KTH Royal Institute of Technology

ID2222 Data Mining

ID2222 Homework 1

Finding Similar Items: Textually Similar Documents

Written by

Andrea Camilloni

Yutong Jiang

Date: November 28, 2022

1 Shingling

The Shingling technique is used to convert a document to a set of strings of length k (k-shingles). The function performs the shingling on a given document, constructing k–shingles of a given length k (e.g., 10). Given the text of a document, the length of the shingles and, optionally, a boolean variable *remove_blank*, it first represents the document as a list of k-shingles, then, it computes a hash value for each unique shingle and returns a set of hashed k-shingles. The variable *remove_blank* is set by default as False. If True, the text of the given document is first preprocessed to remove white space, new lines, etc. Shingling documents is insufficient due to the comparison algorithm's scalability. For a set of n documents, one must conduct a $\frac{n(n-1)}{2}$ comparison, which is practically $O(n^2)$. Imagine having 1 million documents, then the number of comparison will be $5 \cdot 10^{11}$. Since the document matrix is sparse, storing it in its current form will result in a significant memory overhead. We hashed the shingles because of this.

2 CompareSets

To compare two documents, we can use the Jaccard Similarity metric to compute the similarity of the sets of k-shingles of the two documents. Given two sets $C1$ and $C2$, the Jaccard Similarity is computed as follows:

$$sim(C1, C2) = \frac{|C1 \cap C2|}{|C1 \cup C2|} \tag{1}$$

So, given two sets of hashed shingles $C1$ and $C2$, the python function returns the similarity value $sim(C1, C2)$.

3 MinHashing

The MinHashing technique converts large sets to short signatures while preserving the similarity. The MinHashing is used to reduce the complexity for computing the similarity. The MinHashing function builds a minHash signature of a given length n from a given set of integers (a set of hashed shingles) for each element in the given input, i.e. the hashed shingles. The hash function at each iteration is computed as follows:

$$hash(a, b, x) = (ax - b)\%(2^{32} - 1) \tag{2}$$

where x is the shingle, a and b are random numbers between 1 and $1e4$. Thus, by removing the sparseness while maintaining the similarity, min-hashing was used to solve the space complexity problem.

4 CompareSignatures

Given two minhash signatures, the function *CompareSignatures()* estimates the similarity of the two as a fraction of components in which they agree.

5 LSH

Given a collection of minhash signatures and a number of bands, the LSH function (using banding and hashing) finds candidate pairs of signatures agreeing on at least a threshold t of their components. The threshold t is computed as follows:

$$t \approx (\frac{1}{b})^{\frac{1}{r}} = (\frac{1}{b})^{\frac{b}{n}} \tag{3}$$

where b is the number of bands, n the number of documents and r is the row length of the signature matrix. LSH significantly reduces the time complexity of the algorithm for finding similar pairs.

6 Code explanation

As we have implemented 2 codes by each person respectively, we will give brief introductions to both of them.

In the code implemented by Yutong, function of **shingle** is designed to construct k-shingles of a given length k. By setting **remove_blank** as True, the blank in the documents could be removed. Function of **compare_sets** is designed to compare the Jaccard Similarity of two sets. Function of **minhash** is designed to build a minhash signature of a given length k. **CompareSignatures** is designed to compare two signatures given by minhash function. **LSH** is designed to implement LSH function which is to find candidate pairs of signatures agreeing on at least a fraction t of their component. To run the code and check the results, block 1-8 in jupyter notebook should be ran in order.

In the code implemented by Andrea, function of **shingle_document** is designed to construct k-shingles of a given length k. The function of **jaccard_similarity** is designed to compare Jaccard Similarity of two sets. The function of **func** is to design a hash function and the function of **minhashing** is designed to build a minhash signature of a given length k. **CompareSignatures** is deisgned to compare signatures given by minhash function. Function of **LSH** is designed to build a find candidate pairs of signatures agreeing on at least a fraction of their component. To run the code, block 1, 11-19 should be ran in order.

