Asmt 2: Document Similarity and Hashing

Anirudh Narasimhamurthy(u0941400)

February 18, 2015

1 Creating k-Grams

In this part of the question I constructed k-grams for the given documents based on both characters and words. For the k-grams based on characters, 27 characters i.e 26 lower cased alphabets plus space was considered. In the initial copy of the documents, doc1 had a '-' and it also had `\r` and `\n` characters. I had handled them in my python script but I realized the documents were cleaned and the latest version does not have other characters apart from space and alphabets. For the k-grams based on words space was considered as the delimiter between the words and I had stored the list of all words in a list and then found out the k-grams for the individual documents.

Part A

Distinct k-Grams

For storing the k-grams of all the documents, I had made use of the set data structure in python. It also ensures that whenever an addition is made to the set, it only adds distinct elements and hence duplicates are handled and would not be able to enter. The list of distinct k-grams for both characters and words for all the 4 documents have been tabulated below:

k-Grams	Doc1	Doc2	Doc3	Doc4
2-grams characters	337	378	267	298
3-grams characters	1330	1590	853	1074
3-grams words	1021	1027	353	505

Table 1: Distinct k-grams for each document

These are the 12 different numbers which we had been asked to report for this part of the question.

Part B

Jaccard Similarity

Once the k-grams for all the documents was computed, calculating the Jaccard Similarity was a straight forward implementation. All the k-grams for the four different documents were stored in the sets and so the Jaccard Similarity between any two pairs of documents was computed using the following formula:

$$JS(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

In our case A and B might refer to Doc1, Doc2 or Doc2,Doc3 or Doc3,Doc4 and so on.

Programmatically this translates to finding the set of all k-grams common to two docuements and the total set of distinct k-grams in both documents. Python has built in methods for finding the union and intersection of two sets. Though it has built-in methods, I had coded the logic for union and intersection and used that in my program.

The Jaccard similarity was computed for the different pairs of documents and the results have been tabulated below:

Jaccard Similarity	**Doc1**	**Doc2**	**Doc3**	**Doc4**
Document 1		**0.8523**	**0.7207**	**0.7255**
Document 2			**0.6623**	**0.6984**
Document 3				**0.7121**

Table 2: Jaccard similarity for all document pairs for 2-grams characters

The Jaccard Similarity between the same documents would always be 1 and hence the diagnol entries for (Document1,Doc1), (Document2, Doc2) and (Document3,Doc3) are always 1. If we were to think in terms of sets these would be singleton sets and hence the union or the intersection would yield the set itself. This is not asked in the question. But I just wanted make my results clear for the grader.

Also JS(A,B)=JS(B,A) as per the definition. Hence the values for (Document2,Doc1), (Document3,Doc1) and (Document3, Doc2) have been left blank as they have already been computed and tabulated in (Document1,Doc2),(Document1, Doc3) and (Document2, Doc3) respectively

Jaccard Similarity	**Doc1**	**Doc2**	**Doc3**	**Doc4**
Document 1		**0.6957**	**0.4631**	**0.3984**
Document 2			**0.3864**	**0.3529**
Document 3				**0.3522**

Table 3: Jaccard similarity for all document pairs for 3-grams characters

Jaccard Similarity	**Doc1**	**Doc2**	**Doc3**	**Doc4**
Document 1		**0.3617**	**0.04169**	**0.00065**
Document 2			**0.016949**	**0.000653**
Document 3				**0.001166**

Table 4: Jaccard similarity for all document pairs for 3-grams words

There are six values in each of the three tables above which have been highlighted in bold font. These are the required 18 numbers to be reported for this question.

So this was the expalantion for how I had gone about computing the values which were asked to be reported for Question 1.

2 Min Hashing

Part A

Building MinHash signature and calculating Jaccard Similarity

In this part of the assignment, we are asked to build a min-hash signature for the documents D1 and D2 using 3-grams based on characters. This process has to be done with different values of t={10,50,100,250,500} hash functions. For each of the different values of we are also asked to report the Jaccard Similarity between the pair of documents.

