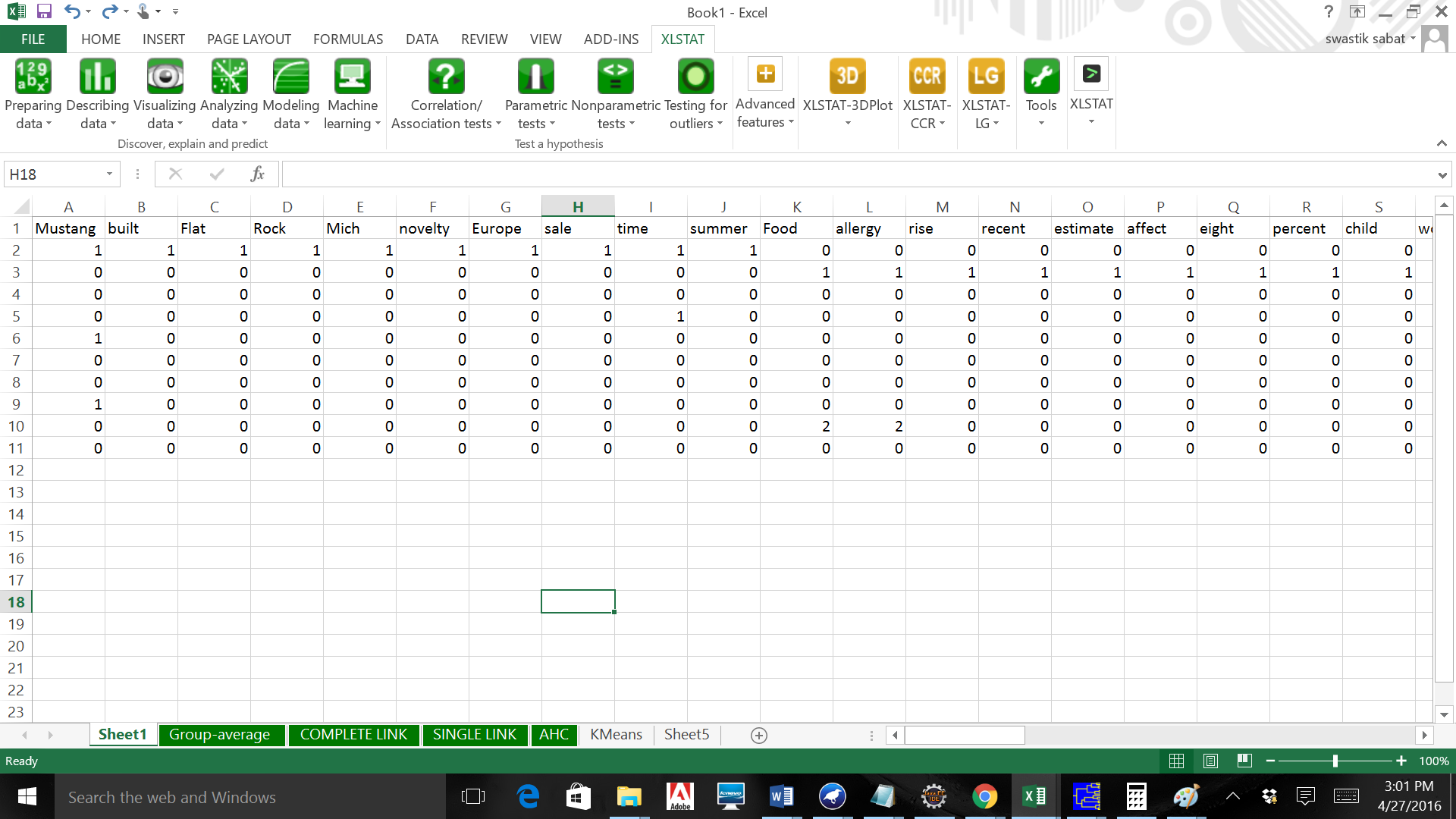
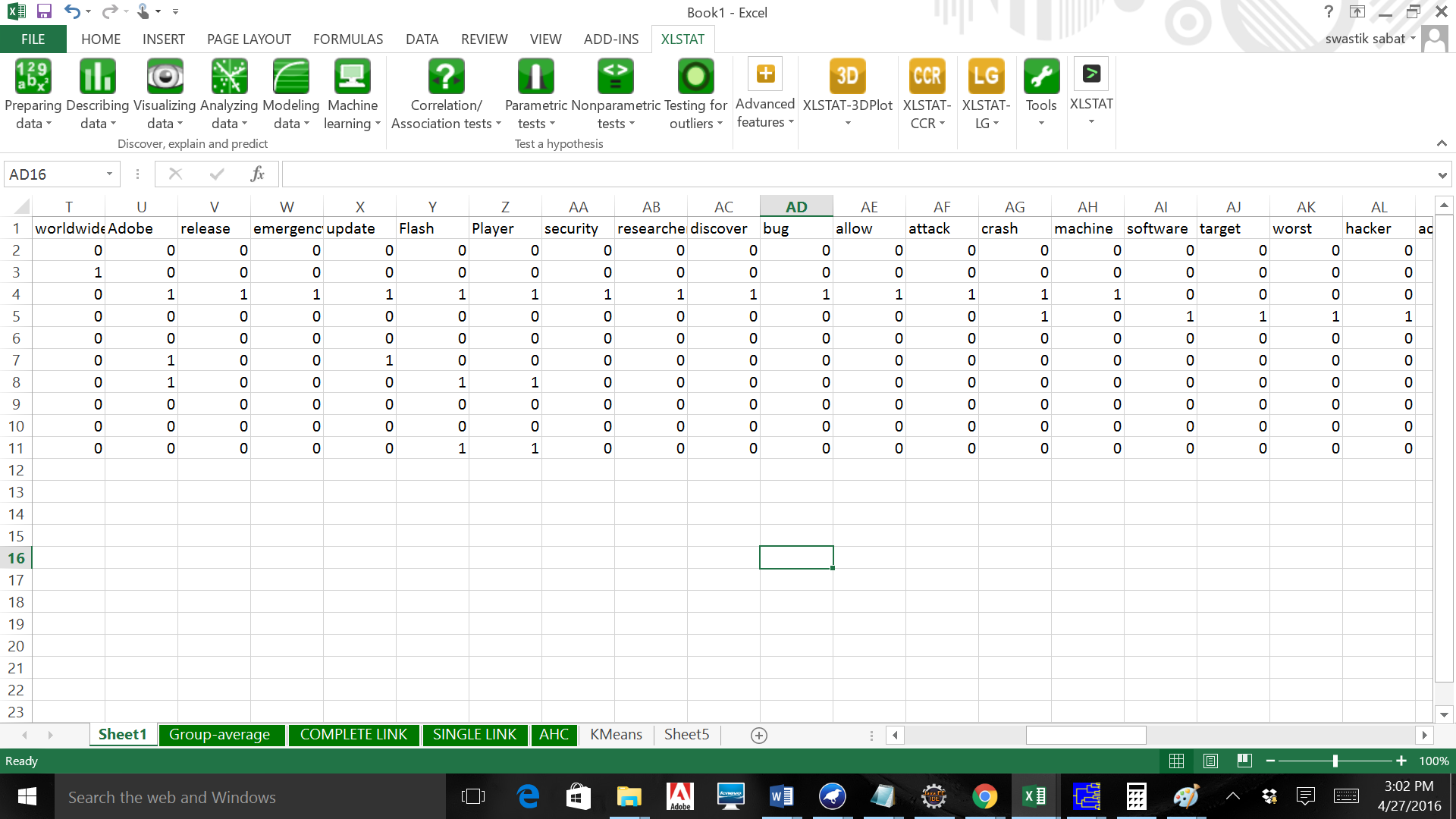
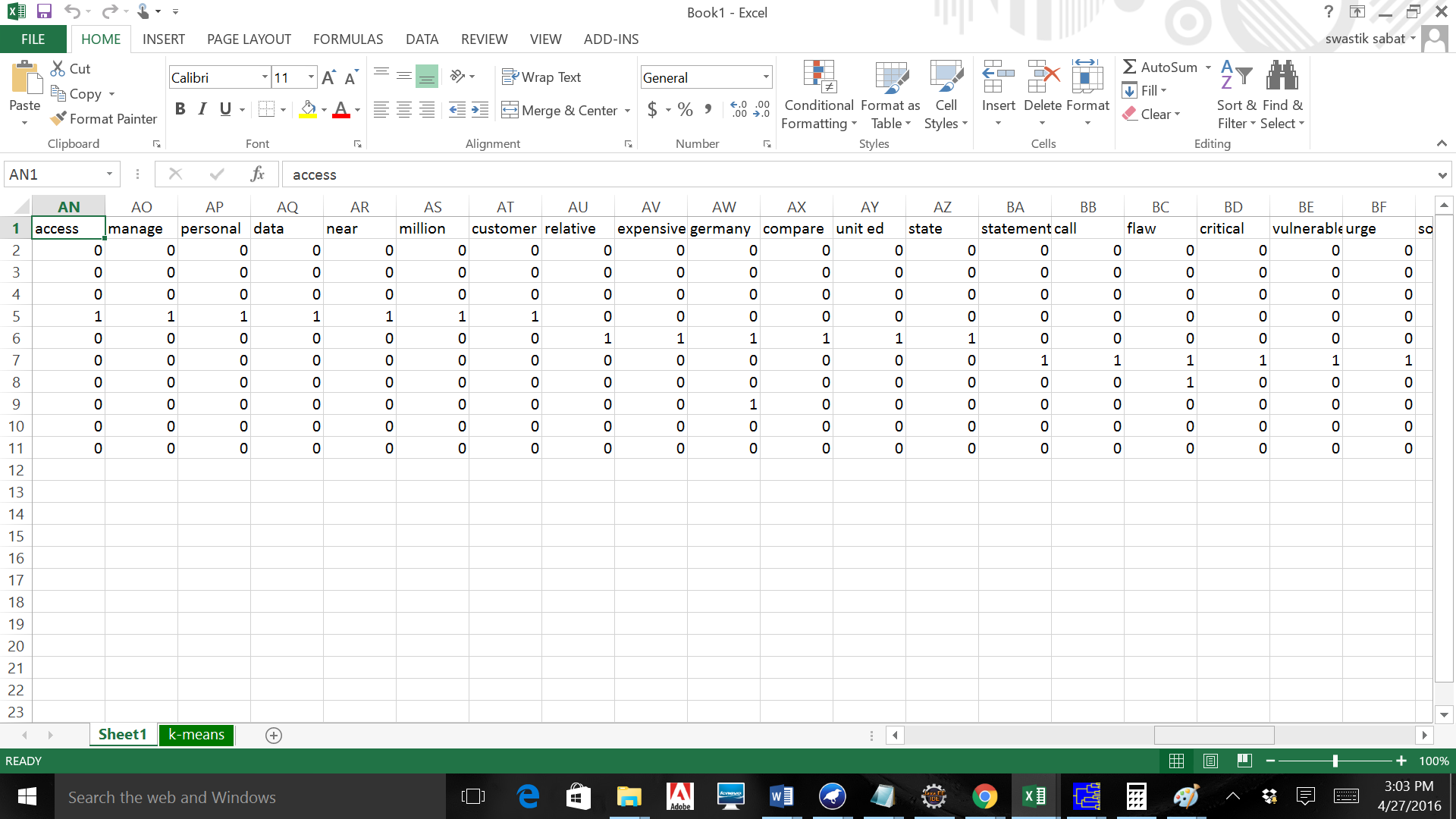
