Student Name: Ekal Golas Student NetID: exg140230

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

University of Texas at Dallas

Department of Computer Science

CS6322 – Information Retrieval

Spring 2016

Instructor: Dr. Sanda Harabagiu

Take-Home Final Exam

Issued: April 20th 2016

Due: April 27th 2016 –in class

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Problem 1 (60 points) :**

A crawl of the Web has returned the following documents:

D1

The Mustang, built in Flat Rock, Mich., is still a novelty in Europe,

where it went on sale for the first time last summer.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

D2

Food allergies have been on the rise in recent years and are currently estimated to affect up to eight percent of children worldwide.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

D3

Adobe has released an emergency update to its Flash Player after security researchers discovered a bug that allows attackers to take over and then crash users' machines.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

D4

This is not the first time Adobe software has been targeted. The worst attack came in 2013, when hackers managed to access personal data for nearly 3 million customers.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

D5

The Mustang is relatively expensive in Germany compared to the United States.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

D6

In a statement, Adobe called the flaw a "critical vulnerability" and urged users to update as soon as possible.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

D7

Adobe said it appeared that the flaw was being actively exploited on systems running Windows with Flash Player.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

D8

There are few cars available in Germany that offer the Mustang's performance capabilities at a similar price.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

D9

Active avoidance of food allergens in baby's diets did not protect them from developing food allergies

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

D10

Flash Player is widely used for watching videos, animations and other multimedia.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. (TOTAL: 10 points) Use K-Means to cluster the collection of web document in k=3 clusters. List the final clusters (3 points) and their centroids (4 points). How did you decide to stop the clustering process? (3 points) – you can write a program to do so – attach the source printout of the program at the end of the exam.

SOLUTION:

The document set is converted to a set of word counts as: -

D1 1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0

D2 0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0

D3 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0

D4 0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0

D5 0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0

D6 1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0

D7 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0

D8 0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0

D9 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0

D10 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1

K-means is done using Weka as the tool considering the document vectors as input and the vocabulary set as the attributes. The stopping criterion used in Weka is the check of monotonically decreasing RSS (Residual Sum of Squares). When the monotonicity of the RSS value in the clusters is not met it converges and gives the local minimum set of clusters

No. of clusters – 3

Random seed – 10

Max number of instances allowed – 500

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Attribute** | **Full Data** | **0** | **1** | **2** |
|  |  | **(D2 D5 D8 D9 D10)** | **(D3 D6 D7)** | **(D1 D4)** |
| Class | 5.5 | 6.8 | 5.3333 | 2.5 |
| mustang | 0.2 | 0.2 | 0 | 0.5 |
| built | 0.1 | 0 | 0 | 0.5 |
| flat | 0.1 | 0 | 0 | 0.5 |
| rock | 0.1 | 0 | 0 | 0.5 |
| mich | 0.1 | 0 | 0 | 0.5 |
| still | 0.1 | 0 | 0 | 0.5 |
| novelty | 0.1 | 0 | 0 | 0.5 |
| europe | 0.1 | 0 | 0 | 0.5 |
| went | 0.1 | 0 | 0 | 0.5 |
| sale | 0.1 | 0 | 0 | 0.5 |
| first | 0.2 | 0 | 0 | 1 |
| time | 0.2 | 0 | 0 | 1 |
| last | 0.1 | 0 | 0 | 0.5 |
| summer | 0.1 | 0 | 0 | 0.5 |
| food | 0.3 | 0.6 | 0 | 0 |
| allergies | 0.2 | 0.4 | 0 | 0 |
| rise | 0.1 | 0.2 | 0 | 0 |
| recent | 0.1 | 0.2 | 0 | 0 |
| years | 0.1 | 0.2 | 0 | 0 |
| currently | 0.1 | 0.2 | 0 | 0 |
| estimated | 0.1 | 0.2 | 0 | 0 |
| affect | 0.1 | 0.2 | 0 | 0 |
| eight | 0.1 | 0.2 | 0 | 0 |
| percent | 0.1 | 0.2 | 0 | 0 |
| children | 0.1 | 0.2 | 0 | 0 |
| worldwide | 0.1 | 0.2 | 0 | 0 |
| adobe | 0.4 | 0 | 1 | 0.5 |
| released | 0.1 | 0 | 0.3333 | 0 |
| emergency | 0.1 | 0 | 0.3333 | 0 |
| update | 0.2 | 0 | 0.6667 | 0 |
| flash | 0.3 | 0.2 | 0.6667 | 0 |
| player | 0.3 | 0.2 | 0.6667 | 0 |
| security | 0.1 | 0 | 0.3333 | 0 |
| researchers | 0.1 | 0 | 0.3333 | 0 |
| discovered | 0.1 | 0 | 0.3333 | 0 |
| bug | 0.1 | 0 | 0.3333 | 0 |
| allows | 0.1 | 0 | 0.3333 | 0 |
| attackers | 0.1 | 0 | 0.3333 | 0 |
| take | 0.1 | 0 | 0.3333 | 0 |
| crash | 0.1 | 0 | 0.3333 | 0 |
| users | 0.2 | 0 | 0.6667 | 0 |
| machines | 0.1 | 0 | 0.3333 | 0 |
| software | 0.1 | 0 | 0 | 0.5 |
| targeted | 0.1 | 0 | 0 | 0.5 |
| worst | 0.1 | 0 | 0 | 0.5 |
| attack | 0.1 | 0 | 0 | 0.5 |
| came | 0.1 | 0 | 0 | 0.5 |
| hackers | 0.1 | 0 | 0 | 0.5 |
| managed | 0.1 | 0 | 0 | 0.5 |
| access | 0.1 | 0 | 0 | 0.5 |
| personal | 0.1 | 0 | 0 | 0.5 |
| data | 0.1 | 0 | 0 | 0.5 |
| nearly | 0.1 | 0 | 0 | 0.5 |
| million | 0.1 | 0 | 0 | 0.5 |
| customers | 0.1 | 0 | 0 | 0.5 |
| relatively | 0.1 | 0.2 | 0 | 0 |
| expensive | 0.1 | 0.2 | 0 | 0 |
| germany | 0.2 | 0.4 | 0 | 0 |
| compared | 0.1 | 0.2 | 0 | 0 |
| united | 0.1 | 0.2 | 0 | 0 |
| states | 0.1 | 0.2 | 0 | 0 |
| statement | 0.1 | 0 | 0.3333 | 0 |
| called | 0.1 | 0 | 0.3333 | 0 |
| flaw | 0.2 | 0 | 0.6667 | 0 |
| critical | 0.1 | 0 | 0.3333 | 0 |
| vulnerability | 0.1 | 0 | 0.3333 | 0 |
| urged | 0.1 | 0 | 0.3333 | 0 |
| soon | 0.1 | 0 | 0.3333 | 0 |
| possible | 0.1 | 0 | 0.3333 | 0 |
| said | 0.1 | 0 | 0.3333 | 0 |
| appeared | 0.1 | 0 | 0.3333 | 0 |
| actively | 0.1 | 0 | 0.3333 | 0 |
| exploited | 0.1 | 0 | 0.3333 | 0 |
| systems | 0.1 | 0 | 0.3333 | 0 |
| running | 0.1 | 0 | 0.3333 | 0 |
| windows | 0.1 | 0 | 0.3333 | 0 |
| cars | 0.1 | 0.2 | 0 | 0 |
| available | 0.1 | 0.2 | 0 | 0 |
| offer | 0.1 | 0.2 | 0 | 0 |
| mustangs | 0.1 | 0.2 | 0 | 0 |
| performance | 0.1 | 0.2 | 0 | 0 |
| capabilities | 0.1 | 0.2 | 0 | 0 |
| similar | 0.1 | 0.2 | 0 | 0 |
| price | 0.1 | 0.2 | 0 | 0 |
| active | 0.1 | 0.2 | 0 | 0 |
| avoidance | 0.1 | 0.2 | 0 | 0 |
| allergens | 0.1 | 0.2 | 0 | 0 |
| babys | 0.1 | 0.2 | 0 | 0 |
| diets | 0.1 | 0.2 | 0 | 0 |
| protect | 0.1 | 0.2 | 0 | 0 |
| developing | 0.1 | 0.2 | 0 | 0 |
| widely | 0.1 | 0.2 | 0 | 0 |
| used | 0.1 | 0.2 | 0 | 0 |
| watching | 0.1 | 0.2 | 0 | 0 |
| videos | 0.1 | 0.2 | 0 | 0 |
| animations | 0.1 | 0.2 | 0 | 0 |
| multimedia | 0.1 | 0.2 | 0 | 0 |

Number of iterations: 2

Within cluster sum of squared errors: 68.09094650205762

The stopping criterion used in Weka is the check of monotonically decreasing RSS (Residual Sum of Squares). When the monotonicity of the RSS value in the clusters is not met it converges and gives the local minimum set of clusters.

