Problem 1 (60 points): A crawl of the Web has returned the 100,000 documents. You have indexed them and represented each document as a vector, using a variant of the tf.idf weighting scheme. You randomly selected ten documents, represented in the following way:

­Table

Description automatically generated

A. (TOTAL: 15 points) You decide to use K-Means to cluster these 10 documents in k=3 clusters. Show the initial seeds of the three clusters and explain how they were selected (1 points) List the clusters you obtain after 5 iterations as lists of document IDs (e.g. {D2, D3}) (5 points) and their centroids (9 points). If you write a program to do generate the clusters– attach the source printout of the program at the end of the exam.

To pick the initial seeds, the cosine-similarities of all documents are computed against

D = [1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]. Then, the 3 initial centroids are picked at index 3,5, and 7 in the sorted documents by cosine-similarities computed above.

After 5 iterations, the centroids are [3, 9, 5] and their clusters are:

* Centroid 3 – Cluster = [3]
* Centroid 9 – Cluster = [8,9]
* Centroid 5 – Cluster = [1, 2, 4, 5, 6, 7, 10]

The program of 1A is in **APPENDIX**  section.

B. (15 points) Evaluate the quality of your clusters with (a) Purity (3 points) (b) Normalized Mutual Information (12 points) if the documents have been classified into the following 3 classes: TRAVEL={D1, D3, D7} HEALTH= {D2, D4, D6} WEATHER={D5, D8, D9, D10} Please provide all details of you computations

*SOLUTION:*

*PURITY*

Given centroid = 3 – cluster = [3]

* Class TRAVEL = {D1, D3, D7} => accuracy = 1
* Class HEALTH = {D2, D4, D6} => accuracy = 0
* Class WEATHER = {D5, D8, D9, D10} => accuracy = 0
* Purity(Centroid = 3) = 1

Given centroid = 9 – Cluster = [8,9]

* Class TRAVEL = {D1, D3, D7} => accuracy = 0
* Class HEALTH = {D2, D4, D6} => accuracy = 0
* Class WEATHER = {D5, D8, D9, D10} => accuracy = 2
* Purity(Centroid = 9) = 2

Given centroid = 5 – cluster = [1,2,4,5,6,7,10]

* Class TRAVEL = {D1, D3, D7} => accuracy = 2
* Class HEALTH = {D2, D4, D6} => accuracy = 3
* Class WEATHER = {D5, D8, D9, D10} => accuracy = 2
* Purity(Centroid = 5) = 3

Purity = (1 + 2 + 3) / (10) = 0.6

*NMI*

Given centroid = 3 – cluster = [3]

I(centroid = 3, Class) =

= P(centroid = 3, TRAVEL) \* log (P(centroid = 3, TRAVEL) / (P(centroid = 3) \* P(TRAVEL))

+ P(centroid = 3, HEALTH) \* log (P(centroid = 3, HEALTH) / (P(centroid = 3) \* P(HEALTH))

+ P(centroid = 3, WEALTH) \* log (P(centroid = 3, WEALTH) / (P(centroid = 3) \* P(WEALTH))

= (1/10) \* log(10 \* 1 / (1 \* 3)) + (0/10) \* log(10\*0/(1\*3)) + (0/10) \* log(10 \*0/(1\*4))

= 0.0523 + 0 + 0 = 0.0523

I(centroid = 9, Class) =

= P(centroid = 9, TRAVEL) \* log (P(centroid = 9, TRAVEL) / (P(centroid = 9) \* P(TRAVEL))

+ P(centroid = 9, HEALTH) \* log (P(centroid = 9, HEALTH) / (P(centroid = 9) \* P(HEALTH))

+ P(centroid = 9, WEALTH) \* log (P(centroid = 9, WEALTH) / (P(centroid = 9) \* P(WEALTH))

= (0/10) \* log(10 \* 0 / (2 \* 3)) + (0/10) \* log(10\*0/(2\*3)) + (2/10) \* log(10 \*2/(2\*4))

= 0 + 0 + log(5/2)/5 = 0.0796

I(centroid = 5, Class) =

= P(centroid = 5, TRAVEL) \* log (P(centroid = 5, TRAVEL) / (P(centroid = 5) \* P(TRAVEL))

+ P(centroid = 5, HEALTH) \* log (P(centroid = 5, HEALTH) / (P(centroid = 5) \* P(HEALTH))

+ P(centroid = 5, WEALTH) \* log (P(centroid = 5, WEALTH) / (P(centroid = 5) \* P(WEALTH))

= (2/10) \* log(10 \* 2 / (7 \* 3)) + (3/10) \* log(10\*3/(7\*3)) + (2/10) \* log(10 \*2/(7\*4))

