Mining Big Data - Map-Reduce

Assignment 2

M. Vincent & A. Phansalkar

a1148120 & a1776879

School of Computer Science, The University of Adelaide

April 29, 2020

Report submitted for **COMP SCI 3306 Mining Big Data** at the School of Computer Science, University of Adelaide

Exercise 1

The S-curves for each combination of r and b parameters have been calculated below. The threshold t that defines how similar document need to be in order for be regarded as similar, can be approximated using the formula $t = (1/b)^{1/r}$. The threshold calculation has also been included below.

These results were obtained by computing each of the formulae displayed below in the Excel workbook included in the assignment submission.

s	$1 - (1 - s^3)^{10}$	s	$1 - (1 - s^6)^{20}$	s	$1 - (1 - s^5)^{50}$
0.1	0.010	0.1	0.000	0.1	0.000
0.2	0.077	0.2	0.001	0.2	0.016
0.3	0.239	0.3	0.014	0.3	0.115
0.4	0.484	0.4	0.079	0.4	0.402
0.5	0.737	0.5	0.270	0.5	0.796
0.6	0.912	0.6	0.615	0.6	0.983
0.7	0.985	0.7	0.918	0.7	1.000
0.8	0.999	0.8	0.998	0.8	1.000
0.9	1.000	0.9	1.000	0.9	1.000

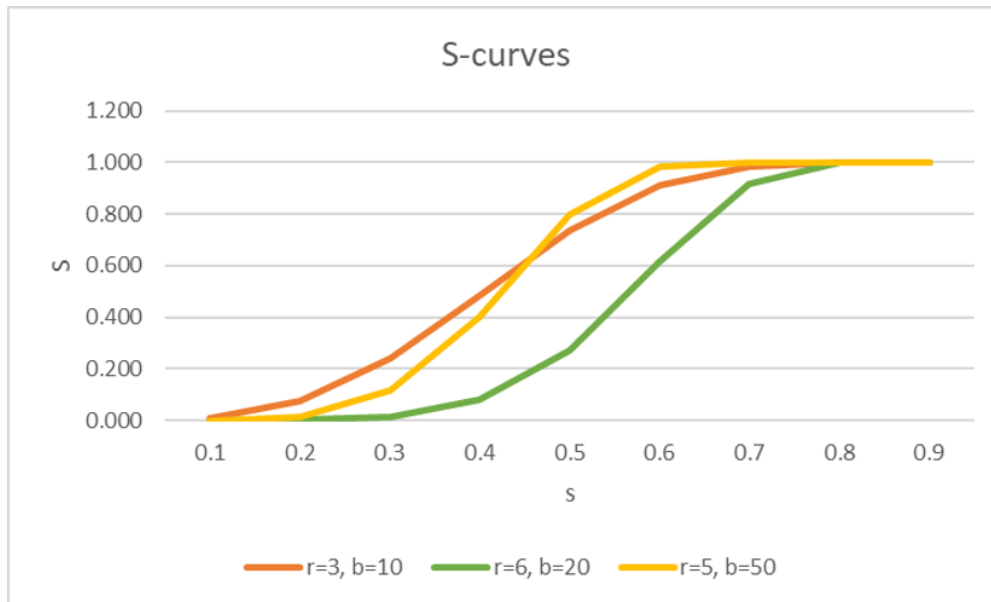| r	b	t
3	10	0.464
6	60	0.607
5	50	0.457

Figure 1: S-curves

Exercise 2

As noted in the text, the probability of a false positive is the probability of a 1 bit raised to the kth power, which is $(1 - e^{-km/n})^k$. k denotes the number of hash functions, m is the number of members in the set S, and n is the number of bits in the array.

If there are 10 billion bits and 2 billion members of S, then $n = 10$ and $m = 2$. Hence, for 3 hash functions, the false positive rate will be $(1 - e^{-(3)2/10})^3 = 0.09184$. And if $k = 4$, the false positive rate is $(1 - e^{-(4)2/10})^4 = 0.09195$.