1. (10 points) Evaluate the quality of your clusters with (a) Purity (3 points) (b) Normalized Mutual Information (3 points) and (c) Rand Index (4 points) if the documents have been classified into the following 4 classes:

CARS = {D1, D5, D8} SOFTWARE= {D3, D4, D6, D7, D10} ALERGIES = {D2, D9}

SOLUTION:

**PURITY**

Cluster 1: D2, D5, D8, D9, D10

Cluster 2: D3, D6, D7

Cluster 3: D1, D4

N = 10

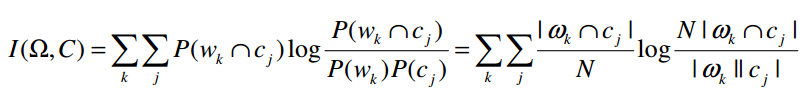
|  |  |  |  |
| --- | --- | --- | --- |
| **Classes** | **Cluster 1** | **Cluster 2** | **Cluster 3** |
| CARS | 2 | 0 | 1 |
| SOFTWARE | 1 | 3 | 1 |
| ALERGIES | 2 | 0 | 0 |

Purity = 1 / N ∑ max |Wk ∩ Cj|

∑ max |Wk ∩ Cj| = 2 + 3 + 1 = 6

Purity = (1/10) \* 5 = **0.6**

**NMI**



For Cluster

P(W1) = 5/10

P(W2) = 3/10

P(W3) = 2/10

For Class

P(C1) = 3/10

P(C2) = 5/10

P(C3) = 2/10

|  |  |
| --- | --- |
| W1 in C1 | 2 |
| W1 in C2 | 1 |
| W1 in C3 | 2 |
| W2 in C1 | 0 |
| W2 in C2 | 3 |
| W2 in C3 | 0 |
| W3 in C1 | 1 |
| W3 in C2 | 1 |
| W3 in C3 | 0 |

Hence, I = 0.507, H = 0.361

NMI = **1.404**

**Rand Index**

𝑅𝐼 (Ω, 𝐶) = (𝑇𝑃+𝑇𝑁) / (𝑇𝑃+𝐹𝑃+𝐹𝑁+𝑇𝑁)

Let CARS = C, SOFTWARE = S, ALERGIES = A

Cluster 1: A2, C5, A8, A9, S10

Cluster 2: S3, S6, S7

Cluster 3: C1, S4

TP = Pairs of documents belonging to same class and clustered together

= (3c2 + 3c2 + 0) = (3 + 3 + 0) = 6

TN = Pairs of documents belonging to different class and clustered differently

= (12 + 8 + 3) = 23

FP = Pairs of documents belonging to different class and clustered together

= (2 + 3 + 1 + 1) + (0) + (1) = 8

FN = Pairs of documents belonging to same class and clustered differently

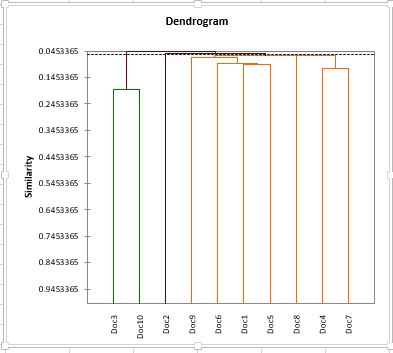
= (0 + 4 + 3 + 1) = 8

𝑅𝐼 (Ω, 𝐶) = (6 + 23) / (6 + 8 + 8 + 23) = 29 / 45 = **0.644**

1. (10 points) Using the same collection of documents, cluster them using (a) Single-link agglomerative clustering (3 points); (b) Complete-link agglomerative clustering (3 points); (c) Group-average link agglomerative clustering (4 points). Make sure to show: (i) the clusters and (ii) the centroids. You can write a program to resolve the problem – attach the source printout of the program at the end of the exam. IF you show only the clusters, you will be credited only 1 point. If the centroids are computed correctly, you will be credited 2 points per method.

SOLUTION

(a)



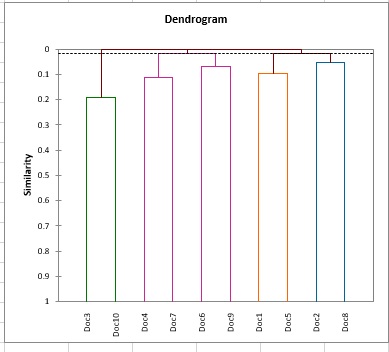
Class centroids:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Class | Doc1 | Doc2 | Doc3 | Doc4 | Doc5 | Doc6 | Doc7 | Doc8 | Doc9 | Doc10 |
| 1 | 0.189 | 0.031 | 0.033 | 0.181 | 0.176 | 0.18 | 0.183 | 0.176 | 0.17 | 0.007 |
| 2 | 0.039 | 1 | 0.014 | 0.057 | 0.034 | 0.022 | 0.015 | 0.055 | 0 | 0 |
| 3 | 0.018 | 0.007 | 0.595 | 0.015 | 0.038 | 0.018 | 0.017 | 0.012 | 0.023 | 0.595 |

Clusters and centroid vector:

|  |  |
| --- | --- |
| Centroid of cluster 1 (1,4,5,6,7,8,9) | {0.189, 0.031,0.033,0.181,0.176,0.18,0.183, 0.176,0.17,0.007} |
| Centroid of cluster 2 (2) | {0.039,1,0.014,0.057, 0.034,0.022,0.015,0.055,0,0} |
| Centroid of cluster 3 (3, 10) | {0.018, 0.007,0.595,0.015,0.038,0.018,0.017,0.012, 0.023 0.595} |

(b)



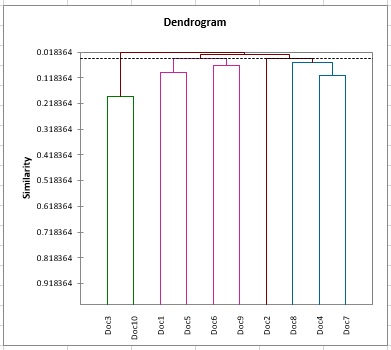
Class centroids:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Class | Doc1 | Doc2 | Doc3 | Doc4 | Doc5 | Doc6 | Doc7 | Doc8 | Doc9 | Doc10 |
| 1 | 0.548 | 0.036 | 0.038 | 0.026 | 0.548 | 0.068 | 0.029 | 0.039 | 0.017 | 0.018 |
| 2 | 0.051 | 0.527 | 0.019 | 0.06 | 0.025 | 0.016 | 0.034 | 0.527 | 0.013 | 0 |
| 3 | 0.018 | 0.007 | 0.595 | 0.015 | 0.038 | 0.018 | 0.017 | 0.012 | 0.023 | 0.595 |
| 4 | 0.041 | 0.023 | 0.033 | 0.289 | 0.029 | 0.278 | 0.292 | 0.039 | 0.282 | 0.003 |

Clusters and centroid vector:

|  |  |
| --- | --- |
| Centroid of cluster 1 (1,5) | {0.548, 0.036,0.038,0.026,0.548,0.068,0.029,0.039, 0.017,0.018} |
| Centroid of cluster 2 (2, 8) | {0.051, 0.527,0.019,0.06,0.025,0.016,0.034, 0.527,0.013,0} |
| Centroid of cluster 3 (3, 10) | {0.018,0.007,0.595,0.015,0.038,0.018,0.017,0.012, 0.023,0.595} |
| Centroid of cluster 4 (4, 6, 7, 9) | {0.041, 0.023,0.033,0.289,0.029,0.278,0.292,0.039, 0.282,0.003} |