= -0.0042 + 0.0464 – 0.0292

= 0.013

**I(Cluster, Class) = 0.0523 + 0.0796 + 0.013 = 0.1449**

P(TRAVEL) = 3/10 = 0.3

P(HEALTH) = 3/10 = 0.3

P(WEALTH) = 4/10= 0.4

**H(Class) = (-0.3) \* log(0.3) \*2 – 0.4\*log(0.4) = 0.473**

P(Centroid = 3) = 1/10 = 0.1

P(Centroid = 9) = 2/10 = 0.2

P(Centroid = 5) = 7/10 = 0.7

**H(Cluster) = -0.1\*log(0.1) – 0.2\*log(0.2) – 0.7\*log(0.7) = 0.3483**

**NMI(Cluster, Class) = 0.1449 \* 2/ (0.483 + 0.3483) = 0.3486**

(20 points) Using the same collection of documents, cluster them using Complete-link agglomerative clustering. Show the clusters as lists of document IDs (e.g. {D3, D5, D7}) (12 points). How did you decide to stop the clustering? (3 points). How did you decide to cut the dendogram to form the clusters? (5 points). If you decide to write a program to resolve the agglomerative clustering problem – attach the source printout of the program at the end of the exam. SOLUTION

Similarity is based on cosine similarity function defined in 1A. The computation of cosine similarity is not displayed.

Initialize the indicator array

I = [1, 1, 1, 1, 1, 1, 1, 1, 1, 1]

Table 1 C= 0

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | D1 | D2 | D3 | D4 | D5 | D6 | D7 | D8 | D9 | D10 |
| D1 | 1 | 0.469 | 0.409 | 1.000 | 0.648 | 0.725 | 0.711 | 0.652 | 0.535 | 1.0000 |
| D2 | S-1.2 | 1 | 0.472 | 0.469 | 0.521 | 0.585 | 0.699 | 0.482 | 0.402 | 0.469 |
| D3 | S-1,3 | S-2,3 | 1 | 0.409 | 0.414 | 0.521 | 0.590 | 0.382 | 0.554 | 0.409 |
| D4 | S-1,4 | S-2,4 | S-3,4 | 1 | 0.648 | 0.725 | 0.711 | 0.652 | 0.535 | 1.000 |
| D5 | S-1,5 | S-2,5 | S-3,5 | S-4,5 | 1 | 0.550 | 0.687 | 0.498 | 0.775 | 0.648 |
| D6 | S-1,6 | S-2,6 | S-3,6 | S-4,6 | S-5,6 | 1 | 0.791 | 0.805 | 0.632 | 0.725 |
| D7 | S-1,7 | S-2,7 | S-3,7 | S-4,7 | S-5,7 | S-6,7 | 1 | 0.690 | 0.772 | 0.711 |
| D8 | S-1,8 | S-2,8 | S-3,8 | S-4,8 | S-5,8 | S-6,8 | S-7,8 | 1 | 0.678 | 0.652 |
| D9 | S-1,9 | S-2,9 | S-3,9 | S-4,9 | S-5,9 | S-6,9 | S-7,9 | S-8,9 | 1 | 0.535 |
| D10 | S-1,10 | S-2,10 | S-3,10 | S-4,10 | S-5,10 | S-6,10 | S-7,10 | S-8,10 | S-9,10 | 1 |

Look at the above C = 0, the similarity between D3 and D8 is smallest (S = 0.382)

Table 2 C = 1

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | D1 | D2 | D3-8 | D4 | D5 | D6 | D7 | D9 | D10 |
| D1 | 1 | 0.469 | 0.409 | 1.000 | 0.648 | 0.725 | 0.711 | 0.535 | 1.0000 |
| D2 | S-1.2 | 1 | 0.472 | 0.469 | 0.521 | 0.585 | 0.699 | 0.402 | 0.469 |
| D3-8 | S-1,3 | S-2,3 | 1 | 0.409 | 0.414 | 0.521 | 0.590 | 0.554 | 0.409 |
| D4 | S-1,4 | S-2,4 | S-3,4 | 1 | 0.648 | 0.725 | 0.711 | 0.535 | 1.000 |
| D5 | S-1,5 | S-2,5 | S-3,5 | S-4,5 | 1 | 0.550 | 0.687 | 0.775 | 0.648 |
| D6 | S-1,6 | S-2,6 | S-3,6 | S-4,6 | S-5,6 | 1 | 0.791 | 0.632 | 0.725 |
| D7 | S-1,7 | S-2,7 | S-3,7 | S-4,7 | S-5,7 | S-6,7 | 1 | 0.772 | 0.711 |
| D9 | S-1,9 | S-2,9 | S-3,9 | S-4,9 | S-5,9 | S-6,9 | S-7,9 | 1 | 0.535 |
| D10 | S-1,10 | S-2,10 | S-3,10 | S-4,10 | S-5,10 | S-6,10 | S-7,10 | S-9,10 | 1 |

Evaluate using Complete-link:

* S(D3-8 – D1) = sim({D3, D8}, {D1}) = min(S13, S18) = S13 = 0.409
* S(D3-8 – D2) = sim({D3, D8}, {D2}) = min(S23, S28) = S23 = 0.472
* S(D3-8 – D4) = sim({D3, D8}, {D4}) = min(S34, S48) = S34 = 0.409
* S(D3-8 – D5) = sim({D3, D8}, {D5}) = min(S35, S58) = S35 = 0.414
* S(D3-8 – D6) = sim({D3, D8}, {D6}) = min(S36, S68) = S36 = 0.521
* S(D3-8 – D7) = sim({D3, D8}, {D7}) = min(S37, S78) = S37 = 0.590
* S(D3-8 – D9) = sim({D3, D8}, {D9}) = min(S39, S89) = S39 = 0.554
* S(D3-8 – D10) = sim({D3, D8}, {D10}) = min(S310, S810) = S310 = 0.409