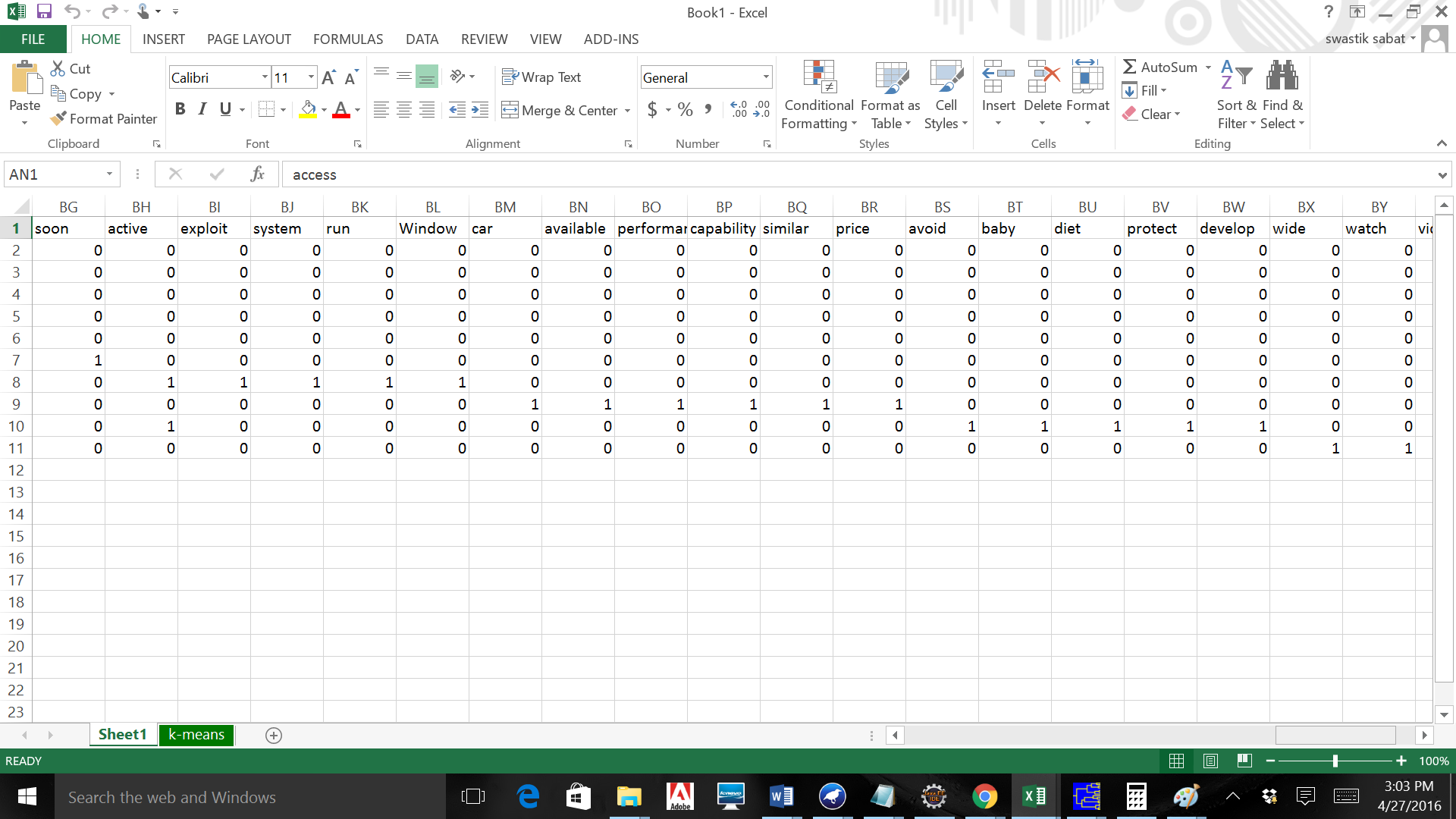
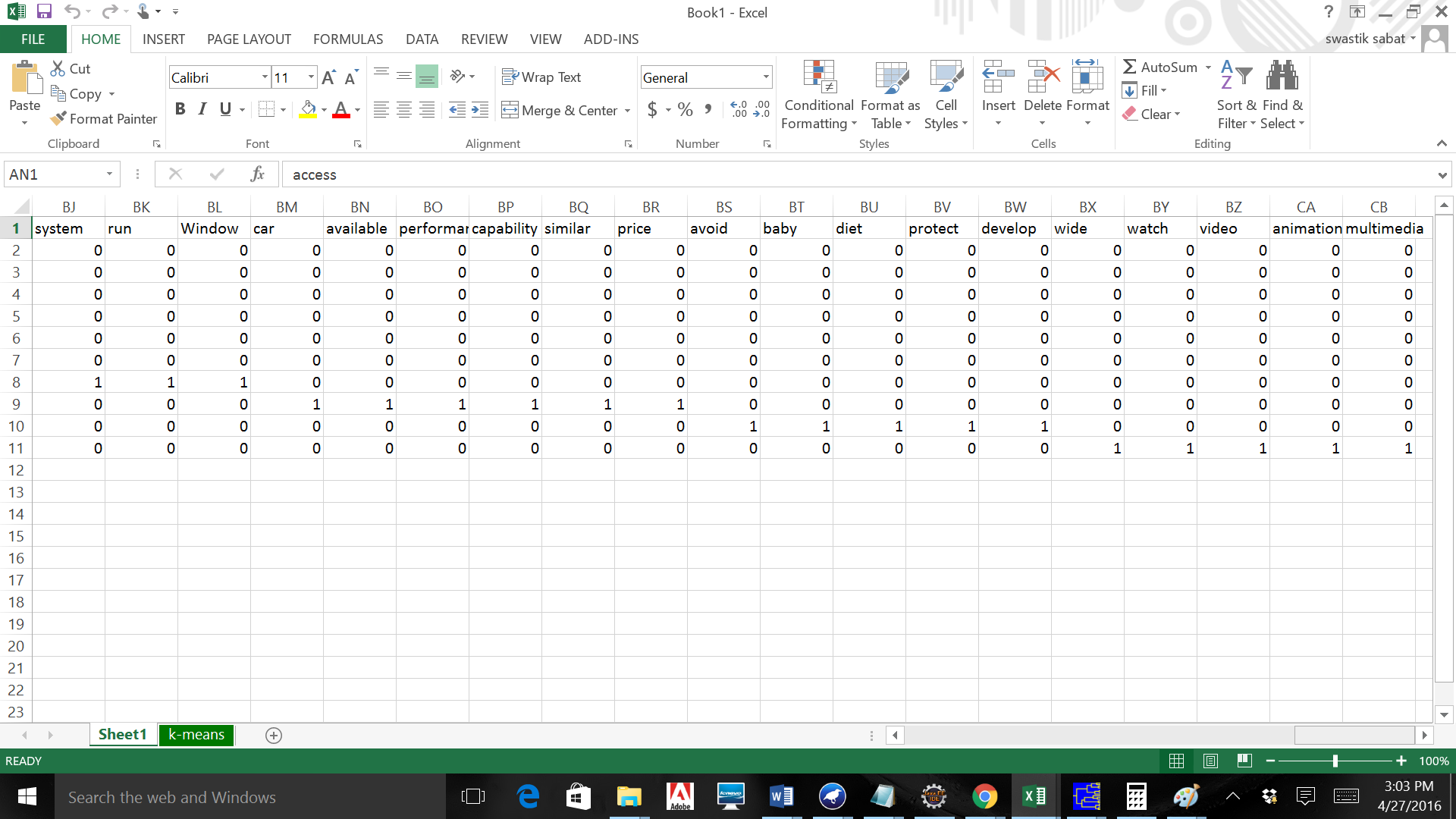
1)

