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`

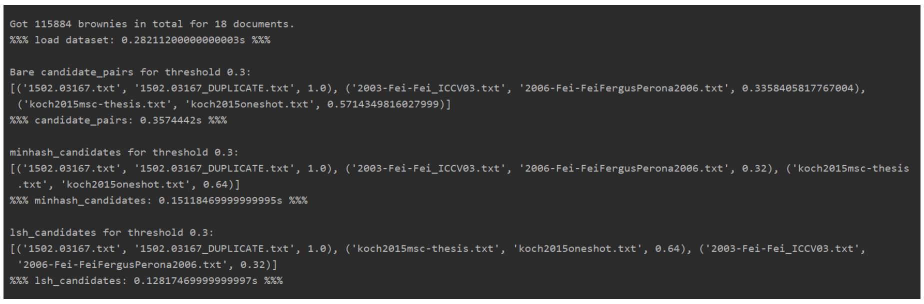
The class `Lsh` implements the LSH technique. Given a collection of MinHash signatures (integer arrays) and a similarity threshold t, the `Lsh` class uses banding and hashing to find all potential candidate pairs of signatures that agree on at least 1 of their hashed bands. Hashing of bands is done by converting the bands to tuples and using the default Python hash function. Once the potential candidates are found their MinHash signatures are compared to get their similarity estimates.

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 $K = 6$, MinHash signatures of size $N = 50$, LSH using bands of size $B = 2$, and a threshold of 0.3 can be seen in Figure 1. It's also worth noting that using the formula for LSH we get a threshold of $(\frac{1}{b})^{\frac{1}{r}} = (\frac{1}{25})^{\frac{1}{2}} = 0.2$, lower than the threshold, so we lower the possibility of getting false negatives. This, however, slows down LSH to some extent.



```
Got 115884 brownies in total for 18 documents.
%%% load dataset: 0.28211200000000003s %%%

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', 0.3358405817767004),
 ('koch2015msc-thesis.txt', 'koch2015oneshot.txt', 0.5714349816027999)]
%%% candidate_pairs: 0.3574442s %%%

minhash_candidates for threshold 0.3:
[('1502.03167.txt', '1502.03167_DUPLICATE.txt', 1.0), ('2003-Fei-Fei_ICCV03.txt', '2006-Fei-FeiFergusPerona2006.txt', 0.32), ('koch2015msc-thesis
 .txt', 'koch2015oneshot.txt', 0.64)]
%%% minhash_candidates: 0.15118469999999995s %%%

lsh_candidates for threshold 0.3:
[('1502.03167.txt', '1502.03167_DUPLICATE.txt', 1.0), ('koch2015msc-thesis.txt', 'koch2015oneshot.txt', 0.64), ('2003-Fei-Fei_ICCV03.txt',
 '2006-Fei-FeiFergusPerona2006.txt', 0.32)]
%%% lsh_candidates: 0.12817469999999997s %%%
```

Figure 1: Test results

Here is a quick analysis of the results.

First, to shingle the dataset we needed around 0.28 seconds, afterwords we can compare different methods.

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. Comparing the documents using Jaccard similarity took about 0.36 seconds.

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, and two articles written by Fei-Fei. Due to the randomness of permutations, there were runs in which the signature similarity was larger and smaller than the Jaccard similarity of the original hashed shingles. There is, however, a noticeable improvement in time performance; comparing MinHash signatures of size $N = 50$ took just slightly above 0.15 seconds and provided pretty accurate estimates of the true Jaccard similarity. In that respect, MinHash was more than twice as quick as calculating Jaccard similarities.

Finally, we tested our implementation of LSH. We see it was able to find all the candidates and it was slightly quicker than MinHashing ($0.13s$). The threshold we observed was 0.3, and LSH's theoretical threshold was 0.2. If we wanted more speed we could have used bands of higher width, however it would come with an increase in false negatives.

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 45 minutes or more, depending on shingle size. We tested different values of threshold for these two configurations of parameters:

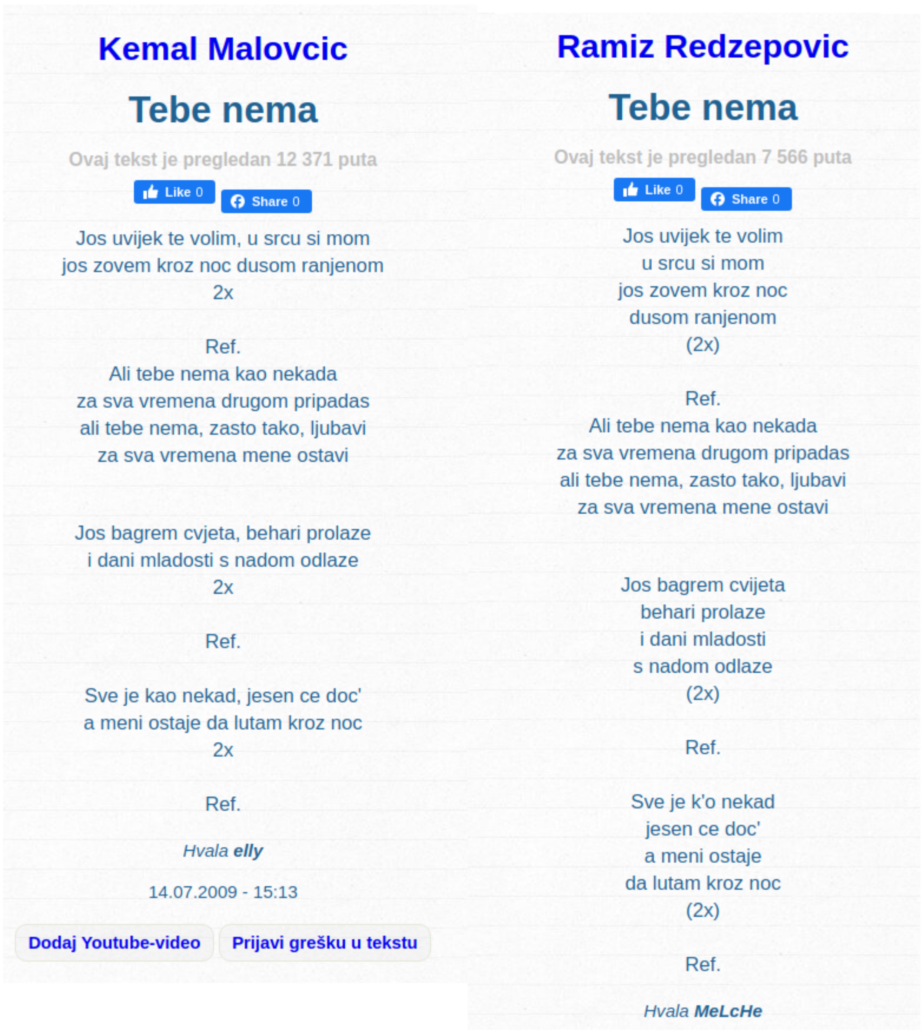
- 5-shingles, signature length of 100, band width of 5; resulted in 301492 different shingles
- 9-shingles, signature length of 100, band width of 20; resulted in 8889062 different shingles

For the first configuration and a high threshold of 0.5, we got 124 candidates, some of which we selected to show in the table 1. In figure 2 a comparison of almost identical lyrics is shown, where as in figure 3 a comparison of very similar lyrics is shown.

First song	Second song	Similarity
Hanka Paldum: Ako ti kazem	Nada Obric: Ako ti kazem	0.62
Sinan Sakic: Sinoc sretoh prijatelja	Sinan Sakic: Iznenada u kafani staroj	0.93
Dzej Ramadanovski: Sta to bese ljubav	Oliver Dragojevic: Sto to bjese ljubav	0.69
Marko Tolja: Moja prva ljubav	Oliver Dragojevic: Prva ljubav	0.91
Miroslav Skoro: Ne vjeruje srce pameti	Amadeus Band: Ne veruje srce pameti	0.74
Miroslav Skoro: Idem, idem ej, zivote	Ciro Gasparac: Idem, idem, ej zivote	0.95
Karolina Goceva: Kad volis	Tose Proeski: Kad volis	0.51
Zeljko Samardzic: Ostala si uvijek ista	Tose Proeski: Ostala si uvijek ista	0.83
Gazde: Jos i danas zamirise tresnja	Tose Proeski: Jos i danas zamirise tresnja	0.75
Amir Kazic Leo: Ostala si uvijek ista	Tose Proeski: Ostala si uvijek ista	0.94
Tiho Orlic: Stari se	Thompson: Stari se (duet Tiho Orlic)	0.83
Thompson: Sokolov krik	Thompson: Ora et labora	0.96
Kemal Malovcic: Tebe nema	Ramiz Redzepovic: Tebe nema	0.97

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

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

Ramiz Redzepovic

Tebe nema

Ovaj tekst je pregledan 7 566 puta

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)