Update the indicator array:

I = [1, 1, 1, 1, 1, 1, 1, 0, 1, 1]

Look at the above C = 1, the similarity between D2 and D9 is smallest (S = 0.402)

Table 3 C = 2

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | D1 | D2-9 | D3-8 | D4 | D5 | D6 | D7 | D10 |
| D1 | 1 | 0.469 | 0.409 | 1.000 | 0.648 | 0.725 | 0.711 | 1.0000 |
| D2-9 | S-1.2 | 1 | 0.382 | 0.469 | 0.521 | 0.585 | 0.699 | 0.469 |
| D3-8 | S-1,3 | S-3,8 | 1 | 0.409 | 0.414 | 0.521 | 0.590 | 0.409 |
| D4 | S-1,4 | S-2,4 | S-3,4 | 1 | 0.648 | 0.725 | 0.711 | 1.000 |
| D5 | S-1,5 | S-2,5 | S-3,5 | S-4,5 | 1 | 0.550 | 0.687 | 0.648 |
| D6 | S-1,6 | S-2,6 | S-3,6 | S-4,6 | S-5,6 | 1 | 0.791 | 0.725 |
| D7 | S-1,7 | S-2,7 | S-3,7 | S-4,7 | S-5,7 | S-6,7 | 1 | 0.711 |
| D10 | S-1,10 | S-2,10 | S-3,10 | S-4,10 | S-5,10 | S-6,10 | S-7,10 | 1 |

Evaluate using Complete-link:

* S(D2-9 – D1) = sim({D2, D9}, {D1}) = min(S12, S19) = S12 = 0.469
* S(D2-9 – D3-8) = sim({D3, D8}, {D2-9}) = min(S23, S28 , S38, S39) = S38 = 0.382
* S(D2-9 – D4) = sim({D2, D9}, {D4}) = min(S24, S49) = S24 = 0.469
* S(D2-9 – D5) = sim({D2, D9}, {D5}) = min(S25, S59) = S25 = 0.521
* S(D2-9 – D6) = sim({D2, D9}, {D6}) = min(S26, S69) = S26 = 0.585
* S(D2-9 – D7) = sim({D2, D9}, {D7}) = min(S27, S79) = S27 = 0.699
* S(D2-9 – D10) = sim({D2, D9}, {D10}) = min(S210, S910) = S210 = 0.469

Update the indicator array:

I = [1, 1, 1, 1, 1, 1, 1, 0, 0, 1]

Look at the above C = 2, the similarity between D2-9 and D3-8 is smallest (S = 0.382)

Table 4 C = 3

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | D1 | D2-3-8-9 | D4 | D5 | D6 | D7 | D10 |
| D1 | 1 | 0.409 | 1.000 | 0.648 | 0.725 | 0.711 | 1.0000 |
| D2-3-8-9 | S-1.3 | 1 | 0.409 | 0.414 | 0.521 | 0.59 | 0.409 |
| D4 | S-1,4 | S-3,4 | 1 | 0.648 | 0.725 | 0.711 | 1.000 |
| D5 | S-1,5 | S-3,5 | S-4,5 | 1 | 0.550 | 0.687 | 0.648 |
| D6 | S-1,6 | S-3,6 | S-4,6 | S-5,6 | 1 | 0.791 | 0.725 |
| D7 | S-1,7 | S-3,7 | S-4,7 | S-5,7 | S-6,7 | 1 | 0.711 |
| D10 | S-1,10 | S-3,10 | S-4,10 | S-5,10 | S-6,10 | S-7,10 | 1 |

Evaluate using Complete-link:

* S(D2 – D1) = sim({D2, D3, D8, D9}, {D1}) = min(S12, S13, S18, S19) = S13 = 0.409
* S(D2 – D4) = sim({D2, D3, D8, D9}, {D4}) = min(S24, S34, S48, S49) = S34 = 0.409
* S(D2-9 – D5) = sim({D2, D3, D8, D9}, {D5}) = min(S25, S35, S58, S59) = S35 = 0.414
* S(D2-9 – D6) = sim({D2, D3, D8, D9}, {D6}) = min(S26, S36, S68, S69) = S36 = 0.521
* S(D2-9 – D7) = sim({D2, D3, D8, D9}, {D7}) = min(S27, S37, S78, S79) = S37 = 0.59
* S(D2-9 – D10) = sim({D2, D3, D8, D9}, {D10}) = min(S210, S310, S810, S910) = S310 = 0.409

Update the indicator array:

I = [1, 1, 0, 1, 1, 1, 1, 0, 0, 1]

Look at the above C = 3 the similarity between D2 and D1 is smallest (S = 0.409)