1. 
2.    

Centroid of Cluster 1:

Size of Cluster 1: 1

Docs in Cluster 1: D4

Centroid:

Centroid of Cluster 2:

Size of Cluster 2: 8

Docs in Cluster 1: D1,D2,D5,D6,D7,D8,D9,D10

Centroid:

Centroid of Cluster 3:

Size of Cluster 3: 1

Docs in Cluster 1: D3

Centroid:

K-means is done using Weka as the tool with 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.

Instances: 10

Attributes: 79

Mustang

built

Flat

Rock

Mich

novelty

Europe

sale

time

summer

Food

allergy

rise

recent

estimate

affect

eight

percent

child

worldwide

Adobe

release

emergency

update

Flash

Player

security

researcher

discover

bug

allow

attack

crash

machine

software

target

worst

hacker

access

manage

personal

data

near

million

customer

relative

expensive

germany

compare

unit ed

state

statement

call

flaw

critical

vulnerable

urge

soon

active

exploit

system

run

Window

car

available

performance

capability

similar

price

avoid

baby

diet

protect

develop

wide

watch

video

animation

multimedia

Test mode: evaluate on training data

=== Clustering model (full training set) ===

kMeans

======

Number of iterations: 2

Within cluster sum of squared errors: 55.6875

Initial starting points (random):

Cluster 0: 0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0

Cluster 1: 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0

Cluster 2: 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0

Missing values globally replaced with mean/mode

Final cluster centroids:

Cluster#

Attribute Full Data 0 1 2

(10.0) (1.0) (8.0) (1.0)

=========================================================

Mustang 0.3 0 0.375 0

built 0.1 0 0.125 0

Flat 0.1 0 0.125 0

Rock 0.1 0 0.125 0

Mich 0.1 0 0.125 0

novelty 0.1 0 0.125 0

Europe 0.1 0 0.125 0

sale 0.1 0 0.125 0

time 0.2 1 0.125 0

summer 0.1 0 0.125 0

Food 0.3 0 0.375 0

allergy 0.3 0 0.375 0

rise 0.1 0 0.125 0

recent 0.1 0 0.125 0

estimate 0.1 0 0.125 0

affect 0.1 0 0.125 0

eight 0.1 0 0.125 0

percent 0.1 0 0.125 0

child 0.1 0 0.125 0

worldwide 0.1 0 0.125 0

Adobe 0.3 0 0.25 1

release 0.1 0 0 1

emergency 0.1 0 0 1

update 0.2 0 0.125 1

Flash 0.3 0 0.25 1

Player 0.3 0 0.25 1

security 0.1 0 0 1

researcher 0.1 0 0 1

discover 0.1 0 0 1

bug 0.1 0 0 1

allow 0.1 0 0 1

attack 0.2 1 0 1

crash 0.1 0 0 1

machine 0.1 0 0 1

software 0.1 1 0 0

target 0.1 1 0 0

worst 0.1 1 0 0

hacker 0.1 1 0 0

access 0.1 1 0 0

manage 0.1 1 0 0

personal 0.1 1 0 0

data 0.1 1 0 0

near 0.1 1 0 0

million 0.1 1 0 0

customer 0.1 1 0 0

relative 0.1 0 0.125 0

expensive 0.1 0 0.125 0

germany 0.2 0 0.25 0

compare 0.1 0 0.125 0

unit ed 0.1 0 0.125 0

state 0.1 0 0.125 0

statement 0.1 0 0.125 0

call 0.1 0 0.125 0

flaw 0.2 0 0.25 0

critical 0.1 0 0.125 0

vulnerable 0.1 0 0.125 0

urge 0.1 0 0.125 0

soon 0.1 0 0.125 0

active 0.2 0 0.25 0

exploit 0.1 0 0.125 0

system 0.1 0 0.125 0

run 0.1 0 0.125 0

Window 0.1 0 0.125 0

car 0.1 0 0.125 0

available 0.1 0 0.125 0

performance 0.1 0 0.125 0

capability 0.1 0 0.125 0

similar 0.1 0 0.125 0

price 0.1 0 0.125 0

avoid 0.1 0 0.125 0

baby 0.1 0 0.125 0

diet 0.1 0 0.125 0

protect 0.1 0 0.125 0

develop 0.1 0 0.125 0

wide 0.1 0 0.125 0

watch 0.1 0 0.125 0

video 0.1 0 0.125 0

animation 0.1 0 0.125 0

multimedia 0.1 0 0.125 0

Time taken to build model (full training data) : 0 seconds

=== Model and evaluation on training set ===

Clustered Instances

0 1 ( 10%)