7 Results

In our experiments, the length of each shingles is 3 without removing blank, the total signatures obtained by minhash is set to 100, the band in LSH is set to 25. Hence, the threshold is set as 0.4472. The number of documents we used are 10, 20 and 30.
The Jaccard similarity matrix for 10 documents is shown below.

```
array([[1.        , 0.34420096, 0.36344411, 0.33444909, 0.37790422,
        0.18502203, 0.38711195, 0.36795069, 0.3472177 , 0.34637802],
       [0.34420096, 1.        , 0.46225984, 0.44242424, 0.29666757,
        0.13131313, 0.29429107, 0.48647377, 0.4138093 , 0.35112285],
       [0.36344411, 0.46225984, 1.        , 0.43957845, 0.31881372,
        0.14383339, 0.31986532, 0.46182918, 0.4304653 , 0.35805022],
       [0.33444909, 0.44242424, 0.43957845, 1.        , 0.28977273,
        0.14173703, 0.30309923, 0.43434105, 0.41623743, 0.32969134],
       [0.37790422, 0.29666757, 0.31881372, 0.28977273, 1.        ,
        0.2098922 , 0.37170385, 0.32669197, 0.31355361, 0.31892369],
       [0.18502203, 0.13131313, 0.14383339, 0.14173703, 0.2098922 ,
        1.        , 0.19327217, 0.1454727 , 0.1555719 , 0.18306878],
       [0.38711195, 0.29429107, 0.31986532, 0.30309923, 0.37170385,
        0.19327217, 1.        , 0.32230889, 0.31607083, 0.31754996],
       [0.36795069, 0.48647377, 0.46182918, 0.43434105, 0.32669197,
        0.1454727 , 0.32230889, 1.        , 0.41786367, 0.37644553],
       [0.3472177 , 0.4138093 , 0.4304653 , 0.41623743, 0.31355361,
        0.1555719 , 0.31607083, 0.41786367, 1.        , 0.36126329],
       [0.34637802, 0.35112285, 0.35805022, 0.32969134, 0.31892369,
        0.18306878, 0.31754996, 0.37644553, 0.36126329, 1.        ]])
```

Figure 1: Jaccard similarity matrix

The similarity matrix we get by minhash for 10 documents is shown below

```
array([[1.  , 0.35, 0.4 , 0.29, 0.42, 0.17, 0.35, 0.31, 0.31, 0.29],
       [0.35, 1.  , 0.47, 0.47, 0.41, 0.16, 0.31, 0.55, 0.46, 0.44],
       [0.4 , 0.47, 1.  , 0.51, 0.37, 0.14, 0.35, 0.54, 0.46, 0.38],
       [0.29, 0.47, 0.51, 1.  , 0.32, 0.11, 0.29, 0.51, 0.43, 0.39],
       [0.42, 0.41, 0.37, 0.32, 1.  , 0.18, 0.39, 0.36, 0.35, 0.36],
       [0.17, 0.16, 0.14, 0.11, 0.18, 1.  , 0.17, 0.15, 0.1 , 0.19],
       [0.35, 0.31, 0.35, 0.29, 0.39, 0.17, 1.  , 0.3 , 0.34, 0.37],
       [0.31, 0.55, 0.54, 0.51, 0.36, 0.15, 0.3 , 1.  , 0.46, 0.48],
       [0.31, 0.46, 0.46, 0.43, 0.35, 0.1 , 0.34, 0.46, 1.  , 0.36],
       [0.29, 0.44, 0.38, 0.39, 0.36, 0.19, 0.37, 0.48, 0.36, 1.  ]])
```

Figure 2: Similarity matrix obtained by minhash

The results of similar pairs and specific time for jaccard similarity/ minhash and LSH are shown below.

doc	pairs of Jaccard Similarity	pairs of minhash	pairs of LSH
10	3	10 (time 0.6053s)	8 (time 0.0244s)
20	5	30 (time 1.3926s)	25 (time 0.0617s)
30	18	64 (time 1.8279s)	53 (time 0.1206s)