(c)



Class centroids:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Class | Doc1 | Doc2 | Doc3 | Doc4 | Doc5 | Doc6 | Doc7 | Doc8 | Doc9 | Doc10 |
| 1 | 0.304 | 0.024 | 0.037 | 0.024 | 0.287 | 0.302 | 0.029 | 0.029 | 0.276 | 0.011 |
| 2 | 0.039 | 1 | 0.014 | 0.057 | 0.034 | 0.022 | 0.015 | 0.055 | 0 | 0 |
| 3 | 0.018 | 0.007 | 0.595 | 0.015 | 0.038 | 0.018 | 0.017 | 0.012 | 0.023 | 0.595 |
| 4 | 0.036 | 0.042 | 0.028 | 0.392 | 0.027 | 0.017 | 0.388 | 0.373 | 0.029 | 0.001 |

Clusters and centroid vector:

|  |  |
| --- | --- |
| Centroid of cluster 1 (1, 5, 6, 9) | {0.304,0.024,0.037,0.024,0.287,0.302,0.029,0.029, 0.276,0.011} |
| Centroid of cluster 2 (2) | {0.039, 1,0.014,0.057, 0.034,0.022,0.015,0.055,0,0} |
| Centroid of cluster 3 (3, 10) | {0.018, 0.007,0.595,0.015,0.038,0.018,0.017,0.012, 0.023,0.595} |
| Centroid of cluster 4 (4, 7, 8) | {0.036,0.042,0.028,0.392,0.027,0.017,0.388,0.373, 0.029,0.001} |

1. (10 points) Compute the page ranks of each of the Web pages from the graph crawled to find documents D1, D2, …, D10. You can write a program to resolve the problem – attach the source printout of the program at the end of the exam

Consider the following web graph:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

D1D2, D1 D3, D1 D5, D1 D7

D2 D4, D2 D7

D3 D1, D3 D6, D3 D10

D4 D5, D4 D6, D4 D9

D5 D6, D5 D7, D5 D9, D5 D10

D6 D2, D6 D3, D6 D8

D7 D1, D7 D3, D7 D9

D8 D1, D8 D5, D8 D10

D9 D2, D9 D5, D9 D10

D10 D1, D10 D3, D10 D6

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*SOLUTION*

**Page rank Computation:**

P(A) = (1 – D) + D[ P(T1) / C(T1) + … + P(Tn) / C(Tn) ]

A - Node

P – Page rank

D - damping factor = 0.85

T1, T2 …, Tn are the source nodes that point to Node A.

C(T1) is the number of outgoing links from T1.

PR(A) denotes the pagerank of node A.

|  |  |  |  |
| --- | --- | --- | --- |
| **Node** | **Source Nodes** | **Out Degree** | **Target Nodes** |
| D1 | D3, D7, D8, D10 | 4 | D2, D5, D3, D7 |
| D2 | D1, D6, D9 | 2 | D4, D7 |
| D3 | D1, D6, D7, D10 | 3 | D1, D6, D10 |
| D4 | D2 | 3 | D5, D6, D9 |
| D5 | D1, D4 ,D8 | 4 | D6, D7, D9, D10 |
| D6 | D3, D4, D5, D9, D10 | 3 | D2, D3, D8 |
| D7 | D1, D2, D5 | 3 | D1, D3, D9 |
| D8 | D6 | 3 | D1, D5, D10 |
| D9 | D4, D5, D7 | 3 | D2, D6, D10 |
| D10 | D3, D5, D8, D9 | 3 | D1, D3, D6 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Node** | **Source Nodes** | **Out Degree** | **Page Rank (PR)** |
| D1 | D3, D7, D8, D10 | 4 | PR(D1) = 0.15 + 0.85[PR(D3)/3 + PR(D7)/3 + PR(D8)/3 + PR(D10)/3] |
| D2 | D1, D6, D9 | 2 | PR(D2) = 0.15 + 0.85[PR(D1)/4 + PR(D6)/3 + PR(D9)/3] |
| D3 | D1, D6, D7, D10 | 3 | PR(D3) = 0.15 + 0.85[PR(D1)/4 + PR(D6)/3 + PR(D7)/3 + PR(D10)/3] |
| D4 | D2 | 3 | PR(D4) = 0.15 + 0.85[PR(D2)/2] |
| D5 | D1, D4 ,D8 | 4 | PR(D5) = 0.15 + 0.85[PR(D1)/4 + PR(D4)/3 + PR(D8)/3] |
| D6 | D3, D4, D5, D9, D10 | 3 | PR(D6) = 0.15 + 0.85[PR(D3)/3 + PR(D4)/3 + PR(D5)/4 + PR(D9)/3 + PR(D10)/3] |
| D7 | D1, D2, D5 | 3 | PR(D7) = 0.15 + 0.85[PR(D1)/4 + PR(D2)/2 + PR(D5)/4] |
| D8 | D6 | 3 | PR(D8) = 0.15 + 0.85[PR(D6)/3] |
| D9 | D4, D5, D7 | 3 | PR(D9) = 0.15 + 0.85[PR(D4)/3 + PR(D5)/4 + PR(D7)/3] |
| D10 | D3, D5, D8, D9 | 3 | PR(D10) = 0.15 + 0.85[PR(D3)/3 + PR(D5)/4+ PR(D8)/3 + PR(D9)/3] |

Using Fixed Point Computation, the Page Rank calculation is done in iterations. The formula is used is:



Total number of pages N= 10 (Given). So, Rank0(u) = 1 / 10

For the first iteration PR (it = 1),

|  |  |  |
| --- | --- | --- |
| **Node** | **Page Rank (PR)** | **PR(IT=1)** |
| D1 | PR(D1) = 0.15 + 0.85[PR(D3)/3 + PR(D7)/3 + PR(D8)/3 + PR(D10)/3] | 0.2633 |
| D2 | PR(D2) = 0.15 + 0.85[PR(D1)/4 + PR(D6)/3 + PR(D9)/3] | 0.2279 |
| D3 | PR(D3) = 0.15 + 0.85[PR(D1)/4 + PR(D6)/3 + PR(D7)/3 + PR(D10)/3] | 0.2562 |
| D4 | PR(D4) = 0.15 + 0.85[PR(D2)/2] | 0.1925 |
| D5 | PR(D5) = 0.15 + 0.85[PR(D1)/4 + PR(D4)/3 + PR(D8)/3] | 0.2279 |
| D6 | PR(D6) = 0.15 + 0.85[PR(D3)/3 + PR(D4)/3 + PR(D5)/4 + PR(D9)/3 + PR(D10)/3] | 0.2845 |
| D7 | PR(D7) = 0.15 + 0.85[PR(D1)/4 + PR(D2)/2 + PR(D5)/4] | 0.2350 |
| D8 | PR(D8) = 0.15 + 0.85[PR(D6)/3] | 0.1783 |
| D9 | PR(D9) = 0.15 + 0.85[PR(D4)/3 + PR(D5)/4 + PR(D7)/3] | 0.2279 |
| D10 | PR(D10) = 0.15 + 0.85[PR(D3)/3 + PR(D5)/4+ PR(D8)/3 + PR(D9)/3] | 0.2562 |

Taking the convergence factor as 0.002 i.e. previous +/- 0.002 > current, iterate till convergence is reached