1 8 ( 80%)

2 1 ( 10%)

B. (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}

Cluster 1: D4

Cluster 2: D1,D2,D5,D6,D7,D8,D9,D10

Cluster 3: D3

|  |  |  |  |
| --- | --- | --- | --- |
| Classes | Cluster1 | Cluster2 | Cluster3 |
| CARS | 0 | 3 | 0 |
| SOFTWARE | 1 | 3 | 1 |
| ALLERGIES | 0 | 2 | 0 |

Purity=5/10=.5

NMI

Cluster 1: D4

Cluster 2: D1,D2,D5,D6,D7,D8,D9,D10

Cluster 3: D3

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

|  |  |  |  |
| --- | --- | --- | --- |
| Cluster | Docs |  |  |
| W1 | 1 |  |  |
| W2 | 8 |  |  |
| W3 | 1 |  |  |

|  |  |
| --- | --- |
| Class | Docs |
| C1 | 3 |
| C2 | 5 |
| C3 | 2 |

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

**For Cluster**

P(W1)= 1/10

P(W2)= 8/10

P(W3)= 1/10

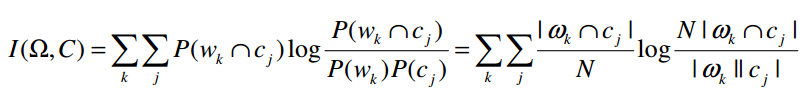
**For Class**

P(C1)= 3/10

P(C2)= 5/10

P(C3)= 2/10

Mutual Information I =



=.507

H=.361

NMI=.507/.361=1.404

*Rand Index*

*TP=7*

FN=7

TP + FP=8c2=28

FP=28-7=21

TN:45-(28+7)=10

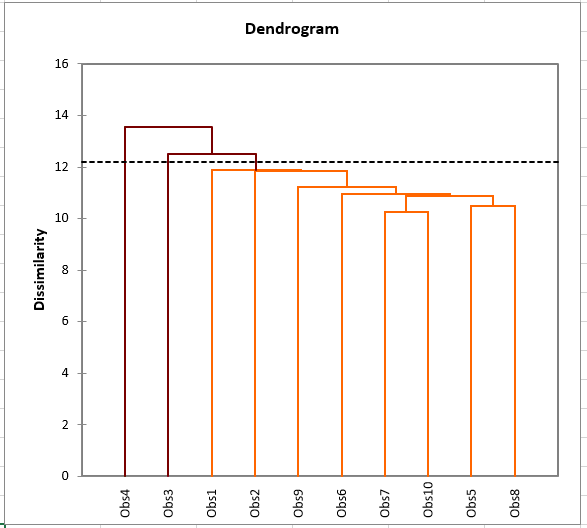
RI=TP+TN/TP+FP+FN+TN =17/45

C. (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.

For all 3 clustering techniques mentioned below, I have used XLSTAT . **COSINE SIMILARITY** is used as the proximity factor.

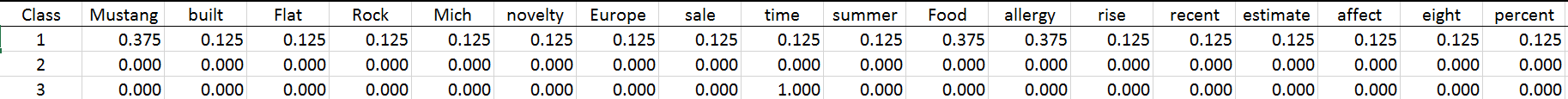
1. A) Single-link agglomerative clustering

|  |  |  |
| --- | --- | --- |
|  | XLSTAT 2016.02.27955 - Agglomerative hierarchical clustering (AHC) - Start time: 4/27/2016 at 11:29:11 AM |  |
|  | Observations/variables table: Workbook = Book1 / Sheet = Sheet1 / Range = Sheet1!$A$1:$CB$11 / 10 rows and 80 columns | |



Centroids:

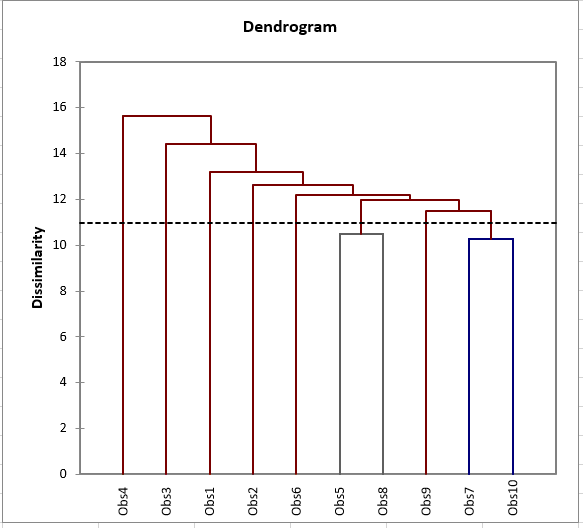
Using XLStat, i got following class centroids:



**Centroid = (sum of all values of instances)/total number of documents**

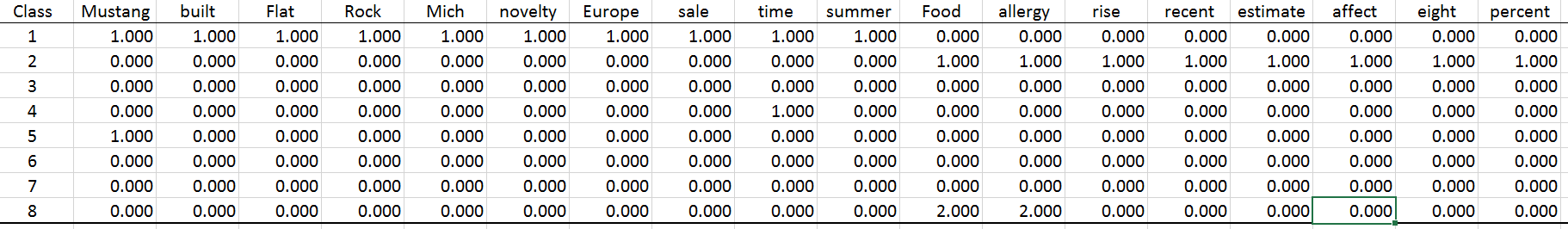
|  |  |
| --- | --- |
| **Centroid of cluster 1** | .97 |
| **Centroid of cluster 2** | 14 |
| **Centroid of cluster 3** | 10 |

Complete-link agglomerative clustering



Centroids:

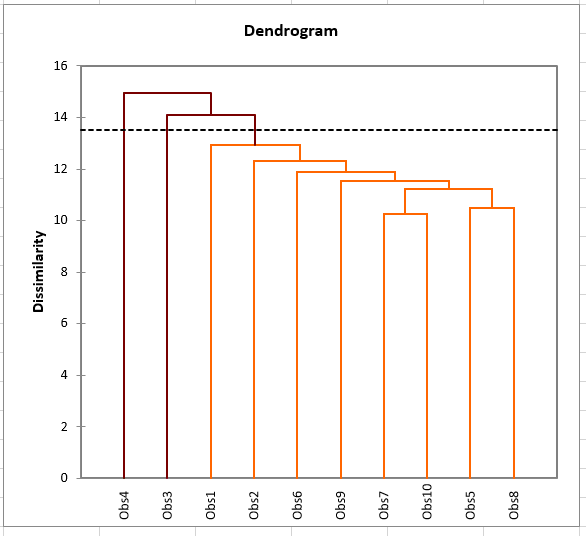
Using XLStat, we get following class centroids:

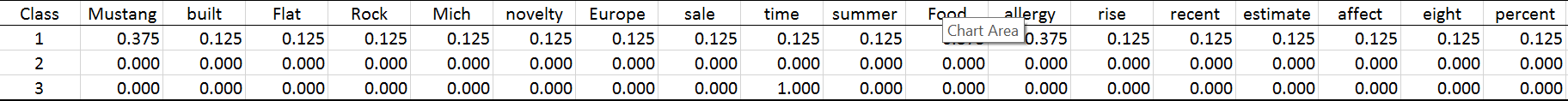


**Centroid = (sum of all values of instances)/total number of documents**

|  |  |
| --- | --- |
| **Centroid of cluster 1** | 10 |
| **Centroid of cluster 2** | 10 |
| **Centroid of cluster 3** | 14 |
| **Centroid of cluster 4** | 11 |
| **Centroid of cluster 5** | 2.5 |
| **Centroid of cluster 6** | 4.5 |
| **Centroid of cluster 7** | 2 |
| **Centroid of cluster 8** | 3 |

*( c)* Group-average link agglomerative clustering





**Centroid = (sum of all values of instances)/total number of documents**

|  |  |
| --- | --- |
| **Centroid of cluster 1** | .97 |
| **Centroid of cluster 2** | 14 |
| **Centroid of cluster 3** | 6.5 |

D. (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*

|  |  |  |  |
| --- | --- | --- | --- |
| Node | Source | Target | Outdegree |
| D1 | D3, D7, D8, D10 | D2, D3, D5, D7 | 4 |
| D2 | D1, D6, D9 | D4, D7 | 2 |
| D3 | D1, D6, D7, D10 | D1, D6, D10 | 3 |
| D4 | D2 | D5, D6, D9 | 3 |
| D5 | D1, D4, D8, D9 | D6, D7, D9, D10 | 4 |
| D6 | D3, D4, D5, D10 | D2, D3, D8 | 3 |
| D7 | D1, D2, D5 | D1, D3, D9 | 3 |
| D8 | D6 | D1, D5, D10 | 3 |
| D9 | D4, D5, D7 | D2, D5, D10 | 3 |
| D10 | D3, D5, D8, D9 | D1, D3, D6 | 3 |

PageRank or PR(A) can be calculated using a simple iterative algorithm,

Where,

* d = 0.85
* T1, T2, … Tn are the source node pointing to Node D, and
* C(T1) is the out degree of T1.

We have N = 10

|  |  |  |  |
| --- | --- | --- | --- |
| Node | Source | Outdegree | PageRanks |
| 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, D9 | 4 | PR(D5) = 0.15 + 0.85(PR(D1)/4 + PR(D4)/3 + PR(D8)/3 + PR(D9)/3) |
| D6 | D3, D4, D5, D10 | 3 | PR(D6) = 0.15 + 0.85(PR(D3)/3 + PR(D4)/3 + PR(D5)/4 + 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) |

Fix point computation, Page ranks are calculated in iterations.Iterations are performed till pages ranks converge.

Here, we have total number of pages N = 10, thus

Initial Pagerank Scores are as follows:

|  |  |  |
| --- | --- | --- |
| Node | PageRanks | PR(i=1) |
| D1 | PR(D1) = 0.15 + 0.85((1/10)/3 + (1/10)/3 + (1/10)/3 + (1/10)/3) | 0.263333 |
| D2 | PR(D2) = 0.15 + 0.85((1/10)/4 + (1/10)/3 + (1/10)/3) | 0.227917 |
| D3 | PR(D3) = 0.15 + 0.85((1/10)/4 + (1/10)/3 + (1/10)/3 + (1/10)/3) | 0.25625 |
| D4 | PR(D4) = 0.15 + 0.85((1/10)/2) | 0.1925 |
| D5 | PR(D5) = 0.15 + 0.85((1/10)/4 + (1/10)/3 + (1/10)/3 + (1/10)/3) | 0.25625 |
| D6 | PR(D6) = 0.15 + 0.85((1/10)/3 + (1/10)/3 + (1/10)/4 + (1/10)/3) | 0.25625 |
| D7 | PR(D7) = 0.15 + 0.85((1/10)/4 + (1/10)/2 + (1/10)/4) | 0.235 |
| D8 | PR(D8) = 0.15 + 0.85((1/10)/3) | 0.178333 |
| D9 | PR(D9) = 0.15 + 0.85((1/10)/3 + (1/10)/4 + (1/10)/3) | 0.227917 |
| D10 | PR(D10) = 0.15 + 0.85((1/10)/3 + (1/10)/4 + (1/10)/3 + (1/10)/3) | 0.25625 |

We will now continue with further iterations till the ranks converge with convergence factor of 0.001

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Node | **PR(i=1)** | **PR(i=2)** | **PR(i=3)** | **PR(i=4)** | **PR(i=5)** | **PR(i=6)** | **PR(i=7)** | **PR(i=8)** | **PR(i=9)** |
| D1 | 0.263333 | 0.412319 | 0.543774 | 0.658802 | 0.756053 | 0.838657 | 0.908942 | 0.968666 | 1.019431 |
| D2 | 0.227917 | 0.343139 | 0.444389 | 0.529116 | 0.601157 | 0.66248 | 0.71457 | 0.758851 | 0.796493 |
| D3 | 0.25625 | 0.41775 | 0.564483 | 0.689419 | 0.79558 | 0.885797 | 0.962486 | 1.027678 | 1.083089 |
| D4 | 0.1925 | 0.246865 | 0.295834 | 0.338865 | 0.374874 | 0.405492 | 0.431554 | 0.453692 | 0.472512 |
| D5 | 0.25625 | 0.375604 | 0.462881 | 0.537934 | 0.602301 | 0.656801 | 0.703154 | 0.742558 | 0.776047 |
| D6 | 0.25625 | 0.404203 | 0.529236 | 0.634776 | 0.724769 | 0.801134 | 0.866045 | 0.921234 | 0.968138 |
| D7 | 0.235 | 0.357276 | 0.463268 | 0.55278 | 0.629181 | 0.694142 | 0.749339 | 0.796262 | 0.836147 |
| D8 | 0.178333 | 0.222604 | 0.264524 | 0.29995 | 0.329853 | 0.355351 | 0.376988 | 0.395379 | 0.411016 |
| D9 | 0.227917 | 0.325578 | 0.400989 | 0.463441 | 0.516944 | 0.562471 | 0.601133 | 0.634006 | 0.661947 |
| D10 | 0.25625 | 0.392161 | 0.503497 | 0.596861 | 0.67594 | 0.743329 | 0.800596 | 0.849259 | 0.890628 |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Node | **PR(i=10)** | **PR(i=11)** | **PR(i=12)** | **PR(i=13)** | **PR(i=14)** | **PR(i=15)** | **PR(i=16)** | **PR(i=17)** | **PR(i=18)** |
| D1 | 1.062583 | 1.099261 | 1.130438 | 1.156938 | 1.179463 | 1.19861 | 1.214884 | 1.228718 | 1.240476 |
| D2 | 0.828487 | 0.855682 | 0.878798 | 0.898446 | 0.915147 | 0.929343 | 0.94141 | 0.951667 | 0.960385 |
| D3 | 1.130188 | 1.170222 | 1.204252 | 1.233176 | 1.257763 | 1.278661 | 1.296424 | 1.311523 | 1.324357 |
| D4 | 0.488509 | 0.502107 | 0.513665 | 0.523489 | 0.53184 | 0.538938 | 0.544971 | 0.550099 | 0.554458 |
| D5 | 0.804514 | 0.82871 | 0.849278 | 0.86676 | 0.88162 | 0.89425 | 0.904987 | 0.914112 | 0.921869 |
| D6 | 1.008008 | 1.041898 | 1.070703 | 1.095189 | 1.116001 | 1.133691 | 1.148728 | 1.16151 | 1.172374 |
| D7 | 0.870048 | 0.898865 | 0.923359 | 0.944179 | 0.961875 | 0.976918 | 0.989704 | 1.000572 | 1.00981 |
| D8 | 0.424306 | 0.435602 | 0.445204 | 0.453366 | 0.460303 | 0.4662 | 0.471213 | 0.475473 | 0.479094 |
| D9 | 0.685696 | 0.705884 | 0.723043 | 0.737628 | 0.750026 | 0.760563 | 0.769521 | 0.777134 | 0.783606 |
| D10 | 0.925791 | 0.95568 | 0.981085 | 1.00268 | 1.021035 | 1.036637 | 1.049898 | 1.061171 | 1.070753 |