Approximate Jaccard Similarity between the pair of documents D1 and D2 is given by:

$$JS_t(a,b) = \frac{1}{t} \cdot \sum_{i=1}^{t} \begin{cases} 1 & \text{if } a_i = b_i \\ 0, & \text{if } a_i \neq b_i \end{cases}$$

Methodology Used:

- To generate the different hash functions I made use of SHA1 hash function available in python. By default, the SHA1 hash function works only on strings and not on integers or floats. I had initially applied it to all the k-grams in Doc1 to check the values it was returning. I made use of the hexdigest() attribute to return the value
- The function was returning hexadecimal values and size of the resulting hash was 20 bytes. In other words the hash returned by the function comprised of 40 characters each of which is 4 bits.
- Since we are using min hashing, the signature for the document consists of a vector which has the minimum value among all the values returned by the particular hash function for all the k-grams in the document. Thus when we use t=5 hash functions our signature will be 5 x1 or 1 x 5 vector.
- If the corresponding values in the signatures of D1 and D2 match, then we give it a value of 1, else we give it 0. In other words if the first element of D1's signature is 1 and the first element of D2's signature is also 1, then we would give a value of 1 and compute the Jaccard Similarity as per the expression mentioned above.
- One more important point to be mentioned is that the different hash functions like t=100,250 and 500 were generated by encoding a salt with sha1. Salt was generated randomly and hence this would ensure that we pick a random hash function h from the hash family.

The approximate Jaccard similarity between the pairs of documents D1 and D2 were calculated and results have been tabulated below.

t	**JS(D1,D2)**
10	0.6
50	0.72
100	0.71
250	0.688
500	0.672

Table 5: Jaccard similarity for D1 and D2 based on min-hash signature

Part B

To determine good value of t, we could take different approaches and provide a justification for our answer. As mentioned by **Anusha (TA)** in the class forum, I would like to provide the justification in terms of the running time and the value computed and its closeness to the actual Jaccard Similarity.

The time taken by the different hash functions t=10,50,100,250,500 along with their Jaccard similarity computed via MinHashing signature is tabulated below:

t	**JS(D1,D2)**	Running time(in seconds)
10	0.6	0.0835
50	0.66	0.46788
100	0.70	0.946
250	0.688	2.3527
500	0.674	4.5416
1000	0.687	9.7218

Table 6: Jaccard similarity for D1 and D2 based on min-hash signature

A point to be mentioned. This time was taken or considered only for the lines of code that are computing the Jaccard similarity via the Min Hash signature and not for the entire program.

I also experimented with t=1000 and have tabulated the results. From the experiments, I found that a good value of t from the given values would be **t=100**. The reasons are provided below:

– The actual Jaccard Similarity between D1 and D2 is 0.6957 and the Jaccard Similarity obtained via minhashing signature for t=100 is 0.7 which is very close to the actual value.

– Although t=200 or t=500 would definitely be better considering they are doing several times more and taking an average out of that for Jaccard Similarity. But the problem as we see is the time taken for computing increases as 't' increases.

– The documents provided in the assignment were relatively small by any standards and so for this experiment, it might not seem to be a bad idea to pick t as say 200, but when min hashing is generally applied to large documents the execution time overhead might be there with respect to t=250 and t=500.

– One thing which was clear as to why we shouldn't choose a smaler value of *t* say like 10. This is because 10 iterations is really small and the simlarities computed using it might be way off either on the upside or on the downside.

– Hence t=100 is a good value which satisifes both our accuracy and time constraints. If we were allowed to take/choose an arbitrary value of *t* for the given documents, I feel a value in the range of 150-180 would be really ideal for the given pair of documents.

3 LSH

Part A

We are asked to report the best values of b and r for this question. Here r is the number of bands and b is the number of hashes per band.

Based on the information provided in the notes, we know that given the budget of the number of hash functions t and the threshold value of τ, the value of b can be approximately estimated using the following expression:

$$b \approx -log_{\tau}(t)$$

Plugging in the value of t as 100 and τ as 0.4 we get,

$b \approx -log_{0.4}(100)$

$b \approx 5.02588$

$\boxed{b = 5}$

The budget on the number of hash functions t is split/divided into r and b. Now that we have obtained the value of b in the previous step, we can obtain r using the following formula:

t= b. r

r= t/b

r=100/5

$\boxed{r = 20}$

We can verify if these values of t and r provide us an S-curve where on the x-axis we could have the JS(D1,D2) between the documents represented and the y-axis representing the probability that the pair $D1, D2$ is a candidate to check the true distance.

Using the expression we can find the values needed for the plot

$$f(s) = 1 - (1 - s^b)^r$$

In our case r=20 and b=5.