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Node** | **PR(it = 0)** | **PR(it = 1)** | **PR(it = 2)** | **PR(it = 3)** | **PR(it = 4)** | **PR(it = 5)** |
| **D1** | 0.1 | 0.26333335 | 0.41231948 | 0.5449116 | 0.66123295 | 0.7590232 |
| **D2** | 0.1 | 0.22791667 | 0.3511667 | 0.4592741 | 0.54691255 | 0.621696 |
| **D3** | 0.1 | 0.25625002 | 0.42577782 | 0.5776617 | 0.7055831 | 0.8142681 |
| **D4** | 0.1 | 0.19250001 | 0.24686459 | 0.29924583 | 0.3451915 | 0.38243783 |
| **D5** | 0.1 | 0.22791667 | 0.3110278 | 0.3729086 | 0.4302293 | 0.47951466 |
| **D6** | 0.1 | 0.28458333 | 0.4627587 | 0.6066232 | 0.7265173 | 0.82932544 |
| **D7** | 0.1 | 0.23500001 | 0.35125524 | 0.45295715 | 0.5402283 | 0.61437356 |
| **D8** | 0.1 | 0.17833334 | 0.23063195 | 0.281115 | 0.3218766 | 0.35584658 |
| **D9** | 0.1 | 0.22791667 | 0.3195573 | 0.3855607 | 0.44236726 | 0.4922927 |
| **D10** | 0.1 | 0.25625002 | 0.38614064 | 0.49261746 | 0.58180535 | 0.6578747 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Node** | **PR(it = 6)** | **PR(it = 7)** | **PR(it = 8)** | **PR(it = 9)** | **PR(it = 10)** | **PR(it = 11)** |
| **D1** | 0.84200287 | 0.9127437 | 0.97285 | 1.02392 | 1.0673349 | 1.1042389 |
| **D2** | 0.68575096 | 0.74009085 | 0.786238 | 0.8254775 | 0.8588363 | 0.88718903 |
| **D3** | 0.9067383 | 0.9854562 | 1.052303 | 1.1091261 | 1.157431 | 1.1984895 |
| **D4** | 0.4142208 | 0.44144416 | 0.464539 | 0.48415095 | 0.5008279 | 0.51500547 |
| **D5** | 0.520473 | 0.5553646 | 0.585136 | 0.6104125 | 0.63188803 | 0.6501467 |
| **D6** | 0.91684437 | 0.99109006 | 1.054201 | 1.107863 | 1.1534736 | 1.1922406 |
| **D7** | 0.6774101 | 0.73097026 | 0.776512 | 0.8152231 | 0.84812355 | 0.8760903 |
| **D8** | 0.38497555 | 0.4097726 | 0.430809 | 0.44869035 | 0.46389452 | 0.47681755 |
| **D9** | 0.5343268 | 0.569896 | 0.600199 | 0.6259724 | 0.64786863 | 0.6664791 |
| **D10** | 0.7229123 | 0.77797866 | 0.8248 | 0.8646128 | 0.89845276 | 0.9272146 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Node** | **PR(it = 12)** | **PR(it = 13)** | **PR(it = 14)** | **PR(it = 15)** | **PR(it = 16)** | **PR(it = 17)** |
| **D1** | 1.1356068 | 1.1622692 | 1.1849324 | 1.2041961 | 1.2205703 | 1.2344881 |
| **D2** | 0.911288 | 0.9317728 | 0.9491849 | 0.96398497 | 0.9765651 | 0.9872582 |
| **D3** | 1.2333887 | 1.263053 | 1.2882679 | 1.3097004 | 1.327918 | 1.3434031 |
| **D4** | 0.5270554 | 0.5372974 | 0.54600346 | 0.5534036 | 0.55969363 | 0.56504023 |
| **D5** | 0.6656673 | 0.67885923 | 0.6900723 | 0.69960344 | 0.707705 | 0.71459126 |
| **D6** | 1.2251929 | 1.2532028 | 1.2770109 | 1.2972479 | 1.3144493 | 1.3290706 |
| **D7** | 0.8998623 | 0.92006814 | 0.93724334 | 0.95184207 | 0.96425104 | 0.9747987 |
| **D8** | 0.48780152 | 0.49713802 | 0.50507414 | 0.5118198 | 0.51755357 | 0.5224273 |
| **D9** | 0.6823 | 0.6957476 | 0.7071779 | 0.7168937 | 0.725152 | 0.73217165 |
| **D10** | 0.9516623 | 0.9724432 | 0.9901068 | 1.005121 | 1.017883 | 1.0287308 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Node** | **PR(it = 18)** | **PR(it = 19)** | **PR(it = 20)** | **PR(it = 21)** | **PR(it = 22)** | **PR(it = 23)** |
| **D1** | 1.2463186 | 1.2563744 | 1.2649218 | 1.2721872 | 1.2783625 | 1.2836118 |
| **D2** | 0.9963474 | 1.0040733 | 1.01064 | 1.016222 | 1.0209666 | 1.0249995 |
| **D3** | 1.3565655 | 1.3677533 | 1.3772631 | 1.3853463 | 1.3922172 | 1.3980573 |
| **D4** | 0.5695847 | 0.5734477 | 0.57673115 | 0.579522 | 0.5818944 | 0.5839108 |
| **D5** | 0.72044456 | 0.7254199 | 0.72964895 | 0.7332437 | 0.73629904 | 0.73889625 |
| **D6** | 1.3414986 | 1.3520623 | 1.3610417 | 1.368674 | 1.3751615 | 1.3806758 |
| **D7** | 0.98376405 | 0.99138486 | 0.99786246 | 1.0033684 | 1.0080485 | 1.0120264 |
| **D8** | 0.52657 | 0.5300913 | 0.53308433 | 0.53562844 | 0.537791 | 0.5396291 |
| **D9** | 0.7381383 | 0.74320996 | 0.74752104 | 0.7511853 | 0.7542999 | 0.7569474 |
| **D10** | 1.0379512 | 1.0457888 | 1.0524505 | 1.0581132 | 1.0629263 | 1.0670174 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Node** | **PR(it = 24)** | **PR(it = 25)** | **PR(it = 26)** | **PR(it = 27)** | **PR(it = 28)** | **PR(it = 29)** |
| **D1** | 1.2880735 | 1.2918661 | 1.2950897 | 1.2978299 | 1.3001589 | 1.3021387 |
| **D2** | 1.0284274 | 1.0313413 | 1.0338179 | 1.0359231 | 1.0377126 | 1.0392334 |
| **D3** | 1.4030213 | 1.407241 | 1.4108275 | 1.4138762 | 1.4164675 | 1.4186702 |
| **D4** | 0.5856248 | 0.5870817 | 0.5883201 | 0.58937263 | 0.59026736 | 0.59102786 |
| **D5** | 0.7411039 | 0.74298024 | 0.74457526 | 0.7459309 | 0.7470833 | 0.74806285 |
| **D6** | 1.3853632 | 1.3893473 | 1.3927338 | 1.3956124 | 1.398059 | 1.4001389 |
| **D7** | 1.0154078 | 1.0182818 | 1.0207249 | 1.0228014 | 1.0245665 | 1.0260668 |
| **D8** | 0.54119146 | 0.54251957 | 0.5436484 | 0.54460794 | 0.5454235 | 0.5461167 |
| **D9** | 0.7591977 | 0.76111054 | 0.7627363 | 0.7641183 | 0.765293 | 0.7662915 |
| **D10** | 1.070495 | 1.0734509 | 1.0759634 | 1.078099 | 1.0799145 | 1.0814574 |

|  |  |
| --- | --- |
| **Node** | **PR(it = 30)** |
| **D1** | 1.3038214 |
| **D2** | 1.0405264 |
| **D3** | 1.4205424 |
| **D4** | 0.5916742 |
| **D5** | 0.7488954 |
| **D6** | 1.4019067 |
| **D7** | 1.0273422 |
| **D8** | 0.546706 |
| **D9** | 0.76714015 |
| **D10** | 1.0827689 |

1. (10 points) Use the HITS algorithm to compute the hub and authority score of each Web page.