|  |  |  |
| --- | --- | --- |
| Node | **PR(i=19)** | **PR(i=20)** |
| D1 | 1.250471 | 1.258966 |
| D2 | 0.967795 | 0.974094 |
| D3 | 1.335266 | 1.344539 |
| D4 | 0.558164 | 0.561313 |
| D5 | 0.928463 | 0.934067 |
| D6 | 1.181608 | 1.189458 |
| D7 | 1.017662 | 1.024336 |
| D8 | 0.482173 | 0.484789 |
| D9 | 0.789107 | 0.793782 |
| D10 | 1.078897 | 1.08582 |

Here observe that, by iteration i=20, the page ranks are converging.

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

*SOLUTION*

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Node | **h(i=1)** | **h(i=2)** | **h(i=3)** | **h(i=4)** | **h(i=5)** | **h(i=6)** | **h(i=7)** | **h(i=8)** | **h(i=9)** | **h(i=10)** |
| D1 | 0.402015 | 0.393314 | 0.400363 | 0.378964 | 0.381257 | 0.370523 | 0.37157 | 0.366605 | 0.367088 | 0.3648 |
| D2 | 0.201008 | 0.112376 | 0.104443 | 0.086346 | 0.08621 | 0.080975 | 0.081233 | 0.079494 | 0.079637 | 0.079011 |
| D3 | 0.301511 | 0.337127 | 0.330735 | 0.347784 | 0.345584 | 0.353347 | 0.352497 | 0.355991 | 0.355638 | 0.357218 |
| D4 | 0.301511 | 0.309033 | 0.313328 | 0.319002 | 0.320316 | 0.32247 | 0.322715 | 0.323677 | 0.323696 | 0.324167 |
| D5 | 0.402015 | 0.393314 | 0.400363 | 0.398153 | 0.399094 | 0.400173 | 0.400274 | 0.401428 | 0.401395 | 0.402125 |
| D6 | 0.301511 | 0.224751 | 0.226292 | 0.206272 | 0.20735 | 0.198962 | 0.199471 | 0.195573 | 0.195828 | 0.193987 |
| D7 | 0.301511 | 0.309033 | 0.304624 | 0.307009 | 0.305452 | 0.306316 | 0.305734 | 0.306073 | 0.305852 | 0.305998 |
| D8 | 0.301511 | 0.337127 | 0.330735 | 0.345385 | 0.344098 | 0.349257 | 0.348948 | 0.350814 | 0.350723 | 0.351432 |
| D9 | 0.301511 | 0.309033 | 0.313328 | 0.311806 | 0.313627 | 0.311632 | 0.312387 | 0.311058 | 0.311382 | 0.310641 |
| D10 | 0.301511 | 0.337127 | 0.330735 | 0.33819 | 0.335923 | 0.338624 | 0.337796 | 0.338945 | 0.33863 | 0.339141 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Node | **h(i=11)** | **h(i=12)** | **h(i=13)** | **h(i=14)** | **h(i=15)** |
| D1 | 0.365022 | 0.363963 | 0.364065 | 0.363573 | 0.36362 |
| D2 | 0.079074 | 0.078834 | 0.07886 | 0.078763 | 0.078774 |
| D3 | 0.357064 | 0.357786 | 0.357717 | 0.358049 | 0.358018 |
| D4 | 0.324152 | 0.324386 | 0.324373 | 0.324488 | 0.32448 |
| D5 | 0.402086 | 0.402484 | 0.402458 | 0.402662 | 0.402647 |
| D6 | 0.194117 | 0.193247 | 0.193312 | 0.192902 | 0.192934 |
| D7 | 0.305912 | 0.305978 | 0.305945 | 0.305976 | 0.305963 |
| D8 | 0.351399 | 0.351682 | 0.351669 | 0.351787 | 0.351781 |
| D9 | 0.310782 | 0.3104 | 0.310462 | 0.310272 | 0.3103 |
| D10 | 0.339017 | 0.339249 | 0.3392 | 0.339306 | 0.339285 |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Node | **a(i=1)** | **a(i=2)** | **a(i=3)** | **a(i=4)** | **a(i=5)** | **a(i=6)** | **a(i=7)** | **a(i=8)** | **a(i=9)** | **a(i=10)** |
| D1 | 0.383131 | 0.356506 | 0.38596 | 0.379098 | 0.390802 | 0.388685 | 0.39342 | 0.392675 | 0.394658 | 0.394372 |
| D2 | 0.287348 | 0.297088 | 0.270993 | 0.274782 | 0.261936 | 0.263464 | 0.257246 | 0.257923 | 0.254937 | 0.255247 |
| D3 | 0.383131 | 0.386214 | 0.369536 | 0.368921 | 0.359286 | 0.35917 | 0.354556 | 0.354602 | 0.352435 | 0.352494 |
| D4 | 0.095783 | 0.059418 | 0.032848 | 0.030531 | 0.025213 | 0.025174 | 0.023641 | 0.023717 | 0.023208 | 0.02325 |
| D5 | 0.383131 | 0.386214 | 0.394172 | 0.396908 | 0.395705 | 0.396932 | 0.395271 | 0.395783 | 0.394754 | 0.39497 |
| D6 | 0.383131 | 0.386214 | 0.402383 | 0.401997 | 0.409712 | 0.409085 | 0.413002 | 0.412617 | 0.414574 | 0.414375 |
| D7 | 0.287348 | 0.297088 | 0.262781 | 0.264605 | 0.25213 | 0.253047 | 0.248649 | 0.249061 | 0.247431 | 0.247605 |
| D8 | 0.095783 | 0.089126 | 0.065695 | 0.066151 | 0.060231 | 0.060549 | 0.058088 | 0.058237 | 0.057096 | 0.057171 |
| D9 | 0.287348 | 0.297088 | 0.295629 | 0.297681 | 0.299055 | 0.299272 | 0.300408 | 0.300343 | 0.301047 | 0.300979 |
| D10 | 0.383131 | 0.386214 | 0.402383 | 0.401997 | 0.409712 | 0.409519 | 0.412942 | 0.412858 | 0.414355 | 0.414311 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Node | **a(i=11)** | **a(i=12)** | **a(i=13)** | **a(i=14)** |
| D1 | 0.395229 | 0.395114 | 0.395493 | 0.395445 |
| D2 | 0.253824 | 0.253968 | 0.253292 | 0.253359 |
| D3 | 0.351478 | 0.35152 | 0.351043 | 0.351067 |
| D4 | 0.023067 | 0.023085 | 0.023015 | 0.023023 |
| D5 | 0.394427 | 0.394519 | 0.394248 | 0.394289 |
| D6 | 0.415333 | 0.415236 | 0.415698 | 0.415653 |
| D7 | 0.246965 | 0.247037 | 0.246773 | 0.246803 |
| D8 | 0.056633 | 0.056671 | 0.056417 | 0.056436 |
| D9 | 0.30137 | 0.301329 | 0.301532 | 0.301511 |
| D10 | 0.414973 | 0.414948 | 0.415245 | 0.415232 |

|  |  |  |
| --- | --- | --- |
| Node | Hub Scores | Authority Scores |
| D1 | 0.36362 | 0.395445 |
| D2 | 0.078774 | 0.253359 |
| D3 | 0.358018 | 0.351067 |
| D4 | 0.32448 | 0.023023 |
| D5 | 0.402647 | 0.394289 |
| D6 | 0.192934 | 0.415653 |
| D7 | 0.305963 | 0.246803 |
| D8 | 0.351781 | 0.056436 |
| D9 | 0.3103 | 0.301511 |
| D10 | 0.339285 | 0.415232 |