Table 5 C = 4

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | D1-2-3-8-9 | D4 | D5 | D6 | D7 | D10 |
| D1-2-3-8-9 | 1 | 0.409 | 0.414 | 0.521 | 0.59 | 0.409 |
| D4 | S-3,4 | 1 | 0.648 | 0.725 | 0.711 | 1.000 |
| D5 | S-3,5 | S-4,5 | 1 | 0.550 | 0.687 | 0.648 |
| D6 | S-3,6 | S-4,6 | S-5,6 | 1 | 0.791 | 0.725 |
| D7 | S-3,7 | S-4,7 | S-5,7 | S-6,7 | 1 | 0.711 |
| D10 | S-3,10 | S-4,10 | S-5,10 | S-6,10 | S-7,10 | 1 |

Evaluate using Complete-link:

* S(D2-3-8-9-1 – D4) = sim({D2, D3,D8, D9, D1}, {D4}) = S34 = 0.409
* S(D2-3-8-9-1 – D5) = sim({D2, D3,D8, D9, D1}, {D5}) = S35 = 0.414
* S(D2-3-8-9-1 – D6) = sim({D2, D3,D8, D9, D1}, {D6}) = S36 = 0.521
* S(D2-3-8-9-1 – D7) = sim({D2, D3,D8, D9, D1}, {D7}) = S37 = 0.59
* S(D2-3-8-9-1 – D10) = sim({D2, D3,D8, D9, D1}, {D10}) = S210 = 0.409

Update the indicator array:

I = [0, 1, 0, 1, 1, 1, 1, 0, 0, 1]

Look at the above C = 4 the similarity between D2-9 and D4 is smallest (S = 0.409)

Table 5 C = 5

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | D1-2-3-4-8-9 | D5 | D6 | D7 | D10 |
| D1-2-3-4-8-9 | 1 | 0.414 | 0.521 | 0.59 | 0.409 |
| D5 | S-3,5 | 1 | 0.550 | 0.687 | 0.648 |
| D6 | S-3,6 | S-5,6 | 1 | 0.791 | 0.725 |
| D7 | S-3,7 | S-5,7 | S-6,7 | 1 | 0.711 |
| D10 | S-3,10 | S-5,10 | S-6,10 | S-7,10 | 1 |

Evaluate using Complete-link:

* S(D1-2-3-4-8-9 - D5) = sim({D2, D3, D4, D8, D9, D1}, {D5}) = S35 = 0.414
* S(D1-2-3-4-8-9 – D6) = sim({D2, D3, D4, D8, D9, D1}, {D6}) = S36 = 0.521
* S(D1-2-3-4-8-9 – D7) = sim({D2, D3, D4, D8, D9, D1}, {D7}) = S37 = 0.59
* S(D1-2-3-4-8-9 – D10) = sim({D2, D3, D4, D8, D9, D1}, {D10}) = S210 = 0.409

Update the indicator array:

I = [0, 1, 0, 0, 1, 1, 1, 0, 0, 1]

Look at the above C = 5 the similarity between D2-9 and D10 is smallest (S = 0.409)

Table 5 C = 6

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | D1-2-3-4-8-9-10 | D5 | D6 | D7 |
| D1-2-3-4-8-9-10 | 1 | 0.521 | 0.585 | 0.59 |
| D5 | S-2,5 | 1 | 0.550 | 0.687 |
| D6 | S-2,6 | S-5,6 | 1 | 0.791 |
| D7 | S-2,7 | S-5,7 | S-6,7 | 1 |

Evaluate using Complete-link:

* S(D1-2-3-4-8-9-10 – D5) = sim({D1, D2, D3, D4, D8, D9, D10}, {D5}) = S35 = 0.414
* S(D1-2-3-4-8-9-10 – D6) = sim({D1, D2, D3, D4, D8, D9, D10}, {D6}) = S36 = 0.521
* S(D1-2-3-4-8-9-10 – D7) = sim({D1, D2, D3, D4, D8, D9, D10}, {D7}) = S37 = 0.59
* S(D1-2-3-4-8-9-10 – D10) = sim({D1, D2, D3, D4, D8, D9, D10}, {D10}) = S210 = 0.409

Update the indicator array:

I = [0, 1, 0, 0, 1, 1, 1, 0, 0, 0]

Look at the above C = 6 the similarity between D2-9 and D5 is smallest (S = 0.521)

Table 5 C = 7

|  |  |  |  |
| --- | --- | --- | --- |
|  | D1-2-3-4-5-8-9-10 | D6 | D7 |
| D1-2-3-4-5-8-9-10 | 1 | 0.585 | 0.59 |
| D6 | S-2,6 | 1 | 0.791 |
| D7 | S-2,7 | S-6,7 | 1 |

Evaluate using Complete-link:

* S(D1-2-3-4-5-8-9-10 – D5) = sim({D1, D2, D3, D4, D5, D8, D9, D10}, {D5}) = min(S25, S59) = S25 = 0.521
* S(D1-2-3-4-5-8-9-10 – D6) = sim({D1, D2, D3, D4, D5, D8, D9, D10}, {D6}) = min(S26, S69) = S26 = 0.585
* S(D1-2-3-4-5-8-9-10 – D7) = sim({D1, D2, D3, D4, D5, D8, D9, D10}, {D7}) = min(S27, S79) = S27 = 0.699