We can observe in Figure 2 below that the function is minimized when $k = 3$.
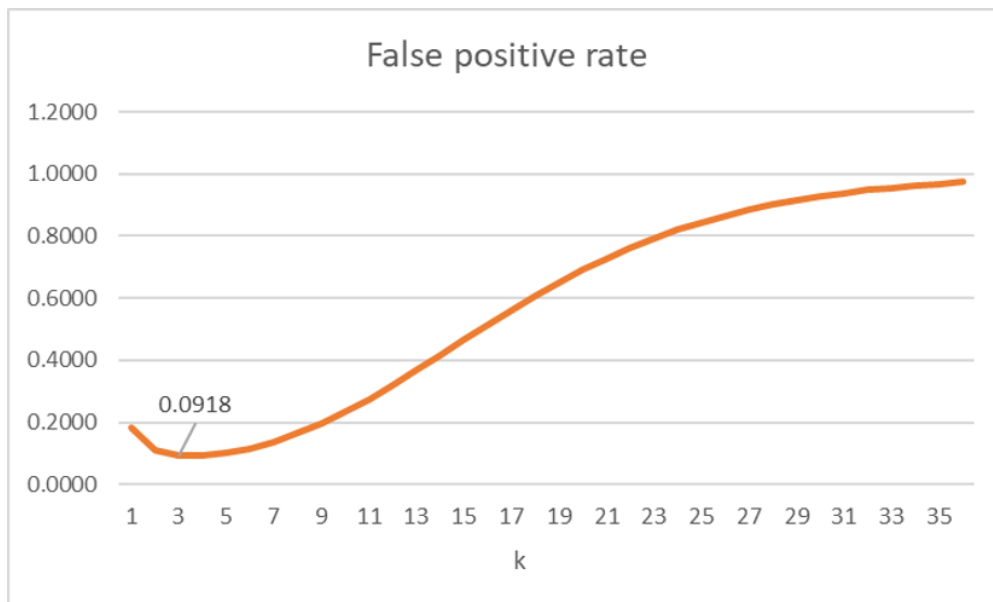

Figure 2: False positive rates

Exercise 3

The PageRank algorithm described in in Section 5.1 and 5.2 (Leskovec, Rajaraman and Ullman) has been implemented in Python. The script is saved in as the source file PageRank.py that is included in the assignment submission.

Our implementation primarily relied on the Python pandas library, particularly the data processing methods for dataframes. Most of the computation required for the algorithm was implemented as column operations on dataframes.

Firstly, the input file was written to a dataframe. The number of edges from each node was then calculated and stored in a separate dataframe. This was then merged back to the input dataframe as a new column, so that each FromNodeId had an associated number of edges. In another new column, 1 was divided by the number of edges, which allowed the non-zero elements of the transition matrix to be represented as a dataframe column.

A separate dataframe is used to represent the probability distribution of the location of a random surfer at each iteration of the algorithm. Probabilities are initialised to 1 divided by the number of nodes. Another merge operation is performed to create a new column in the main dataframe with the current probability distribution, again joining on the FromNodeId column.

By multiplying the transition probability with the current PageRank probability at each iteration, we can compute the individual products required for the dot products in matrix multiplication. The final dot products are computed by summing the individual products, having grouped by the ToNodeId column.

The top ten nodes by the largest PageRank have been included below. This result was obtained by running 60 iterations of the algorithm, at an execution time of 96.762 seconds.

Node Id	PageRank
41909	0.000439
504140	0.000425
597621	0.000416
751384	0.000402
486980	0.000402
384666	0.000400
537039	0.000394
605856	0.000390
32163	0.000384
765334	0.000373

Exercise 4

The following results were calculated using the Excel workbook that was included in the assignment submission. The $h(x)$ function was entered as a formula in the workbook, and the binary representation of this number of obtained using an inbuilt formula. The maximum tail length was observed in the binary numbers, which was then used to determine the estimate of the number of distinct elements.

Scenario 1

Question 1

The tail length for each stream element is calculated in the table below. The resulting estimate of the number of distinct elements in the stream is $2^0 = 1$.

x	$h(x)$	$h(x)$ binary	tail length
3	7	00111	0
1	3	00011	0
4	9	01001	0
6	13	01101	0
5	11	01011	0
9	19	10011	0

Question 2

The tail length for each stream element is calculated in the table below. The resulting estimate of the number of distinct elements in the stream is $2^4 = 16$.

x	$h(x)$	$h(x)$ binary	tail length
3	16	10000	4
1	10	01010	1
4	19	10011	0
6	25	11001	0
5	22	10110	1
9	2	00010	1

Question 3

The tail length for each stream element is calculated in the table below. The resulting estimate of the number of distinct elements in the stream is $2^4 = 16$.

x	$h(x)$	$h(x)$ binary	tail length
3	7	01100	2
1	3	00100	2
4	9	10000	4
6	13	11000	3
5	11	10100	2
9	19	00100	2

