 Testing and evaluation

We tested the performance of our implementation on two distinct examples: 17 science articles concerning neural networks and a database of Croatian song lyrics containing 58000 songs.

3.1 Science articles

We used 17 science articles concerning neural networks and compared performances of doing Jaccard similarity directly, and then using the MinHash and LSH methods we implemented. To create a comparative result, we used each of the methods to generate all similar candidate pairs with some threshold t. One of the articles was duplicated to make sure there are no false negatives in our implementation. The results of a test using shingles of size $K = 5$, MinHash

signatures of size $N = 50$, LSH using bands of size $B = 5$, and a threshold of 0.3 can be seen in Figure 1.

```
Got 141878 brownies in total for 18 documents.
%%% Load papers: 0.3238635999999999s %%%

Bare candidate_pairs for threshold 0.3:
[('1502.03167.txt', '1502.03167_DUPLICATE.txt', 1.0), ('2003-Fei-Fei_ICCV03.txt', '2006-Fei-FeiFergusPerona2006.txt',
 0.30247871094368295), ('koch2015msc-thesis.txt', 'koch2015oneshot.txt', 0.5403414123520413)]
%%% Load papers + candidate_pairs: 0.7149935999999999s %%%

minhash_candidates for threshold 0.3:
[('1502.03167.txt', '1502.03167_DUPLICATE.txt', 1.0), ('koch2015msc-thesis.txt', 'koch2015oneshot.txt', 0.56)]
%%% Load papers + minhash_candidates: 0.5019893s %%%

lsh_candidates for threshold 0.3:
[('1502.03167.txt', '1502.03167_DUPLICATE.txt', 1.0)]
%%% Load papers + lsh_candidates: 0.49929179999999995s %%%
```

Figure 1: Test results

Here is a quick analysis of the results.

Let's look at the calculated Jaccard similarities. We can see that the two duplicate texts have a similarity of 1.0 as expected. Other than that we can see relatively high similarities between two pairs of articles, with articles in pairs being written by the same authors, Fei-Fei and Koch. Generating the hashed shingles for each work as well as comparing them using Jaccard similarity took about 0.71 seconds, with around 0.38 seconds of that taken up by calculating Jaccard similarities.

Now we do the same thing using the MinHashing technique. As expected, MinHash detected the duplicate, and also managed to detect the similarity between two articles written by the same author Koch, but the estimated similarity between Fei-Fei was lower than our threshold of 0.3 and failed to make the cut. Due to the randomness of permutations, there were also runs in which the signature similarity was larger than the Jaccard similarity of the original hashed shingles. There is, however, a noticeable improvement in time performance; shingling and comparing MinHash signatures of size $N = 50$ took just slightly above 0.5 seconds, with just 0.18 seconds for the MinHash generation and comparisons, and provided pretty accurate estimates of the true Jaccard similarity. Ignoring the process of shingling the documents, which can't be avoided, MinHash was twice as quick as calculating Jaccard similarities.

Then, we test our implementation of LSH. We can see that it found the duplicate successfully as we could expect, but it failed to find other similar texts. Banding seems to be a very discriminative procedure in general and so, at least on these examples, its estimates of similarity were much lower than those of Jaccard and MinHashing. It took approximately the same time as MinHashing, so in this case it was not worth using.

Finally, we ran a few more tests to accommodate LSH. We lowered the size of shingles to $K = 3$ which created more similarity between sets and ran our

test again using other parameters unchanged. As expected, we could see a lot more pairs generated with classic Jaccard comparisons and MinHashing, and LSH managed to find, other than the duplicate, similarity between two articles written by Koch. However, our implementation of LSH was still as quick as MinHashing, so its worth is quite debatable.

3.2 Croatian song lyrics

For further data mining, we scraped 58000 song lyrics from a Croatian website tekstovi.net and used the implemented LSH technique to find song entries that are identical or have high similarity. The LSH technique took about 40 minutes or more, depending on shingle size. We tested different values of threshold for these two configurations of parameters:

- 5-shingles, signature length of 100, band size of 10 and 20; resulted in 301492 different shingles
- 9-shingles, signature length of 100, band size of 20; resulted in 8889062 different shingles

For the first configuration and a high threshold of 0.7, we got 124 candidates, some of which we selected to show in the table 1. In figure 2 a comparison of almost identical texts is shown.