Update the indicator array:

I = [0, 1, 0, 0, 0, 1, 1, 0, 0, 0]

Look at the above C = 7 the similarity between D2-9 and D6 is smallest (S = 0.585)

Table 5 C = 8

|  |  |  |
| --- | --- | --- |
|  | D1-2-3-4-5-6-8-9-10 | D7 |
| D1-2-3-4-5-6-8-9-10 | 1 | 0.59 |
| D7 | S-2,7 | 1 |

Evaluate using Complete-link:

* S(D1-2-3-4-5-6-8-9-10 – D5) = sim({D1, D2, D3, D4, D5, D6, D8, D9, D10}, {D5}) = min(S25, S59) = S25 = 0.521
* S(D1-2-3-4-5-6-8-9-10 – D7) = sim({D1, D2, D3, D4, D5, D6, D8, D9, D10}, {D7}) = min(S27, S79) = S27 = 0.699

Update the indicator array:

I = [0, 1, 0, 0, 0, 0, 1, 0, 0, 0]

Look at the above C = 8 the similarity between D2-9 and D7 is smallest (S = 0.59)

Finally, we have a single cluster.

I stopped the clustering when there is a single cluster after a series of merging.

Update the indicator array:

I = [0, 1, 0, 0, 0, 0, 0, 0, 0, 0]

Here, I specify that each centroid has a maximum cluster of 5. Then the denogram is cut as below.

Diagram, schematic

Description automatically generatedeach of the Web D. (5 points) Compute the page ranks of pages from the graph crawled to find documents D1, D2, …, D10. Consider the following web graph that connects the 10 Web pages:

A picture containing graphical user interface

Description automatically generated

How do you initialize the PageRank values? (1 point). Provide the values of the PageRanks for each of the 10 web pages after iteration 1, 2, 3 and 4. (4 points). If you decide to write a program to resolve the problem – attach the source printout of the program at the end of the exam.

Given 10 pages, then the initial PageRank values are 1/10.

The damping factor is set to 0.85.

*SOLUTION*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Node | Source Nodes | Count | Iteration = 0 | Iteration = 1 | Iteration = 2 | Iteration = 3 | Iteration = 4 |
| D1 | D3, D7, D8, D10 | 4 | 1/10 | 0.15 + 0.85(0.1/3 + 0.1/3 + 0.1/3 + 0.1/3) = 0.263 | 0.15 + 0.85(0.252/3 + 0.199/3 + 0.178/3 + 0.256/3) = 0.401 | 0.15 + 0.85(0.401/3 + 0.269/3 + 0.227/3 + 0.396/3) = 0.516 | 0.15 + 0.85(0.53/3 + 0.331/3 + 0.274/3 + 0.515/3) = 0.6175 |
| D2 | D1, D4, D6, D9 | 4 | 1/10 | 0.15 + 0.85(0.1/5 + 0.1/3 + 0.1/3 + 0.1/3) = 0.252 | 0.15 + 0.85(0.263/5 + 0.178/3 + 0.273/3 + 0.273/3) = 0.399 | 0.15 + 0.85(0.401/5 + 0.221/3 + 0.437/3 + 0.42/3) = 0.524 | 0.15 + 0.85(0.516/5 + 0.264/3 + 0.575/3 + 0.538/3) = 0.628 |
| D3 | D1, D6, D7, D10 | 4 | 1/10 | 0.15 + 0.85(0.1/5 + 0.1/3 + 0.1/3 + 0.1/3) = 0.252 | 0.15 + 0.85(0.263/5 + 0.273/3 + 0.199/3 + 0.256/3) = 0.401 | 0.15 + 0.85(0.401/5 + 0.437/3 + 0.269/3 + 0.396/3) = 0.53 | 0.15 + 0.85(0.516/5 + 0.575/3 + 0.331/3 + 0.515/3) = 0.64 |
| D4 | D2 | 1 | 1/10 | 0.15 + 0.85(0.1/3) = 0.178 | 0.15 + 0.85(0.252/3) = 0.221 | 0.15 + 0.85(0.399/3) = 0.263 | 0.15 + 0.85(0.524/3) = 0.298 |
| D5 | D1, D8, D9 | 3 | 1/10 | 0.15 + 0.85(0.1/5 + 0.1/3 + 0.1/3) = 0.224 | 0.15 + 0.85(0.263/5 + 0.178/3 + 0.273/3) = 0.322 | 0.15 + 0.85(0.401/5 + 0.227/3 + 0.42/3) = 0.401 | 0.15 + 0.85(0.516/5 + 0.274/3 + 0.538/3) = 0.468 |
| D6 | D1, D3, D4, D5, D10 | 5 | 1/10 | 0.15 + 0.85(0.1/5 + 0.1/3 + 0.1/3 + 0.1/4 + 0.1/3) = 0.273 | 0.15 + 0.85(0.263/5 + 0.252/3 + 0.178/3 + 0.224/4 + 0.256/3) = 0.437 | 0.15 + 0.85(0.401/5 + 0.401/3 + 0.221/3 + 0.322/4 + 0.396/3) = 0.575 | 0.15 + 0.85(0.516/5 + 0.53/3 + 0.263/3 + 0.401/4 + 0.515/3) = 0.694 |
| D7 | D2, D5 | 2 | 1/10 | 0.15 + 0.85(0.1/3 + 0.1/4) = 0.199 | 0.15 + 0.85(0.252/3 + 0.224/4) = 0.269 | 0.15 + 0.85(0.399/3 + 0.322/4) = 0.331 | 0.15 + 0.85(0.524 /3 + 0.401/4) = 0.384 |
| D8 | D6 | 1 | 1/10 | 0.15 + 0.85(0.1/3) = 0.178 | 0.15 + 0.85(0.273/3) = 0.227 | 0.15 + 0.85(0.437/3) = 0.274 | 0.15 + 0.85(0.575/3) = 0.313 |
| D9 | D1, D2, D4, D5, D7 | 5 | 1/10 | 0.15 + 0.85(0.1/5 + 0.1/3 + 0.1/3 + 0.1/4 + 0.1/3) = 0.273 | 0.15 + 0.85(0.263/5 + 0.252/3 + 0.178/3 + 0.224/4 + 0.199/3) = 0.42 | 0.15 + 0.85(0.401/5 + 0.399/3 + 0.221 /3 + 0.322/4 + 0.269/3) = 0.538 | 0.15 + 0.85(0.516/5 + 0.524/3 + 0.263/3 + 0.401/4 + 0.331/3) = 0.64 |
| D10 | D3, D5, D8, D9 | 4 | 1/10 | 0.15 + 0.85(0.1/3 + 0.1/4 + 0.1/3 + 0.1/3) = 0.256 | 0.15 + 0.85(0.252/3 + 0.224/4 + 0.178/3 + 0.272/3) = 0.396 | 0.15 + 0.85(0.401/3 + 0.322/4 + 0.227/3 + 0.42/3) = 0.515 | 0.15 + 0.85(0.53/3 + 0.401/4 + 0.274/3 + 0.538/3) = 0.615 |