SOLUTION

1. Let,

Hub score of a page X: h(X)

Authority score of page X: a(X)

1. Initially,

For all X, h(X)= 1 and a(X)=1

1. In each iteration:

h(X)= sum of all authority scores of all pages which are target to X.

a(X)= sum of all hub scores of all pages which are source to X

1. Normalization:

Hub score = hub score / √ sum of square all hub scores

Authority score = authority score / √ sum of square of all authority scores

Iteration 1:

Calculating the Authority and Hub Score for first Iteration:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Node** | **h(X)** | **Normalised h(X)** | **a(X)** | **Normalised A(X)** |
| D1 | 4 | 0.707 | 4 | 0.707 |
| D2 | 2 | 0.555 | 3 | 0.832 |
| D3 | 3 | 0.6 | 4 | 0.8 |
| D4 | 3 | 0.949 | 1 | 0.316 |
| D5 | 4 | 0.8 | 3 | 0.6 |
| D6 | 3 | 0.514 | 5 | 0.857 |
| D7 | 3 | 0.707 | 3 | 0.707 |
| D8 | 3 | 0.949 | 1 | 0.316 |
| D9 | 3 | 0.707 | 3 | 0.707 |
| D10 | 3 | 0.6 | 4 | 0.8 |

Taking the convergence factor as 0.002 i.e. previous +/- 0.002 > current, iterate till convergence is reached. Calculating the authority score for next iterations: -

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Node** | **A(it = 0)** | **A(it = 1)** | **A(it = 2)** | **A(it = 3)** | **A(it = 4)** | **A(it = 5)** | **A(it = 6)** | **A(it = 7)** |
| **D1** | 1 | 0.707 | 0.697 | 0.714 | 0.711 | 0.717 | 0.714 | 0.718 |
| **D2** | 1 | 0.832 | 0.883 | 0.898 | 0.904 | 0.909 | 0.909 | 0.911 |
| **D3** | 1 | 0.8 | 0.73 | 0.73 | 0.716 | 0.719 | 0.714 | 0.717 |
| **D4** | 1 | 0.316 | 0.248 | 0.206 | 0.192 | 0.187 | 0.182 | 0.183 |
| **D5** | 1 | 0.6 | 0.647 | 0.656 | 0.654 | 0.658 | 0.656 | 0.658 |
| **D6** | 1 | 0.857 | 0.883 | 0.896 | 0.897 | 0.9 | 0.899 | 0.9 |
| **D7** | 1 | 0.707 | 0.681 | 0.675 | 0.663 | 0.666 | 0.659 | 0.663 |
| **D8** | 1 | 0.316 | 0.237 | 0.215 | 0.2 | 0.2 | 0.196 | 0.198 |
| **D9** | 1 | 0.707 | 0.702 | 0.694 | 0.689 | 0.691 | 0.688 | 0.69 |
| **D10** | 1 | 0.8 | 0.791 | 0.805 | 0.801 | 0.806 | 0.803 | 0.805 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Node** | **A(it = 8)** | **A(it = 9)** | **A(it = 10)** | **A(it = 11)** | **A(it = 12)** | **A(it = 13)** |
| **D1** | 0.715 | 0.718 | 0.716 | 0.717 | 0.716 | 0.717 |
| **D2** | 0.91 | 0.912 | 0.91 | 0.912 | 0.911 | 0.912 |
| **D3** | 0.714 | 0.717 | 0.715 | 0.716 | 0.715 | 0.716 |
| **D4** | 0.18 | 0.182 | 0.18 | 0.181 | 0.18 | 0.181 |
| **D5** | 0.656 | 0.658 | 0.657 | 0.658 | 0.657 | 0.657 |
| **D6** | 0.899 | 0.9 | 0.899 | 0.9 | 0.899 | 0.9 |
| **D7** | 0.659 | 0.661 | 0.659 | 0.661 | 0.659 | 0.661 |
| **D8** | 0.196 | 0.197 | 0.196 | 0.197 | 0.196 | 0.197 |
| **D9** | 0.689 | 0.69 | 0.689 | 0.69 | 0.689 | 0.69 |
| **D10** | 0.804 | 0.805 | 0.804 | 0.805 | 0.804 | 0.805 |

After 13 iterations, convergence was reached.

The hub score for the same iterations is as follows: -

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Node** | **H(it = 0)** | **H(it = 1)** | **H(it = 2)** | **H(it = 3)** | **H(it = 4)** | **H(it = 5)** | **H(it = 6)** | **H(it = 7)** |
| **D1** | 1 | 0.707 | 0.717 | 0.7 | 0.703 | 0.697 | 0.7 | 0.696 |
| **D2** | 1 | 0.555 | 0.469 | 0.44 | 0.427 | 0.416 | 0.418 | 0.411 |
| **D3** | 1 | 0.6 | 0.683 | 0.683 | 0.698 | 0.695 | 0.7 | 0.697 |
| **D4** | 1 | 0.949 | 0.969 | 0.979 | 0.981 | 0.982 | 0.983 | 0.983 |
| **D5** | 1 | 0.8 | 0.763 | 0.755 | 0.756 | 0.753 | 0.755 | 0.753 |
| **D6** | 1 | 0.514 | 0.47 | 0.444 | 0.443 | 0.436 | 0.439 | 0.435 |
| **D7** | 1 | 0.707 | 0.732 | 0.738 | 0.748 | 0.746 | 0.752 | 0.749 |
| **D8** | 1 | 0.949 | 0.971 | 0.977 | 0.98 | 0.98 | 0.981 | 0.98 |
| **D9** | 1 | 0.707 | 0.712 | 0.72 | 0.725 | 0.723 | 0.725 | 0.724 |
| **D10** | 1 | 0.6 | 0.612 | 0.594 | 0.598 | 0.592 | 0.596 | 0.593 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Node** | **H(it = 8)** | **H(it = 9)** | **H(it = 10)** | **H(it = 11)** | **H(it = 12)** | **H(it = 13)** |
| **D1** | 0.699 | 0.697 | 0.698 | 0.697 | 0.698 | 0.697 |
| **D2** | 0.415 | 0.411 | 0.414 | 0.411 | 0.413 | 0.411 |
| **D3** | 0.7 | 0.698 | 0.699 | 0.698 | 0.699 | 0.698 |
| **D4** | 0.984 | 0.983 | 0.984 | 0.983 | 0.984 | 0.983 |
| **D5** | 0.755 | 0.753 | 0.754 | 0.753 | 0.754 | 0.754 |
| **D6** | 0.438 | 0.436 | 0.438 | 0.436 | 0.437 | 0.436 |
| **D7** | 0.752 | 0.75 | 0.752 | 0.751 | 0.752 | 0.751 |
| **D8** | 0.981 | 0.98 | 0.981 | 0.98 | 0.981 | 0.98 |
| **D9** | 0.725 | 0.724 | 0.725 | 0.724 | 0.725 | 0.724 |
| **D10** | 0.595 | 0.593 | 0.595 | 0.594 | 0.595 | 0.594 |

After 13 iterations, the hub score also stabilizes. So the final scores are: -

|  |  |  |
| --- | --- | --- |
| **Node** | **A(it = 13)** | **H(it = 13)** |
| **D1** | 0.717 | 0.697 |
| **D2** | 0.912 | 0.411 |
| **D3** | 0.716 | 0.698 |
| **D4** | 0.181 | 0.983 |
| **D5** | 0.657 | 0.754 |
| **D6** | 0.9 | 0.436 |
| **D7** | 0.661 | 0.751 |
| **D8** | 0.197 | 0.98 |
| **D9** | 0.69 | 0.724 |
| **D10** | 0.805 | 0.594 |

1. (10 points) Consider that each cluster obtained at step B represents a different topic. Compute the topic-sensitive ranks of each Web page. You can write a program to resolve the problem – attach the source printout of the program at the end of the exam.