F. (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.

**Topic sensitive Page ranks**

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

M is built as: If there is a link from page j to page I, then Mij has the value (1/Cij) and else entries are 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 | ½ | 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 |

**Problem 2 (25 points) :**

*A. (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.co*](http://www.cnn.com/)*m*  [*money.cnn.co*](http://www.bbc.com/)*m*  [*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/~mannin*](http://www.cs.stanford.edu/~manning)*g*  [*www.cs.stanford.edu/~n*](http://www.cs.stanford.edu/~ng)*g*  [*http://machinelearning.cis.cornell.edu/pages/people.ph*](http://machinelearning.cis.cornell.edu/pages/people.php)*p*  [*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.co*](http://www.macys.com/)*m*

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

*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*:

In Mercator scheme, front queue ensures the prioritization of URLs whereas back queue ensures the politeness.

**Front Queues**

Suppose we have 3 front queues FQ1, FQ2 and FQ3 in which FQ1 has highest priority and FQ3 has lowest priority. The URL which require more frequent refreshing should be given more priority (so they will be placed in FQ1) and URL that require less crawling should be given lesser priority (so they will be placed in FQ3).

|  |  |  |
| --- | --- | --- |
| **FQ1** | **FQ2** | **FQ3** |
| [www.cnn.com](http://www.cnn.com/) | [*http://shop.nordstrom.com*](http://shop.nordstrom.com/)*/* | [*www.cs.stanford.edu/~mannin*](http://www.cs.stanford.edu/~mannin)*g* |
| [money.cnn.com](http://www.bbc.com/) | [*http://www.newbalance.com*](http://www.newbalance.com)*/* | [*www.cs.stanford.edu/~n*](http://www.cs.stanford.edu/~n)*g* |
| [*www.macys.co*](http://www.macys.com/)*m* | [*www.cs.stanford.edu*](http://www.cs.stanford.edu/)*/* | [*http://machinelearning.cis.cornell.edu/pages/people.ph*](http://machinelearning.cis.cornell.edu/pages/people.ph)*p* |
| [*http://www.amazon.co*](http://www.amazon.co)*m* | [*www.cs.cornell.edu*](http://www.cs.cornell.edu/)*/* | [*http://www.cs.cornell.edu/home/kleinber*](http://www.cs.cornell.edu/home/kleinber)*/* |
|  | [*www.cnn.edu/world*](http://www.cnn.edu/world/)*/* | [*http://www.newbalance.com/women/clothing/short-sleeve-shirts-1*](http://www.newbalance.com/women/clothing/short-sleeve-shirts-1)*/* |

**Back Queues**

To pass URL to the Back Queue from the Front Queue, URL from the queue with highest priority should be pulled out at higher rate than the rest of the queues. So, the order will be:

{FQ1}

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

Assigning separate queue for each host:

**BQ1:**

[*www.cnn.co*](http://www.cnn.com/)*m*  [*money.cnn.co*](http://www.bbc.com/)*m*  [*www.cnn.edu/world*](http://www.cnn.edu/world/)*/*

BQ2

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

BQ3

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

BQ4

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

BQ5

[*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/)*/*

BQ6

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

[*www.cs.stanford.edu/~mannin*](http://www.cs.stanford.edu/~manning)*g*  [*www.cs.stanford.edu/~n*](http://www.cs.stanford.edu/~ng)*g*  [*http://machinelearning.cis.cornell.edu/pages/people.ph*](http://machinelearning.cis.cornell.edu/pages/people.php)*p*  [*http://www.cs.cornell.edu/home/kleinber*](http://www.cs.cornell.edu/home/kleinber/)*/*

SOLUTION:

**Shingles for doc1:**

The class

class will

will cover

cover link

link analysis

**Shingles for doc2:**

analysis techniques

techniques shall

shall be

be presented

presented in

in the

|  |  |
| --- | --- |
| **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’s 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%

=> Documents are not similar

The following results are generated by 25 random permutations with python program:

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 | 11 |
| **P2** | 4 | 8 |
| **P3** | 1 | 9 |
| **P4** | 5 | 10 |
| **P5** | 5 | 10 |
| **P6** | 2 | 8 |
| **P7** | 1 | 1 |
| **P8** | 1 | 5 |
| **P9** | 4 | 8 |
| **P10** | 4 | 10 |
| **P11** | 4 | 12 |
| **P12** | 4 | 12 |
| **P13** | 1 | 7 |
| **P14** | 2 | 10 |
| **P15** | 3 | 11 |
| **P16** | 4 | 10 |
| **P17** | 2 | 13 |
| **P18** | 2 | 12 |
| **P19** | 1 | 1 |
| **P20** | 2 | 11 |
| **P21** | 5 | 8 |
| **P22** | 5 | 12 |
| **P23** | 2 | 10 |
| **P24** | 5 | 8 |
| **P25** | 2 | 10 |

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

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

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 | 3  2 |
| **J = 6** | h1(6)=3  h2(6)=2 | 0  0 | 3  2 |
| **J = 7** | h1(7)=4  h2(7)=3 | 0  0 | 3  2 |
| **J = 8** | h1(8)=0  h2(8)=4 | 0  0 | 0  2 |
| **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 |
| **J = 12** | h1(12)=4  h2(12)=3 | 0  0 | 0  0 |
| **J = 13** | h1(13)=0  h2(13)=4 | 0  0 | 0  0 |

min(h1(W1)) = 0 == 0 = min(h1(W2))

min(h2(W1)) = 0 == 0 = min(h2(W2))

**Jaccard coefficient: J(W1, W2) = (1 + 1) / 2 = 1**

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*:

**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.

The Terms in the vocabulary are :

|V| = 29

1. access
2. adobe
3. allow
4. attack
5. attackers
6. bug
7. crash
8. customers
9. data
10. discovered
11. emergency
12. flash
13. hackers
14. machines
15. managed
16. million
17. nearly
18. personal
19. player
20. released
21. researchers
22. security
23. software
24. take
25. targeted
26. time
27. update
28. users
29. worst

The stems are: |S| = 28

V(attack) = {attack, attackers}

1. access
2. adobe
3. allow
4. attack
5. bug
6. crash
7. customer
8. data
9. discover
10. emergency
11. flash
12. hacker
13. machine
14. manage
15. million
16. near
17. personal
18. player
19. release
20. research
21. security
22. software
23. take
24. target
25. time
26. update
27. user
28. worst

|  |  |  |
| --- | --- | --- |
| Terms (t) | ft, D3 | ft, D4 |
| access | 0 | 1 |
| adobe | 1 | 1 |
| allow | 1 | 0 |
| attack | 1 | 1 |
| bug | 1 | 0 |
| crash | 1 | 0 |
| customer | 0 | 1 |
| data | 0 | 1 |
| discover | 1 | 0 |
| emergency | 1 | 0 |
| flash | 1 | 0 |
| hacker | 0 | 1 |
| machine | 1 | 0 |
| manage | 0 | 1 |
| million | 0 | 1 |
| near | 0 | 1 |
| personal | 0 | 1 |
| player | 1 | 0 |
| release | 1 | 0 |
| research | 1 | 0 |
| security | 1 | 0 |
| software | 0 | 1 |
| target | 0 | 1 |
| take | 1 | 0 |
| time | 0 | 1 |
| update | 1 | 0 |
| user | 1 | 0 |
| worst | 0 | 1 |