E. (5 points) Use the HITS algorithm to compute the hub and authority score of each Web page from the graph used in D. How do you initialize the hub and authority values of each web page? (1 point). Provide the hub and authority values of each web page after iteration 1, 2, 3 and 4. (4 points). If you decide to write a program to resolve this problem – attach the source printout of the program at the end of the exam.

SOLUTION

Initially, each web page has is hub and authority value scores equal to 1.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Node | Linked Nodes | Source Nodes | Iteration 0 | | Iteration 1 | | Iteration 2 | | Iteration 3 | | Iteration 4 | |
| Hub | Auth-  ority | Hub | Auth-  ority | Hub | Auth-  ority | Hub | Auth-  ority | Hub | Auth-  Ority |
| D1 | D2, D3, D5, D6, D9 | D3, D7, D8, D10 | 1 | 1 | 5 | 4 | 3+3+4  +3+3  =16 | 4+2+1+  4=11 | 8+11+  12+9+  10=50 | 15+7  +5+13  =40 | 30+36  +41+  31+31  =169 | 48+25  +16+  44=133 |
| D2 | D4, D7, D9 | D1, D4, D6, D9 | 1 | 1 | 3 | 4 | 3+2+3  =8 | 4+1+5  +5=15 | 9+11+10  =30 | 11+4  +16+14  =45 | 27+41  +31  =99 | 40+15  +53+  47=155 |
| D3 | D1, D6, D10 | D1, D6, D7, D10 | 1 | 1 | 3 | 4 | 5+3+3  =11 | 4+5+2  +4=15 | 16+9+11  =36 | 11+16  +7+14  =48 | 50+31  +36  =117 | 40+53  +25+44  =162 |
| D4 | D2, D6, D9 | D2 | 1 | 1 | 3 | 1 | 3+3+3  =9 | 4 | 8+9+10  =27 | 15 | 30+31  +31=  92 | 45 |
| D5 | D6, D7, D9, D10 | D1, D8, D9 | 1 | 1 | 4 | 3 | 3+3+3  +3=12 | 4+1+5  =10 | 9+11+  10+11  =41 | 11+5  +14=30 | 31+41  +31+  36  =139 | 40+16  +47=103 |
| D6 | D2, D3, D8 | D1, D3, D4, D5, D10 | 1 | 1 | 3 | 5 | 3+3+3  =9 | 4+4+1+  3+4=16 | 8+11+  12=31 | 11+15  +4+10 +13=53 | 30+36  +39  =105 | 40+48  +15+30  +44  =177 |
| D7 | D1, D3, D9 | D2, D5 | 1 | 1 | 3 | 2 | 5+3+3  =11 | 4+3=7 | 16+15+  10=41 | 15+10  =25 | 50+36  +31=  147 | 45+30  =75 |
| D8 | D1, D5, D10 | D6 | 1 | 1 | 3 | 1 | 5+4+3  =12 | 5 | 16+12  +11=39 | 16 | 50+41  +36= 127 | 53 |
| D9 | D2, D5, D10 | D1, D2, D4, D5, D7 | 1 | 1 | 3 | 5 | 3+4+3  =10 | 4+4+1+  3+2=14 | 8+12+  11=31 | 11+15  +4+10  +7=47 | 30+41  +36=  107 | 40+45  +15+30  +25  =155 |
| D10 | D1, D3, D6 | D3, D5, D8, D9 | 1 | 1 | 3 | 4 | 5+3+3  =11 | 4+3+1  +5=13 | 16+11+9  =36 | 15+10  +5+14  =44 | 50+36  +31=  117 | 48+30  +16+47  =141 |