*SOLUTION*

**Step 1:** Build Stochastic Matrix. If there is a hyperlink from i to j then enter Aij =1; else it is 0

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **D1** | **D2** | **D3** | **D4** | **D5** | **D6** | **D7** | **D8** | **D9** | **D10** |
| **D1** | 0 | 1/4 | 1/4 | 0 | 1/4 | 0 | 1/4 | 0 | 0 | 0 |
| **D2** | 0 | 0 | 0 | 1/2 | 0 | 0 | 1/2 | 0 | 0 | 0 |
| **D3** | 1/3 | 0 | 0 | 0 | 0 | 1/3 | 0 | 0 | 0 | 1/3 |
| **D4** | 0 | 0 | 0 | 0 | 1/3 | 1/3 | 0 | 0 | 1/3 | 0 |
| **D5** | 0 | 0 | 0 | 0 | 0 | 1/4 | 1/4 | 0 | 1/4 | 1/4 |
| **D6** | 0 | 1/3 | 1/3 | 0 | 0 | 0 | 0 | 1/3 | 0 | 0 |
| **D7** | 1/3 | 0 | 1/3 | 0 | 0 | 0 | 0 | 0 | 1/3 | 0 |
| **D8** | 1/3 | 0 | 0 | 0 | 1/3 | 0 | 0 | 0 | 0 | 1/3 |
| **D9** | 0 | 1/3 | 0 | 0 | 1/3 | 0 | 0 | 0 | 0 | 1/3 |
| **D10** | 1/3 | 0 | 1/3 | 0 | 0 | 1/3 | 0 | 0 | 0 | 0 |

**Step 2:**

Here, α =0.85

Multiply matrix M by (1-α) [(1-α) = 0.15 = 15/100 = **3/20**]

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **D1** | **D2** | **D3** | **D4** | **D5** | **D6** | **D7** | **D8** | **D9** | **D10** |
| **D1** | 0 | 3/80 | 3/80 | 0 | 3/80 | 0 | 3/80 | 0 | 0 | 0 |
| **D2** | 0 | 0 | 0 | 3/40 | 0 | 0 | 3/40 | 0 | 0 | 0 |
| **D3** | 1/20 | 0 | 0 | 0 | 0 | 1/20 | 0 | 0 | 0 | 1/20 |
| **D4** | 0 | 0 | 0 | 0 | 1/20 | 1/20 | 0 | 0 | 1/20 | 0 |
| **D5** | 0 | 0 | 0 | 0 | 0 | 3/80 | 3/80 | 0 | 3/80 | 3/80 |
| **D6** | 0 | 1/20 | 1/20 | 0 | 0 | 0 | 0 | 1/20 | 0 | 0 |
| **D7** | 1/20 | 0 | 1/20 | 0 | 0 | 0 | 0 | 0 | 1/20 | 0 |
| **D8** | 1/20 | 0 | 0 | 0 | 1/20 | 0 | 0 | 0 | 0 | 1/20 |
| **D9** | 0 | 1/20 | 0 | 0 | 1/20 | 0 | 0 | 0 | 0 | 1/20 |
| **D10** | 1/20 | 0 | 1/20 | 0 | 0 | 1/20 | 0 | 0 | 0 | 0 |

**Step 3:** Multiply Rank (uniform rank)with sum of row entries of above computed matrix

Given, uniform ranks (from Part D calculations)

|  |  |  |  |
| --- | --- | --- | --- |
| **Node** | **Uniform Page Rank** | **M \* (1-α) \* Rank (steps)** | **Rank** |
| **D1** | 1.3038214 | (3/80 + 3/80 + 3/80 + 3/80) \* 1.3038214 | 0.19557321 |
| **D2** | 1.0405264 | (3/40 + 3/40) \* 1.0405264 | 0.15607896 |
| **D3** | 1.4205424 | (1/20 + 1/20 + 1/20) \* 1.4205424 | 0.21308136 |
| **D4** | 0.5916742 | (1/20 + 1/20 + 1/20) \* 0.5916742 | 0.08875113 |
| **D5** | 0.7488954 | (3/80 + 3/80 + 3/80 + 3/80) \* 0.7488954 | 0.11233431 |
| **D6** | 1.4019067 | (1/20 + 1/20 + 1/20)\* 1.4019067 | 0.210286005 |
| **D7** | 1.0273422 | (1/20 + 1/20 + 1/20)\* 1.0273422 | 0.15410133 |
| **D8** | 0.546706 | (1/20 + 1/20 + 1/20) \* 0.546706 | 0.0820059 |
| **D9** | 0.76714015 | (1/20 + 1/20 + 1/20) \* 0.76714015 | 0.115071023 |
| **D10** | 1.0827689 | (1/20 + 1/20 + 1/20) \* 1.0827689 | 0.162415335 |

**Step 4:** Computing Topic specific ranks

(Personalization vector pi = 1/N (1/10 = 0.1) for document belongs to cluster; 0 otherwise)

1. Topic Specific Rank for Cluster 1 (Doc set = D2 D5 D8 D9 D10)

|  |  |  |  |
| --- | --- | --- | --- |
| **Node** | **Uniform Page Rank** | **Rank** | **Topic Specific Rank** |
| **D1** | 1.3038214 | 0.19557321 | 0.19557321 |
| **D2** | 1.0405264 | 0.15607896 | 0.25607896 |
| **D3** | 1.4205424 | 0.21308136 | 0.21308136 |
| **D4** | 0.5916742 | 0.08875113 | 0.08875113 |
| **D5** | 0.7488954 | 0.11233431 | 0.21233431 |
| **D6** | 1.4019067 | 0.210286005 | 0.210286005 |
| **D7** | 1.0273422 | 0.15410133 | 0.15410133 |
| **D8** | 0.546706 | 0.0820059 | 0.1820059 |
| **D9** | 0.76714015 | 0.115071023 | 0.215071023 |
| **D10** | 1.0827689 | 0.162415335 | 0.262415335 |

1. Topic Specific Rank for Cluster 2 (Doc set = D3 D6 D7)

|  |  |  |  |
| --- | --- | --- | --- |
| **Node** | **Uniform Page Rank** | **Rank** | **Topic Specific Rank** |
| **D1** | 1.3038214 | 0.19557321 | 0.19557321 |
| **D2** | 1.0405264 | 0.15607896 | 0.15607896 |
| **D3** | 1.4205424 | 0.21308136 | 0.31308136 |
| **D4** | 0.5916742 | 0.08875113 | 0.08875113 |
| **D5** | 0.7488954 | 0.11233431 | 0.11233431 |
| **D6** | 1.4019067 | 0.210286005 | 0.310286005 |
| **D7** | 1.0273422 | 0.15410133 | 0.25410133 |
| **D8** | 0.546706 | 0.0820059 | 0.0820059 |
| **D9** | 0.76714015 | 0.115071023 | 0.115071023 |
| **D10** | 1.0827689 | 0.162415335 | 0.162415335 |

1. Topic Specific Rank for Cluster 1 (Doc set = D1 D4)

|  |  |  |  |
| --- | --- | --- | --- |
| **Node** | **Uniform Page Rank** | **Rank** | **Topic Specific Rank** |
| **D1** | 1.3038214 | 0.19557321 | 0.29557321 |
| **D2** | 1.0405264 | 0.15607896 | 0.15607896 |
| **D3** | 1.4205424 | 0.21308136 | 0.21308136 |
| **D4** | 0.5916742 | 0.08875113 | 0.18875113 |
| **D5** | 0.7488954 | 0.11233431 | 0.11233431 |
| **D6** | 1.4019067 | 0.210286005 | 0.210286005 |
| **D7** | 1.0273422 | 0.15410133 | 0.15410133 |
| **D8** | 0.546706 | 0.0820059 | 0.0820059 |
| **D9** | 0.76714015 | 0.115071023 | 0.115071023 |
| **D10** | 1.0827689 | 0.162415335 | 0.162415335 |

**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:*

[*www.cnn.com*](http://www.cnn.com)

[*money.cnn.com*](http://money.cnn.com)

[*www.cnn.edu/world/*](http://www.cnn.edu/world/)

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

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

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

[*www.cs.stanford.edu/~ng*](http://www.cs.stanford.edu/~ng)

[*http://machinelearning.cis.cornell.edu/pages/people.php*](http://machinelearning.cis.cornell.edu/pages/people.php)[*http://www.cs.cornell.edu/home/kleinber/*](http://www.cs.cornell.edu/home/kleinber/)

[*http://shop.nordstrom.com/*](http://shop.nordstrom.com/)

[*http://www.newbalance.com/*](http://www.newbalance.com/)

[*http://www.newbalance.com/women/clothing/short-sleeve-shirts-1/*](http://www.newbalance.com/women/clothing/short-sleeve-shirts-1/)

[*www.macys.com*](http://www.macys.com)

[*http://www.amazon.com*](http://www.amazon.com)

*Show how you arrange the URLs in the front queues (3 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:*

Given Queues are**:** 3 front queues and 6 back queues.