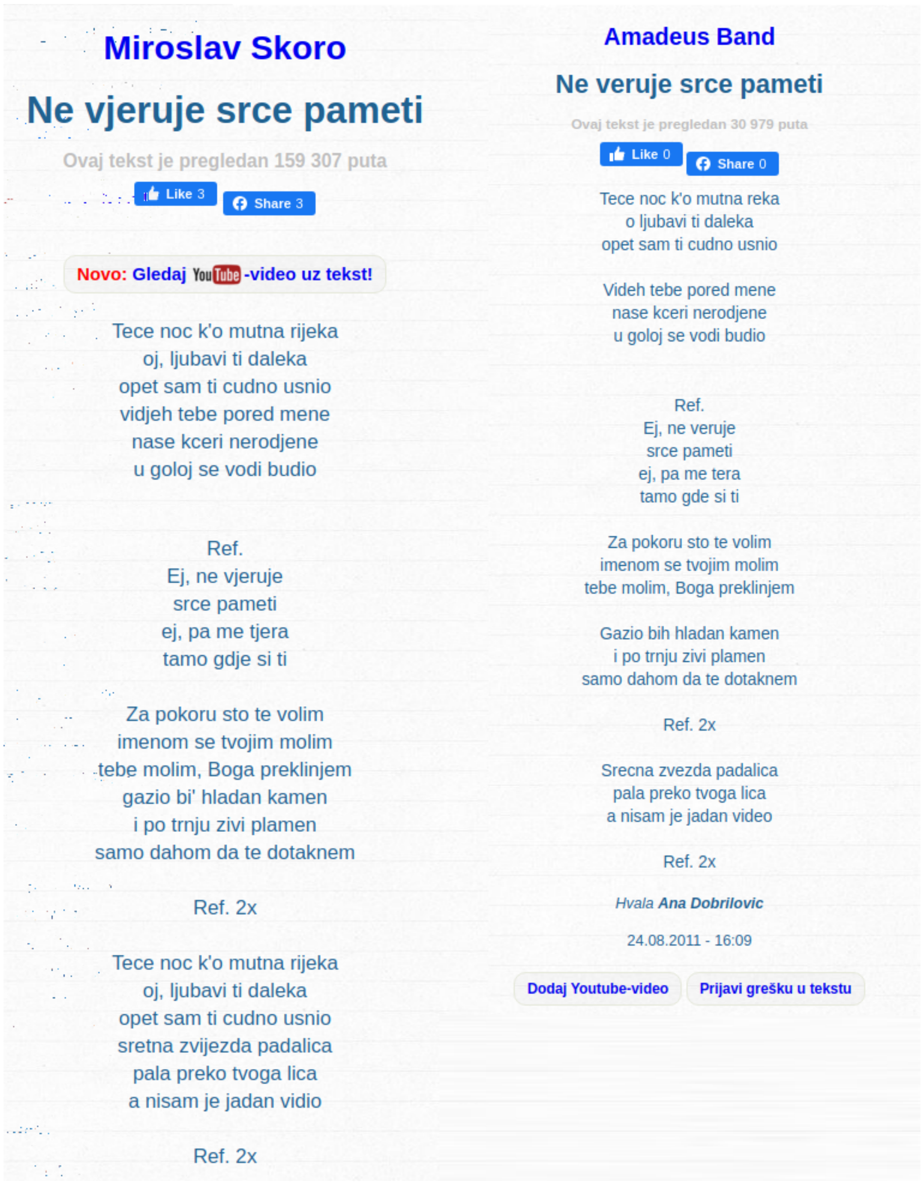
Ref.

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 almost identical songs of resulting LSH similarity of 97%



Miroslav Skoro

Ne vjeruje srce pameti

Ovaj tekst je pregledan 159 307 puta

Tece noc k'o mutna rijeka
oj, ljubavi ti daleka
opet sam ti cudno usnio
vidjeh tebe pored mene
nase kceri nerodjene
u goloj se vodi budio

Ref.
Ej, ne vjeruje
srce pameti
ej, pa me tjera
tamo gdje si ti

Za pokoru sto te volim
imenom se tvojim molim
tebe molim, Boga preklinjem
gazio bi' hladan kamen
i po trnju zivi plamen
samo dahom da te dotaknem

Ref. 2x

Tece noc k'o mutna rijeka
oj, ljubavi ti daleka
opet sam ti cudno usnio
sretna zvijezda padalica
pala preko tvoga lica
a nisam je jadan vidio

Ref. 2x

Amadeus Band

Ne veruje srce pameti

Ovaj tekst je pregledan 30 979 puta

Tece noc k'o mutna reka
o ljubavi ti daleka
opet sam ti cudno usnio

Videh tebe pored mene
nase kceri nerodjene
u goloj se vodi budio

Ref.
Ej, ne veruje
srce pameti
ej, pa me tera
tamo gde si ti

Za pokoru sto te volim
imenom se tvojim molim
tebe molim, Boga preklinjem

Gazio bih hladan kamen
i po trnju zivi plamen
samo dahom da te dotaknem

Ref. 2x

Srecna zvezda padalica
pala preko tvoga lica
a nisam je jadan video

Ref. 2x

*Hvala **Ana Dobrilovic***

24.08.2011 - 16:09

Figure 3: Comparison of two lyrics for very similar songs of resulting LSH similarity of 74%

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.