First song	Second song	Similarity
Thompson: Sokolov krik	Thompson: Ora et labora	0.75
Duo Mariol: Luce mala	Oliver Dragojevic: Luce mala	0.75
Luka Nizetic: Moje prvo pijanstvo	Oliver Dragojevic: Moje prvo pijanstvo	1
Oliver Dragojevic: Nista nova	Radojka Sverko: Nista nova	0.75
Jole: Meni trebas ti	Oliver Dragojevic: Meni trebas ti	0.75
Kemal Malovcic: Tebe nema	Ramiz Redzepovic: Tebe nema	0.75
Alen Islamovic: Ja nemam prava	4 Asa: Ja nemam prava	0.75
Magazin: Opusti se	Jelena Rozga: Opusti se	1
Magazin: Oprosti, mala	Jelena Rozga: Oprosti mala	0.75
Halid Beslic: Uzalud je vino	Halid Beslic: Rastanak	0.75
Pedja Medenica: Ne pitaj za mene	Brzi: Hvala Bogu, dobro sam	0.75
Natasa Djordjevic: Ne mogu bez tebe	Vanesa: Ne mogu bez tebe	1
Jala Brat: Hrsuz (feat. Buba Corelli)	Buba Corelli: Hrsuz (feat. Jala)	1
Hakala: Zmaj od Bosne	Safet Isovic: Zmaj od Bosne	0.75
Seven Up: Kad ljubav nije sudjena	SMS: Kad ljubav nije sudjena	1
Rade Serbedzija: Uzalud je budim	Branko Miljkovic: Uzalud je budim	0.75
Riblja Corba: Kad hodas	Bajaga: Kad hodas	1

Table 1: Some of the candidates found for 5-shingles, 0.7 threshold, band size of 20 and signature length of 100

We can see that LSH produced duplicate candidates successfully, and did so in a relatively scalable fashion, considering the amount of texts it was given.

Kemal Malovcic

Tebe nema

Ovaj tekst je pregledan 12 371 puta

Like 0 Share 0

Jos uvijek te volim, u srcu si mom
jos zovem kroz noc dusom ranjenom
2x

Ref.
Ali tebe nema kao nekada
za sva vremena drugom pripadas
ali tebe nema, zasto tako, ljubavi
za sva vremena mene ostavi

Jos bagrem cvjeta, behari prolaze
i dani mladosti s nadom odlaze
2x

Ref.

Sve je kao nekad, jesen ce doc'
a meni ostaje da lutam kroz noc
2x

Ref.

Hvala ***elly***

14.07.2009 - 15:13

Dodaj Youtube-video | Prijavi grešku u tekstu

Ramiz Redzepovic

Tebe nema

Ovaj tekst je pregledan 7 566 puta

Like 0 Share 0

Jos uvijek te volim
u srcu si mom
jos zovem kroz noc
dusom ranjenom
(2x)

Ref.
Ali tebe nema kao nekada
za sva vremena drugom pripadas
ali tebe nema, zasto tako, ljubavi
za sva vremena mene ostavi

Jos bagrem cvijeta
behari prolaze
i dani mladosti
s nadom odlaze
(2x)

Ref.

Sve je k'o nekad
jesen ce doc'
a meni ostaje
da lutam kroz noc
(2x)

Ref.

Hvala ***MeLcHe***

Figure 2: Comparison of two lyrics for very similar songs of resulting LSH similarity of 75%

4 Conclusion

We have implemented and tested common methods used in finding similar items, specifically similar text documents. MinHashing was noticeably faster than calculating Jaccard similarities on original hashed shingles, but our implementation of LSH didn't seem to be much faster than MinHashing. Clearly, using smaller shingles produced higher similarity between texts, while using bigger shingles reduced the similarity; depending on the context we can control the level of similarity by changing the size of shingles. Also, depending on the need for better performance, we can use MinHashing with more or less permutations to achieve lower or higher speeds with more or less accurate estimates of similarity, respectively.