Let us consider the given front queues as FQ1, FQ2, and FQ3. Usually the URL’s which require more frequent refreshing and frequent crawling must be assigned higher Priority than the URLs which require less crawling. Considering the three front queues as Priority queues with FQ1 having the highest priority and FQ3 having the least. So the URLs inside FQ1 are: (URLs which require high refreshing)

**FQ1:**

[www.cnn.com](http://www.cnn.com)

[money.cnn.com](http://www.money.cnn.com)

Similarly arranging URL’s based on the crawling rate in front queues, we have the following structure:

**FQ2:**

[www.macys.com](http://www.macys.com)

<http://www.newbalance.com/>

<http://shop.nordstrom.com/>

<http://www.amazon.com>

<http://www.newbalance.com/women/clothing/short-sleeve-shirts-1/>

**FQ3(URL’s which require least crawling):**

[www.cnn.edu/world/](http://www.cnn.edu/world/)

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

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

<http://machinelearning.cis.cornell.edu/pages/people.php> <http://www.cs.cornell.edu/home/kleinber/>

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

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

**Passing URLs from Front Queues to Back Queues:**

While passing the URLs from the front queue to the back queue we should ensure that URLs from the queue with the highest priority should be pulled out at a higher rate than from the rest of the queues. So the iterations would be like: -

{FQ1}, {FQ1, FQ2}, {FQ1, FQ2, FQ3}.

We have 6 back queues and we assign a separate queue for each host. We will end with the following structure: -

**BQ1 (For CNN):**

[www.cnn.com](http://www.cnn.com)

[money.cnn.com](http://www.money.cnn.com)

[www.cnn.edu/world/](http://www.cnn.edu/world/)

**BQ2 (For Macys):**

[www.macys.com](http://www.macys.com)

**BQ3 (For NewBalance):**

<http://www.newbalance.com/>

<http://www.newbalance.com/women/clothing/short-sleeve-shirts-1/>

**BQ4 (For Nordstrom):**

<http://shop.nordstrom.com/>

**BQ5 (For Amazon):**

<http://www.amazon.com>

**BQ6 (For Cornell domain):**

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

<http://machinelearning.cis.cornell.edu/pages/people.php> <http://www.cs.cornell.edu/home/kleinber/>

Since the number of back queues are now exhausted, the following URLs will be assigned after a back queues becomes empty:

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

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

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

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 link analysis.*

*W2: Link analysis techniques shall be presented in the class.*

*Use the Jaccard coefficient to compute the similarity between the sketch vectors (3 points); then generate the signatures for the two web pages and compute the similarity of the signatures (5 points). Finally, use row hashing with the following two hash functions:*

*h1=(x +1)mod 5; h2=(x+2) mod 5; (7 points) Explain why the web pages W1 and W2 are dissimilar in all three methods.*

*SOLUTION:*

Considering Shingling of size 2 for the documents.

**Shingles for W1:**

The class, class will, will cover, cover link, link analysis

**Shingles for W2:**

Link analysis, analysis techniques, techniques shall, shall be, be presented, presented in, in the, the class

So the unique set of shingles is:

|  |  |
| --- | --- |
| **S1** | The class |
| **S2** | class will |
| **S3** | will cover |
| **S4** | cover link |
| **S5** | link analysis |
| **S6** | analysis techniques |
| **S7** | techniques shall |
| **S8** | shall be |
| **S9** | be presented |
| **S10** | presented in |
| **S11** | in the |

Calculating Jaccard similarity:

|  |  |  |
| --- | --- | --- |
|  | **W1** | **W2** |
| **S1** | 1 | 1 |
| **S2** | 1 | 0 |
| **S3** | 1 | 0 |
| **S4** | 1 | 0 |
| **S5** | 1 | 1 |
| **S6** | 0 | 1 |
| **S7** | 0 | 1 |
| **S8** | 0 | 1 |
| **S9** | 0 | 1 |
| **S10** | 0 | 1 |
| **S11** | 0 | 1 |

Jaccard’s similarity = (A n B) / (A u B)

= Number of shingles that occur in both documents / Number of shingles in both documents combined.

= 2 / 11

Requirement for nearly duplicate document is >= 80%

Hence, documents are not similar.

The following results are generated by 25 random permutations with an online tool, <http://www.miniwebtool.com/random-picker/> :

Sketch = list of 25 indexes of first rows with 1 in column Wi by each permutation

Sketch vectors of size 25 for W1 and W2

Signature of the two docs:

|  |  |  |
| --- | --- | --- |
| **Permutation** | **W1** | **W2** |
| **P1** | 3 | 6 |
| **P2** | 3 | 10 |
| **P3** | 2 | 8 |
| **P4** | 5 | 8 |
| **P5** | 4 | 8 |
| **P6** | 1 | 10 |
| **P7** | 3 | 10 |
| **P8** | 2 | 7 |
| **P9** | 3 | 8 |
| **P10** | 3 | 7 |
| **P11** | 5 | 6 |
| **P12** | 2 | 9 |
| **P13** | 2 | 10 |
| **P14** | 3 | 7 |
| **P15** | 1 | 1 |
| **P16** | 3 | 7 |
| **P17** | 4 | 7 |
| **P18** | 1 | 1 |
| **P19** | 3 | 9 |
| **P20** | 4 | 5 |
| **P21** | 4 | 8 |
| **P22** | 2 | 5 |
| **P23** | 4 | 1 |
| **P24** | 3 | 1 |
| **P25** | 4 | 7 |

sim[sketch(Ci), sketch(Cj)] = fraction of permutations where minimum values agree

**Jaccard’s Similarity** = 2/25 = 0.08.

Requirement for nearly duplicate document is >= 80%

Hence, documents are not similar.

Using Row Hashing:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | **W1 slots** | **W2 slots** |
| **J = 1** | h1(1) = 3  h2(1) = 2 | 3  2 | 3  2 |
| **J = 2** | h1(2) = 4  h2(2) = 3 | 3  2 | 3  2 |
| **J = 3** | h1(3) = 0  h2(3) = 4 | 0  2 | 3  2 |
| **J = 4** | h1(4) = 1  h2(4) = 0 | 0  0 | 3  2 |
| **J = 5** | h1(5) = 2  h2(5) = 1 | 0  0 | 2  1 |
| **J = 6** | h1(6) = 3  h2(6) = 2 | 0  0 | 2  1 |
| **J = 7** | h1(7) = 4  h2(7) = 3 | 0  0 | 2  1 |
| **J = 8** | h1(8) = 0  h2(8) = 4 | 0  0 | 0  1 |
| **J = 9** | h1(9) = 1  h2(9) = 0 | 0  0 | 0  0 |
| **J = 10** | h1(10) = 2  h2(10) = 1 | 0  0 | 0  0 |
| **J = 11** | h1(11) = 3  h2(11) = 2 | 0  0 | 0  0 |

Jaccard coefficient: J(W1, W2) = (1 + 1 + 1) / 10 = 0.3

Requirement for nearly duplicate document is >= 80%

Hence, documents are not similar.

**Problem 3 (15 points) :**

Consider the same document collection as in Problem 1.

1. *(10 points) For the query Q0= "Adobe security" you are told that the following documents are relevant: D3, D4. Use pseudo-relevance feedback to expand the query with 2 additional keywords by using the local analysis method based on association clusters. Show the expanded query (2 points), the association clusters without normalization (2 points) and with normalization (2 points) and explain how you selected the 2 new keywords to be added to the initial query (4 points).*

*SOLUTION*



D3: Adobe has released an emergency update to its Flash Player after security researchers discovered a bug that allows attackers to take over and then crash users' machines.

D4: This is not the first time Adobe software has been targeted. The worst attack came in 2013, when hackers managed to access personal data for nearly 3 million customers.

