HW1 - Finding Textually Similar Documents

Tianxiao Zhao <tzh@kth.se>
Yantian You <yantian@kth.se>

Build and run

To run our program, change the name of input document to Integer in sequence(e.g 1,2,3...10), change the variable "*route*" to the path of the input documents, and then run the test class(including main function).

Solution

In homework 1, we created 5 classes, including *Shingling, CompareSets, RandomMinHash, CompareSignatures, text* and *Test*, to find textually similar documents. Here is the introduction of the 5 classes.

Shingling: This class contains a *ShingleHash* Function which will constructs k-shingles of a given length k from a given document. *ShingleHash* also computed the hash value of each shingle by using java *hashcode()* function. It will return a *HashMap<Integer,String>* in which stores hash code as key and corresponding shingle as value.

CompareSets: This class contains *ComputeJaccard* which computes the Jaccard similarity of two Input sets based on hashed shingles we achieved from Shingling class.

RandomMinHash: Including two function, *BooleanSet function* and *Signature function*. *BooleanSet* function converts original shingle sets into a characteristic matrix contains Boolean column-vectors based on a Union Shingle Set. *Signature* function returns a signature of a given length n by using the characteristic matrix we achieved from *BooleanSet* function. We used several hash function of the type *h(x)=(ax+b)%c* to compute different hash sets for one text and chose the minimum hash value as one minHash value for each hash set. The combination of minHash value is the signature for this text. Set a and b are randomly generated and c is the length of Union Shingle Set.

CompareSignatures: It contains a similar function which estimates similarity of two minhash signatures. This similarity is the fraction of the minhash values(rows) in which they agree.

Text: Presents a text object that we need to compare. It contains a hashed shingle set (s), a boolean characteristic vector (Boolean) and a text signature (signatures).

Test: This class realized testing function including a main function. It can automatically import documents we need to compare and compute the shingle sets, Jaccard similarities, signatures and Signature similarities among all the input documents.

Results

To test our implementation, we randomly choose a corpus of several documents from the dataset *Opinosis Opinion/ Review Data Set*, and use our implementation to evaluate the similarities between pairs of documents. Here we provide an example of those documents we picked:

comfort_honda_accord_2008.txt

"Drivers seat not comfortable, the car itself compared to other models of similar class .
It's very comfortable, remarkably large inside and just an overall great vehicle .
Front seats are very uncomfortable .
I'm 6' tall, and find the driving position pretty comfortable .
However, there are a couple of things that kill it for me 1 terrible driver seat comfort, kills my back 2 lack luster interior design, my Acadia has much better comfort 3 the VCM drives me crazy because the constant change in cylinder use is perceptible enough to be an annoyance..."

We picked 8 documents like this and each of them is somewhat lengthy since it contains a collection of user reviews on a given topic. Table 1 shows the detailed information of the documents we used. As we can see, there are some potential similarities between different documents - we can estimate that document 1 and 2 should be relatively similar since they touch a similar topic.

Table 1. Information of picked documents

Document id	Document name	Text length
1	comfort_honda_accord_2008.txt	14666
2	comfort_toyota_camry_2007.txt	10263

3	price_amazon_kindle.txt	10485
4	price_holiday_inn_london.txt	14590
5	screen_ipod_nano_8gb.txt	5738
6	screen_netbook_1005ha.txt	18868
7	staff_bestwestern_hotel_sfo.txt	30907
8	staff_swissotel_chicago.txt	18140

When the code is run, the documents above will first be shingled and converted into a set of Hashed k-shingles. Here we set k = 6 for our documents. We put the hash values and their corresponding shingles into a HashMap. The Hashed 6-shingles for, e.g. document 1, now have the following format:

{-1424119011=able c, -1297075536=er rev, -1360138516=city a, -1424119009=able e, -1424119010=able d, -934885310=redibl, -1424119013=able a, 1167394204='6 an, 1016478879= indee, -1337708466=de tha, 959641891= . It , 1348702327=. I dr, 1348702324=. I do, 959641898= . It', -747957385=xus an, -1424119037=able l, -1360138509=city h, -969734612=s look, 1016642745= issue, -941111357=t like, -1310019123=ed cit, 977222168= A lit, -892499227=stabil, -1297075557=er rea, -1424118991=able w, -1424118992=able v, -1297075555=er rec, -1424118995=able s, -941045799=t not , -1424118996=able r, 1017625748= just ….}