Scenario 2

Question 4

The tail length for each stream element is calculated in the table below. The resulting estimate of the number of distinct elements in the stream is $2^2 = 4$.

x	$h(x)$	$h(x)$ binary	tail length
4	26	11010	1
5	0	00000	0
6	6	00110	1
7	12	01100	2
10	30	11110	1
15	28	11100	2

Question 5

The tail length for each stream element is calculated in the table below. The resulting estimate of the number of distinct elements in the stream is $2^0 = 1$.

x	$h(x)$	$h(x)$ binary	tail length
4	13	01101	0
5	15	01111	0
6	17	10001	0
7	19	10011	0
10	25	11001	0
15	3	00011	0

Question 6

The tail length for each stream element is calculated in the table below. The resulting estimate of the number of distinct elements in the stream is $2^3 = 8$.

x	$h(x)$	$h(x)$ binary	tail length
4	8	01000	3
5	10	01010	1
6	12	01100	2
7	14	01110	1
10	20	10100	2
15	30	11110	1

Exercise 5

Part 1

This theory tells us about the locality and the sensitivity function which are very similar to locality hashing. The difference is people can use the other function to produce the pairs and also these functions which can apply to use the space sets or the other spaces and distance measure. For the functions they may have multiple conditions. One of these are having to collect the closest pairs to become the candidate pair because when we use the functions to consider two values are candidate pairs are not. Now we need to hash first two values of two values are equal then those pairs are candidate pairs. And if the two hashing values are not equal then the functions will determinate those two values are not candidate pair, If two hashing values are not equal then the functions are called family function. For example family of hashing function and usually we use form(d1, d2,p1,p2) to expression of the locality function and d1 is smaller than d2. When these distances are bigger or may be equal d2 then x, y is a candidate pair. By the word, peoples don't have any kind of rule to determinate the distance between the d1 and d2 but we should let d1 and d2 be as close as possible, not when the distance between d1 and d2 is smaller then its a chance that p1 and p2 will be closet pair. The second condition is each and every pair must be independent of each other. The third condition is to be function will be highly efficient to determinate the candidate pair and reduce the false positive rate.

The signal of the hashing function may not be fit for these conditions, so the minimum may be able to find the hash in the short time but signal hashing function is incompatible. If someone wants to find the family of locality sensitive function then they can only use the family of the min hashing function and assume that measured distance is Jaccard distance. We use the same way to summarise the min hash function, and then x and y will be candidate pairs, which is how we get the family of sensitive hash function. Hence it is proven that when Jaccard distance x and y will equal to the probability that a min hash function will have hash x and y with same value. Thus, we can use the same way to find the d2. In the locality of sensitive function we have two algorithms AND and OR construction to construct the family of sensitive function and the new can get the new family of function f. Each and every members of the function f is going to be produce by r members of function in F. Another construction is OR, here we can use OR construction to construct F. By these we can use AND construction and we will be reducing every probability.

Part 2

There are many ways to evaluate the similarities between these vectors. This summary is going to introduce the distance such as cosine distance and normal Euclidean distance which have some special ability to dictate locality sensitive families.

LSH Families and Hamming distance

There is a way of hamming distance to indicate the similarities into binary data strings where they have the same length and measuring methods to measure the hamming distance which is comparing how many different bit positions there are in two binary string. Like if someone wants to compare five with three with the use of hamming distance then first we are going to transfer both five and three to binary data, like 1010 and 1100 with due respect. Now we observe there is a difference between 1010 and 1100 so the hamming distance between five and three is respectively 2. We go to know that if two binary data strings is completely same then their hamming distance will be 0 here the hamming distance is going to show does similarity of two data, which is based on the findings, here we are going to introduce hamming distance which is going to measure vector similarity. Here we know that the vector is a set of various types of numerical numbers, different dimensions is measured by different numbers so it is possible to randomly select position of any two vectors that we want to get compared and finding the hamming distance of the two vectors. Such as $fi(x)=xi$ and $fy(y)=yi$. There is a law off hamming distance LSH. where we know that Jaccard distance will only runs on 0 to 1. when some vectors have very low dimensions show the result of similarities are not accurate.

Hyperplanes and the Cosine Distance

As we all know hyperplanes are subspace of ambient space. Here the dimensions of hyperplane will be the dimensions of ambient space like assume that the ambient space have 3D space ,and the hyperplane of this ambient space 2d, here the space is 2D space, the hyperplane will be 1D line, because by the reduced dimension the hyperplane has infinite choice in lower dimensions.