Terms in local vocabulary in sorted order:

access,adobe,allows,attack,bug,came,crash,customers,data,discovered,emergency,first,flash,hackers,machines,managed,million,nearly,personal,player,released,researchers,security,software,take,targeted,time,update,users,worst

V = 31. Since there are attackers and attack as same stem, stems S = 30 and N = 2.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | **D3** | **D4** |
| **S1** | **discovered** | 1 | 0 |
| **S2** | **access** | 0 | 1 |
| **S3** | **software** | 0 | 1 |
| **S4** | **data** | 0 | 1 |
| **S5** | **nearly** | 0 | 1 |
| **S6** | **emergency** | 1 | 0 |
| **S7** | **update** | 1 | 0 |
| **S8** | **crash** | 1 | 0 |
| **S9** | **security** | 1 | 0 |
| **S10** | **attack** | 1 | 1 |
| **S11** | **hackers** | 0 | 1 |
| **S12** | **managed** | 0 | 1 |
| **S13** | **targeted** | 0 | 1 |
| **S14** | **came** | 0 | 1 |
| **S15** | **customers** | 0 | 1 |
| **S16** | **released** | 1 | 0 |
| **S17** | **player** | 1 | 0 |
| **S18** | **allows** | 1 | 0 |
| **S19** | **researchers** | 1 | 0 |
| **S20** | **adobe** | 1 | 1 |
| **S21** | **personal** | 0 | 1 |
| **S22** | **take** | 1 | 0 |
| **S23** | **users'** | 1 | 0 |
| **S24** | **million** | 0 | 1 |
| **S25** | **bug** | 1 | 0 |
| **S26** | **worst** | 0 | 1 |
| **S27** | **machines** | 1 | 0 |
| **S28** | **time** | 0 | 1 |
| **S20** | **first** | 0 | 1 |
| **S30** | **flash** | 1 | 0 |

We assume N = 2 for each association cluster. The query is Adobe security

S20 = adobe S9 = security

We shall build association clusters S20(2) and S9(2) therefore we shall pick the largest two S20 out of S20,1...S20,9 and largest two S9 out of S9,1..S9,9.

Computing association for “Adobe”:

|  |  |  |
| --- | --- | --- |
| **Pair(u v)** | **Cuv** | **Suv** |
| **M(adobe discovered)** | 1 | 0.25 |
| **M(adobe access)** | 1 | 0.25 |
| **M(adobe software)** | 1 | 0.25 |
| **M(adobe data)** | 1 | 0.25 |
| **M(adobe nearly)** | 1 | 0.25 |
| **M(adobe emergency)** | 1 | 0.25 |
| **M(adobe update)** | 1 | 0.25 |
| **M(adobe crash)** | 1 | 0.25 |
| **M(adobe security)** | 1 | 0.25 |
| **M(adobe attack)** | 1 | 0.4 |
| **M(adobe hackers)** | 1 | 0.25 |
| **M(adobe managed)** | 1 | 0.25 |
| **M(adobe targeted)** | 1 | 0.25 |
| **M(adobe came)** | 1 | 0.25 |
| **M(adobe customers)** | 1 | 0.25 |
| **M(adobe released)** | 1 | 0.25 |
| **M(adobe player)** | 1 | 0.25 |
| **M(adobe allows)** | 1 | 0.25 |
| **M(adobe researchers)** | 1 | 0.25 |
| **M(adobe personal)** | 1 | 0.25 |
| **M(adobe users)** | 1 | 0.25 |
| **M(adobe take)** | 1 | 0.25 |
| **M(adobe million)** | 1 | 0.25 |
| **M(adobe bug)** | 1 | 0.25 |
| **M(adobe worst)** | 1 | 0.25 |
| **M(adobe machines)** | 1 | 0.25 |
| **M(adobe time)** | 1 | 0.25 |
| **M(adobe first)** | 1 | 0.25 |
| **M(adobe flash)** | 1 | 0.25 |

Computing association for “security”:

|  |  |  |
| --- | --- | --- |
| **Pair(u v)** | **Cuv** | **Suv** |
| **M(security discovered)** | 1 | 0.333333 |
| **M(security access)** | 0 | 0 |
| **M(security software)** | 0 | 0 |
| **M(security data)** | 0 | 0 |
| **M(security nearly)** | 0 | 0 |
| **M(security emergency)** | 1 | 0.333333 |
| **M(security update)** | 1 | 0.333333 |
| **M(security crash)** | 1 | 0.333333 |
| **M(security attack)** | 0 | 0.333333 |
| **M(security hackers)** | 0 | 0 |
| **M(security managed)** | 0 | 0 |
| **M(security targeted)** | 0 | 0 |
| **M(security came)** | 0 | 0 |
| **M(security customers)** | 0 | 0 |
| **M(security released)** | 1 | 0.333333 |
| **M(security player)** | 1 | 0.333333 |
| **M(security allows)** | 1 | 0.333333 |
| **M(security researchers)** | 1 | 0.333333 |
| **M(security adobe)** | 1 | 0.25 |
| **M(security personal)** | 0 | 0 |
| **M(security users)** | 1 | 0.333333 |
| **M(security take)** | 1 | 0.333333 |
| **M(security million)** | 0 | 0 |
| **M(security bug)** | 1 | 0.333333 |
| **M(security worst)** | 0 | 0 |
| **M(security machines)** | 1 | 0.333333 |
| **M(security time)** | 0 | 0 |
| **M(security first)** | 0 | 0 |
| **M(security flash)** | 1 | 0.333333 |

Terms for un-normalized cluster:

The terms have same value of 1 and hence by random selection I choose attack and access for “adobe” cluster and access and allow for “shopping” cluster.

Un-normalized association cluster Adobe = {attack, access}

Un-normalized association cluster security = {access, allow}

**Query expansion**

Adobe security attack access allow

Terms for family for normalized cluster:

I choose attack as it has value of 0.4 for “adobe”.

The terms have same value of 0.25 for “adobe” and 0.333 for “security” , hence by sorted selection I choose and access for “adobe” cluster and allow and bug for “shopping” cluster as attack and access are now already in the query.

Normalized association cluster Adobe = {attack, access}

Normalized association cluster security = {access, allow}

**Query expansion**

Adobe security attack access allow

1. *(5 points) Use a global analysis method based on a global similarity thesaurus, expand the same query as in question A with 2 additional keywords. Show the term vectors for each of the keywords of the query Q0 and show how you have computed them (2 points), and select the 2 new keywords to be added to the initial query (3 points). You can write a program to resolve the problem – attach the source printout of the program at the end of the exam*

*SOLUTION*

Let:

t → number of terms in collection

N be the number of documents in collection

fi,j → frequency of occurrence of term Ki in document dj

tj → number of distinct index terms in document dj

itfj → inverse term freq for document dj

itfj = log (t / tj)



here, N = 10, t = 95

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| d(j) | f(adobe, j) | f(security, j) | t(j) | itf(j) | W(adobe, j) | W(security, j) |
| 1 | 0 | 0 | 14 | 0.831 | 0 | 0 |
| 2 | 0 | 0 | 12 | 0.898 | 0 | 0 |
| 3 | 1 | 1 | 16 | 0.774 | 0.4056 | 1 |
| 4 | 1 | 0 | 16 | 0.774 | 0.4056 | 0 |
| 5 | 0 | 0 | 7 | 1.133 | 0 | 0 |
| 6 | 1 | 0 | 11 | 0.936 | 0.4905 | 0 |
| 7 | 1 | 0 | 11 | 0.936 | 0.4905 | 0 |
| 8 | 0 | 0 | 9 | 1.023 | 0 | 0 |
| 9 | 0 | 0 | 9 | 1.023 | 0 | 0 |
| 10 | 0 | 0 | 8 | 1.075 | 0 | 0 |

We get the Term vectors as the following:

K adobe:

(0,0,0.4056,0.4056,0,0.4905,0.4905,0,0,0)

K security:

(0,0,1,0,0,0,0,0,0,0)

Expanded Terms: **adobe security attack access**

**APPENDIX**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Code used to resolve the problems in the final exam.