Problem 2 (25 points) :

1. (10 points) As you are crawling the Web for your search engine, you are using the Mercator scheme and have available 3 front queues and 6 back queues. At the beginning of the crawl, you have visited the following URLs:

[1] [www.bbc.com](http://www.bbc.com)

[2] money.cnn.com

[3] [www.cnn.edu/travel/](http://www.cnn.edu/travel/)

[4] [www.cs.stanford.edu/](http://www.cs.stanford.edu/)

[5] [www.cs.cornell.edu/](http://www.cs.cornell.edu/)

[6] [www.cs.stanford.edu/~manning](http://www.cs.stanford.edu/~manning)

[7] [www.cs.stanford.edu/~feifeili](http://www.cs.stanford.edu/~feifeili)

[8] <http://machinelearning.cis.cornell.edu/pages/people.php>

[9] <http://www.cs.cornell.edu/home/kleinber/>

[10] <http://www.macys.com/>

[11] <http://www.newbalance.com/>

[12] <http://www.nike.com/>

[13] <http://www.amazon.com>

Show how you arrange the URLs in the front queues (2 points) and then how you pass them on the back queues (4 points). Also show the content of the table of hosts to the back queues (4 points)

SOLUTION:

Assume that the crawling visits sites linearly and the front-queue assigns incremental priority to visited sites as it enters the front queue. Hence, the 3 front queues have:

* Queue 1 = {1, 4, 7, 10, 13}
* Queue 2 = {2, 5, 8, 11}
* Queue 3 = {3, 6, 9, 12}

The biased front queue selector pulls URL based on round robin biased to queues of higher priority.

Given 6 back queries, each back query only contains URLs from a single host. Hence, the URLs pulled front queues are routed to back queries when any of them empty.

Table of hosts to back queries:

|  |  |
| --- | --- |
| Host Name | Back queue |
| [www.bbc.com](http://www.bbc.com) | 1 |
| cnn.com | 2 |
| [www.cs.stanford.edu/](http://www.cs.stanford.edu/) | 3 |
| [www.cs.cornell.edu/](http://www.cs.cornell.edu/) | 4 |
| <http://machinelearning.cis.cornell.edu/pages/people.php> | 5 |
| <http://www.macys.com/> | 6 |
| <http://www.newbalance.com/> | 1 |
| <http://www.nike.com/> | 2 |
| <http://www.amazon.com> | 3 |

Iteration 1: first 10 URLs are pulled into back queries. Then, all URLs from every back query is crallwed.

* Query 1 = {1} -> crawled 1st
* Query 2 = {2, 3, 6, 7} -> crawled 2nd
* Query 3 = {4} -> crawled 3rd
* Query 4 = {5, 9} -> crawled 4th
* Query 5 = {8} -> crawled 5th
* Query 6 = {10} -> crawled 6th

Iteration 2:

* Query 1 = {11} -> crawled 7th
* Query 2 = {12} -> crawled 8th
* Query 3 = {13} ->crawled 9th
* Query 4 = {5, 9} -> crawled 4th
* Query 5 = {8} -> crawled 5th
* Query 6 = {10} -> crawled 6th

1. (15 points) To detect duplication on the Web, you are requested to generate a pair of sketch vectors of size 25 from the shingles you create for the content found in the following two web pages:

W1: The class will cover web crawling.

W2: Crawling shall be presented in the class.

Use the Jaccard coefficient to compute the similarity between the sketch vectors (3 points); then use row hashing with the following two hash functions: h1=(x +2)mod 4; h2=(x+3) mod 6; (10 points) Were the web pages W1 and W2 found dissimilar by either of these methods? (2 points)

**Problem 3**

(15 points) : Consider the that after you used the query Q0= "medical disorder" you have obtained the following local document set:

Table

Description automatically generated

Use pseudo-relevance feedback to expand the query by modifying it with 2 additional keywords when using the local analysis method based on association clusters. Show how you found the terms and the stems (and their variants) of the local document set (5 points). Show how you expand the query when using association clusters without normalization (5 points) and explain how you selected the 2 new keywords to be added to the initial query (5 points). Please provide all details showing how you determined the association clusters.

*SOLUTION*

* To find terms, removing punctuations, removing stopwords, and tokenization are performed to generate terms.
* Then, to find stems, either stemming or lemmatizing could be performed to get stems and their variants.
* Also, word variants could be found from WordNet.

N (number of documents) = 4

V (number of terms) = 32

Terms in the local vocabulary: abuse, allergy, among, celery, children, create, disorder, dose, emerge, england, expectancy, fiber, gap, have, health, healthy, issue, life, medicine, medical, mental, mostly, people, poor, provide, rich, risk, sort, trouble, young, vitamin.