After this process, the hashed 6-shingles will be converted into a set of minhash signatures. In our case, we set the length of signatures n = 100 and get signatures for these 8 documents below:

Table 2. MinHash signatures for each document

Document id	MinHash signatures
1	[8, 0, 17, 3, 2, 2, 3, 0, 3, 6, 0, 0, 2, 0, 0, 3, 0, 2, 2, 6, 3, 3, 0, 0, 0, 3, 0, 0, 0, 2, 0, 2, 0, 0, 2, 17, 0, 17, 0, 83, 3, 0, 0, 3, 0, 0, 3, 0, 6, 0, 0, 2, 3, 0, 0, 0, 0, 0, 2, 2, 0, 2, 0, 2, 0, 6, 0, 0, 0, 2, 2, 0, 0, 2, 3, 2, 6, 2, 83, 2, 0, 0, 28, 3, 0, 3, 0, 0, 2, 0, 17, 0, 0, 3, 0, 17, 0, 0, 2, 0]
2	[23, 9, 17, 3, 5, 8, 3, 0, 3, 6, 2, 2, 2, 0, 1, 8, 2, 2, 5, 6, 13, 3, 0, 11, 1, 3, 5, 7, 5, 2, 6, 20, 3, 11, 2, 17, 7, 17, 6, 83, 3, 4, 2, 13, 14, 2, 3, 6, 6, 0, 2, 2, 3, 6, 2, 0, 1,