$f(0.1) = 0.0001$

$f(0.2) = 0.0063$

$f(0.3) = 0.0474$

$f(0.4) = 0.1860$

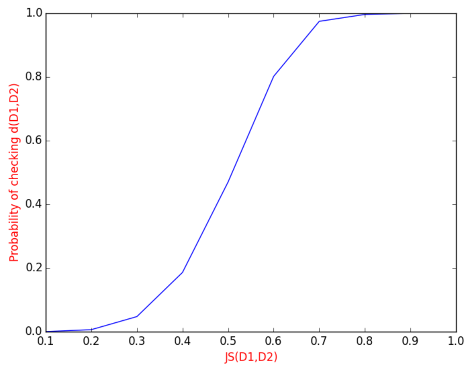
$f(0.5) = 0.4700$

$f(0.6) = 0.80190$

$f(0.7) = 0.9747$

$f(0.8) = 0.9964$

$f(0.9) = 0.9999$



(a) Probability that distance d(D1,D2) is checked in a LSH scheme with r=20 bands and b=5 hahses each

These are the values which were required to be reported for this question.

Part B

In this part we are asked to find what is the probability that each of pair of four documents will be estimated to have a similarity greater than τ

There are two possible explanations for the selection of Jaccard similarity for this question. The text book considers s as the Jaccard similarity which was computed via min hashing, while Professor's notes have s=JS(D1,D2) which was computed in problem 1.

I personally feel given that we have the Jaccard similarity of the pairs of documents which gives us very accurate value, using the approximate value of Jaccard Similarity computed using Min-hashing isn't entirely convinving. But I have decided to show the work for both the appoaches. the calculations are done below:

If we consider the s= Approximate Jaccard Similarity computed via Min Hashing

We have also computed the values of b and r in the previous part. Now we would have to plug in those values in the expression below:

$$f(s) = 1 - (1 - s^b)^r$$

Here f(s) relates to the probability LSH, with our choice of r and b will classify it as being above tau.

For Document1 and Document 2

$f(s) = 1 - (1 - s^b)^r$

$f(s) = 1 - (1 - 0.71^5)^{20}$

$\boxed{f(s) = 0.9813}$

For Document1 and Document 3

$f(s) = 1 - (1 - s^b)^r$

$f(s) = 1 - (1 - 0.42^5)^{20}$

$\boxed{f(s) = 0.23133}$

For Document1 and Document 4

$f(s) = 1 - (1 - s^b)^r$

$f(s) = 1 - (1 - 0.38^5)^{20}$

$\boxed{f(s) = 0.14709}$

For Document2 and Document 3

$f(s) = 1 - (1 - s^b)^r$

$f(s) = 1 - (1 - 0.41^5)^{20}$

$\boxed{f(s) = 0.20789}$

For Document2 and Document 4

$f(s) = 1 - (1 - s^b)^r$

$f(s) = 1 - (1 - 0.32^5)^{20}$

$\boxed{f(s) = 0.0650}$

For Document3 and Document 4

$f(s) = 1 - (1 - s^b)^r$

$f(s) = 1 - (1 - 0.37^5)^{20}$

$\boxed{f(s) = 0.1299}$

Probability that docs have similarity $> \tau$	**Doc1**	**Doc2**	**Doc3**	**Doc4**
Document 1		**0.9813**	**0.23133**	**0.14709**
Document 2			**0.20789**	**0.0650**
Document 3				**0.1299**

Table 7: Probability that each pair of documents will have similarity $> \tau$

If we consider s= Jaccard Similarity between the pairs of Documents

For Document1 and Document 2

$f(s) = 1 - (1 - s^b)^r$

$f(s) = 1 - (1 - 0.6957^5)^{20}$

$\boxed{f(s) = 0.9715}$

For Document1 and Document 3

$f(s) = 1 - (1 - s^b)^r$

$f(s) = 1 - (1 - 0.4631^5)^{20}$

$\boxed{f(s) = 0.3498}$

For Document1 and Document 4

$f(s) = 1 - (1 - s^b)^r$

$f(s) = 1 - (1 - 0.3984^5)^{20}$

$\boxed{f(s) = 0.1827}$

For Document2 and Document 3

$f(s) = 1 - (1 - s^b)^r$

$f(s) = 1 - (1 - 0.3864^5)^{20}$

$\boxed{f(s) = 0.1588}$

For Document2 and Document 4

$f(s) = 1 - (1 - s^b)^r$

$f(s) = 1 - (1 - 0.3529^5)^{20}$

$\boxed{f(s) = 0.1039}$

For Document3 and Document 4

$f(s) = 1 - (1 - s^b)^r$

$f(s) = 1 - (1 - 0.3522^5)^{20}$

$\boxed{f(s) = 0.1029}$

Probability that docs have similarity $> \tau$	**Doc1**	**Doc2**	**Doc3**	**Doc4**
Document 1		**0.9715**	**0.3498**	**0.1827**
Document 2			**0.1588**	**0.1039**
Document 3				**0.1029**

Table 8: Probability that each pair of documents will have similarity $> \tau$

These are the 6 required values which needs to be reported for this question.