The vocabularies of stems (cooccurrence larger than 1):

* V(health) = {health, healthy}

S (number of stems) = 31

Stems in the local vocabulary: abuse, allergy, among, celery, children, create, disorder, dose, emerge, england, expectancy, fiber, gap, have, health, issue, life, medical, medicine, mental, mostly, people, poor, provide, rich, risk, sort, trouble, young, vitamin.

Number of new keywords = 2

To find 2 new keywords, computing the non-normalized association cluster against stems and pick the largest two non-normalized correlation factors:

* Medical
  + C-Medical-x-Health: 1\*1 + 1\*1 + 1\*1 + 1\*2 = 5
  + C-Medical-x-(other stems) < 5 (because other stems occur only one)
* Disorder
  + C-Disorder-x-Health: 1\*1 + 1\*0 + 1\*1 + 1\*2 = 4
  + C-Disorder-x-(other stems) < 4 (because other stems occur only one)

Hence, the expanded query is “medical disorder health” that “health” is used only once because of duplicate.

**APPENDIX**

Code used to resolve the problems in the final exam.

***1A***

# import dependencies

from collections import defaultdict

import numpy as np

def cosine(x, y):

x = np.array(x)

y = np.array(y)

top = np.dot(x,y)

bottom = np.sqrt(np.sum(np.square(x))) \*\

np.sqrt(np.sum(np.square(y)))

return top / bottom

def seed(docs, k):

# define diagonal

D = np.ones(len(list(docs.values())[0]))

# compute cosines

cosines = {k:cosine(D, doc) for k,doc in docs.items()}

# sort cosines

cosines = sorted(cosines.items(), key = lambda x: x[-1])

# select initial clusters at index = [3, 5, 7]

return cosines[3][0], cosines[5][0], cosines[7][0]

#return np.random.randint(low = 1, high = len(docs), size = k)

def kmean(docs, centroids, epochs):

def \_assign\_to\_centroids(doc, ctrs):

# compute cosine angles for every document against centroids

angles = []

for cls in ctrs:

angles.append(cosine(doc, docs[cls]))

# best centroid

centroid = np.argmax(angles)

return ctrs[centroid], angles[centroid]

def \_update\_centroids(clusters, angles):

\_centroids = []

for id, angls in angles.items():

# find new centroid as average of clusters

avg\_ang = np.average(angls)

new\_id = np.argmin(angls - avg\_ang)

\_centroids.append(clusters[id][new\_id])

return \_centroids

# execute kmeans

for e in range(epochs):

clusters = {k:[] for k in centroids}

angles = {k:[] for k in centroids}

# assign docs to clusters

for id, doc in docs.items():

# find closet centroid to each document

cls, angle = \_assign\_to\_centroids(doc, centroids)

# assign docs

clusters[cls].append(id)

angles[cls].append(angle)

# update clusters

centroids = \_update\_centroids(clusters, angles)

return centroids, list(clusters.values())

def main():  
# define parameters k=3  
epochs = 5

# define docs

docs = [

[0.22, 0.31, 0.66, 0.45, 0.48, 0.11, 0.33, 0.89, 0.31,

0.66, 0.11, 0.89, 0.0],

[0.0, 0.75, 0.0, 0.11, 0.0, 0.67, 0.33, 0.0, 0.22, 0.33,

0.0, 0.5, 0.67],

[0.5, 0.0, 0.11, 0.0, 0.0, 0.66, 0.0, 0.11, 0.0, 0.66,

0.23, 0.0, 0.11],

[0.22, 0.31, 0.66, 0.45, 0.48, 0.11, 0.33, 0.89, 0.31,

0.66, 0.11, 0.89, 0.0],

[0.11, 0.31, 0.0, 0.22, 0.11, 0.0, 0.0, 0.5, 0.0, 0.33,

0.0, 0.0, 0.33],

[0.5, 0.22, 0.11, 0.0, 0.0, 0.15, 0.0, 0.33, 0.0, 0.22,

0.0, 0.66, 0.22],

[0.33, 0.5, 0.5, 0.0, 0.0, 0.66, 0.0, 0.75, 0.0, 0.22,

0.0, 0.45, 0.45],

[0.75, 0.14, 0.5, 0.22, 0.48, 0.11, 0.0, 0.25, 0.0, 0.0,

0.0, 0.66, 0.66],

[0.33, 0.0, 0.22, 0.0, 0.0, 0.11, 0.0, 0.5, 0.0, 0.22,

0.0, 0.0, 0.5],

[0.22, 0.31, 0.66, 0.45, 0.48, 0.11, 0.33, 0.89, 0.31,

0.66, 0.11, 0.89, 0.0]

]

docs = {x:y for x, y in zip(

list(range(1, 11)), docs)}

# initialize seed

centroids = seed(docs, k)

# perform clustering

centroids, clusters = kmean(docs = docs, centroids = centroids,

epochs = epochs)

print('After 5 iterations, the centroids are {} and their clusters

are {}'.format(centroids, clusters))

return None

if \_\_name\_\_ == '\_\_main\_\_':

main()