	0, 2, 2, 6, 2, 0, 8, 2, 17, 1, 1, 0, 5, 2, 29, 8, 5, 3, 2, 6, 2, 83, 2, 2, 11, 28, 8, 8, 3, 0, 7, 2, 8, 17, 0, 2, 13, 6, 17, 0, 2, 5, 1]
3	[23, 13, 17, 3, 2, 17, 3, 95, 18, 17, 16, 14, 8, 12, 0, 33, 2, 5, 26, 39, 13, 18, 11, 15, 13, 8, 29, 13, 29, 2, 39, 5, 46, 15, 2, 17, 7, 17, 23, 83, 18, 1, 2, 13, 4, 6, 8, 23, 28, 0, 3, 8, 18, 39, 3, 5, 13, 1, 8, 2, 23, 8, 5, 8, 3, 6, 13, 13, 5, 26, 5, 7, 5, 50, 8, 8, 17, 2, 83, 8, 2, 15, 28, 33, 5, 3, 1, 18, 2, 8, 17, 12, 2, 13, 23, 17, 15, 3, 29, 13]
4	[8, 9, 17, 3, 5, 11, 13, 14, 3, 6, 0, 23, 8, 12, 16, 8, 2, 5, 14, 6, 8, 3, 0, 16, 19, 3, 1, 36, 1, 2, 20, 23, 3, 16, 29, 17, 7, 17, 6, 83, 3, 5, 2, 3, 14, 26, 3, 6, 6, 0, 3, 23, 3, 20, 1, 5, 0, 1, 23, 23, 6, 23, 5, 8, 1, 17, 1, 0, 5, 14, 5, 16, 5, 5, 8, 8, 6, 17, 83, 8, 2, 16, 28, 8, 5, 13, 1, 0, 2, 8, 17, 12, 2, 3, 6, 17, 5, 3, 11, 19]
5	[83, 29, 17, 3, 41, 83, 3, 15, 83, 6, 49, 14, 5, 15, 1, 8, 3, 5, 35, 39, 13, 83, 7, 16, 4, 8, 22, 13, 22, 17, 19, 116, 99, 16, 14, 17, 7, 17, 6, 83, 23, 3, 3, 38, 51, 72, 8, 6, 50, 0, 49, 8, 83, 19, 36, 8, 6, 12, 8, 2, 6, 8, 8, 26, 36, 6, 1, 6, 8, 35, 5, 5, 61, 17, 78, 5, 6, 29, 83, 5, 49, 16, 28, 8, 61, 3, 12, 0, 17, 6, 17, 15, 3, 38, 6, 17, 35, 49, 26, 4]
6	[8, 9, 17, 3, 20, 14, 8, 15, 18, 6, 6, 14, 2, 2, 1, 18, 4, 2, 8, 6, 8, 18, 12, 16, 4, 3, 16, 13, 16, 17, 7, 5, 10, 16, 11, 17, 7, 17, 14, 83, 18, 3, 4, 13, 7, 31, 3, 14, 28, 0, 3, 2, 18, 7, 13, 5, 41, 3, 2, 2, 14, 2, 5, 8, 13, 28, 1, 41, 5, 8, 2, 9, 32, 14, 8, 2, 6, 2, 83, 2, 2, 16, 28, 18, 32, 8, 3, 0, 17, 3, 17, 2, 4, 13, 14, 17, 5, 3, 44, 4]
7	[23, 23, 17, 3, 20, 11, 13, 19, 33, 6, 2, 13, 5, 15, 35, 33, 2, 5, 20, 6, 3, 33, 0, 16, 19, 13, 1, 2, 1, 11, 9, 20, 30, 16, 14, 17, 7, 17, 1, 83, 3, 5, 2, 3, 1, 2, 13, 1, 17, 0, 8, 2, 33, 9, 1, 5, 0, 1, 2, 5, 1, 2, 5, 26, 1, 6, 1, 0, 5, 20, 5, 16, 8, 5, 8, 5, 6, 2, 83, 5, 2, 16, 28, 33, 8, 13, 1, 0, 11, 3, 17, 15, 2, 3, 1, 17, 2, 8, 11, 19]
8	[8, 9, 17, 3, 20, 14, 13, 37, 43, 17, 6, 13, 2, 15, 16, 8, 4, 5, 17, 6, 3, 43, 7, 8, 2, 13, 1, 2, 1, 11, 10, 14, 20, 8, 14, 17, 7, 17, 1, 83, 3, 3, 4, 13, 1, 2, 13, 1, 6, 0, 3, 26, 43, 10, 4, 5, 0, 3, 26, 5, 1, 26, 5, 26, 4, 6, 1, 0, 5, 17, 5, 3, 11, 5, 8, 2, 17, 2, 83, 2, 2, 8, 28, 8, 11, 13, 3, 18, 11, 11, 17, 15, 4, 13, 1, 17, 6, 3, 56, 2]

According to the signatures above, the similarities between pairs of these documents can be obtained, see Table 3.

Table 3. Similarities of document pairs

	1	2	3	4	5	6	7	8
1	1.0	0.46	0.2	0.3	0.17	0.27	0.26	0.22

2	0.46	1.0	0.28	0.43	0.25	0.33	0.32	0.27
3	0.2	0.28	1.0	0.36	0.26	0.31	0.28	0.27
4	0.3	0.43	0.36	1.0	0.26	0.33	0.5	0.36
5	0.17	0.25	0.26	0.26	1.0	0.28	0.28	0.23
6	0.27	0.33	0.31	0.33	0.28	1.0	0.3	0.37
7	0.26	0.32	0.28	0.5	0.28	0.3	1.0	0.49
8	0.22	0.27	0.27	0.36	0.23	0.37	0.49	1.0

From the results above, we could find those similar document pairs by finding the highest similarity quantity in each row (excluding those self-comparing pair with a similarity of 1.0):
1 -> 2;
2 -> 1;
3 -> 4;
4 -> 7;
5 -> 6 and 7;
6 -> 8;
7 -> 4 (and 8 if the 0.01 difference with 4 is acceptable);
8 -> 7.

Conclusion

According to the testing results, the similar document pairs we achieved by using Jaccard similarity are very close to the pairs we achieved by using Signature Similarity. That is to say the signature we achieved through MinHash function can represent the given document with much smaller dataset. However, the similarity between two document is still very low and we believe this can be affected by the length of shingles(k) and the length of signature(n). Though smaller k may provide higher similarity, it will also create shingles that appear in most of document.