Calculating the correlations for Adobe and security

|  |  |  |
| --- | --- | --- |
| Terms (t) | Adobe  cadobe,t =  (fadobe,D3\*ft,D3+fadobe,D4\*ft,D4) | Security  Csecurity,t =  (fsecurity,D3\*ft,D3+fsecurity,D4\*ft,D4) |
| access | 1 | 0 |
| adobe | 2 | 1 |
| allow | 1 | 1 |
| attack | 2 | 1 |
| bug | 1 | 1 |
| crash | 1 | 1 |
| customer | 1 | 0 |
| data | 1 | 0 |
| discover | 1 | 1 |
| emergency | 1 | 1 |
| flash | 1 | 1 |
| hacker | 1 | 0 |
| machine | 1 | 1 |
| manage | 1 | 0 |
| million | 1 | 0 |
| near | 1 | 0 |
| personal | 1 | 0 |
| player | 1 | 1 |
| release | 1 | 1 |
| research | 1 | 1 |
| security | 1 | 1 |
| software | 1 | 0 |
| take | 1 | 1 |
| target | 1 | 0 |
| time | 1 | 0 |
| update | 1 | 1 |
| user | 1 | 1 |
| worst | 1 | 0 |

Un normalized association cluster for adobe : {access, attack}

Un normalized association cluster for security : {bug, crash}

Normalizing the correlations:

|  |  |  |
| --- | --- | --- |
| Terms (t) | Adobe  sadobe,t = cadobe,t/  (cadobe,adobe+ct,t+ cadobe,t) | Security ssecurity,t = csecurity,t/  (csecurity,security+ct,t+ csecurity,t) |
| access | 1/4 = 0.25 | 0 |
| adobe | 2/4 = 0.5 | 1/4 = 0.25 |
| allow | 1/4 = 0.25 | 1/3 = 0.33 |
| attack | 2/5 = 0.4 | 1/4 = 0.25 |
| bug | 1/4 = 0.25 | 1/3 = 0.33 |
| crash | 1/4 = 0.25 | 1/3 = 0.33 |
| customer | 1/4 = 0.25 | 0 |
| data | 1/4 = 0.25 | 0 |
| discover | 1/4 = 0.25 | 1/3 = 0.33 |
| emergency | 1/4 = 0.25 | 1/3 = 0.33 |
| flash | 1/4 = 0.25 | 1/3 = 0.33 |
| hacker | 1/4 = 0.25 | 0 |
| machine | 1/4 = 0.25 | 1/3 = 0.33 |
| manage | 1/4 = 0.25 | 0 |
| million | 1/4 = 0.25 | 0 |
| near | 1/4 = 0.25 | 0 |
| personal | 1/4 = 0.25 | 0 |
| player | 1/4 = 0.25 | 1/3 = 0.33 |
| release | 1/4 = 0.25 | 1/3 = 0.33 |
| research | 1/4 = 0.25 | 1/3 = 0.33 |
| security | 1/4 = 0.25 | 1/3 = 0.33 |
| software | 1/4 = 0.25 | 0 |
| take | 1/4 = 0.25 | 1/3 = 0.33 |
| target | 1/4 = 0.25 | 0 |
| time | 1/4 = 0.25 | 0 |
| update | 1/4 = 0.25 | 1/3 = 0.33 |
| user | 1/4 = 0.25 | 1/3 = 0.33 |
| worst | 1/4 = 0.25 | 0 |

As there are many terms with the same top ranks, I have selected any 2 words for both the clusters,

Normalized association cluster for adobe : {access, attack}

Normalized association cluster for security : {bug, crash}

Expanded query : **adobe security access attack bug crash**

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 theproblem – attach the source printout of the program at the end of the exam

*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.

The Terms in the vocabulary are :

|V| = 29

1. access
2. adobe
3. allow
4. attack
5. attackers
6. bug
7. crash
8. customers
9. data
10. discovered
11. emergency
12. flash
13. hackers
14. machines
15. managed
16. million
17. nearly
18. personal
19. player
20. released
21. researchers
22. security
23. software
24. take
25. targeted
26. time
27. update
28. users
29. worst

The stems are: |S| = 28

V(attack) = {attack, attackers}

1. access
2. adobe
3. allow
4. attack
5. bug
6. crash
7. customer
8. data
9. discover
10. emergency
11. flash
12. hacker
13. machine
14. manage
15. million
16. near
17. personal
18. player
19. release
20. research
21. security
22. software
23. take
24. target
25. time
26. update
27. user
28. worst

idf3 = log(28/16) = 0.243

idf4 = log(28/13) = 0.333

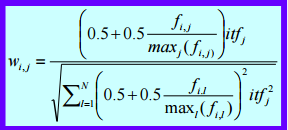
The frequency of each term in docs 3 and 4 are:

|  |  |  |  |
| --- | --- | --- | --- |
| Terms (t) | ft,3 | ft,4 | max(fi,j) |
| access | 0 | 1 | 1 |
| adobe | 1 | 1 | 1 |
| allow | 1 | 0 | 1 |
| attack | 1 | 1 | 1 |
| bug | 1 | 0 | 1 |
| crash | 1 | 0 | 1 |
| customer | 0 | 1 | 1 |
| data | 0 | 1 | 1 |
| discover | 1 | 0 | 1 |
| emergency | 1 | 0 | 1 |
| flash | 1 | 0 | 1 |
| hacker | 0 | 1 | 1 |
| machine | 1 | 0 | 1 |
| manage | 0 | 1 | 1 |
| million | 0 | 1 | 1 |
| near | 0 | 1 | 1 |
| personal | 0 | 1 | 1 |
| player | 1 | 0 | 1 |
| release | 1 | 0 | 1 |
| research | 1 | 0 | 1 |
| security | 1 | 0 | 1 |
| software | 0 | 1 | 1 |
| take | 1 | 0 | 1 |
| target | 0 | 1 | 1 |
| time | 0 | 1 | 1 |
| update | 1 | 0 | 1 |
| user | 1 | 0 | 1 |
| worst | 0 | 1 | 1 |

The weight vectors of each term in docs 3 and 4 are:

ku = (wu,3, wu,4)

Using formula



For example:

Waccess, 3 = ((0.5+0.5\*0/1)\*0.243)/√(0.252\*0.243+0.3332)

= 0.34263

|  |  |  |
| --- | --- | --- |
| Terms (t) | wt,3 | wt,4 |
| access | 0.34263 | 0.93906 |
| adobe | 0.58806 | 0.80586 |
| allow | 0.824985 | 0.565101 |
| attack | 0.58806 | 0.80586 |
| bug | 0.824985 | 0.565101 |
| crash | 0.824985 | 0.565101 |
| customer | 0.34263 | 0.93906 |
| data | 0.34263 | 0.93906 |
| discover | 0.824985 | 0.565101 |
| emergency | 0.824985 | 0.565101 |
| flash | 0.824985 | 0.565101 |
| hacker | 0.34263 | 0.93906 |
| machine | 0.824985 | 0.565101 |
| manage | 0.34263 | 0.93906 |
| million | 0.34263 | 0.93906 |
| near | 0.34263 | 0.93906 |
| personal | 0.34263 | 0.93906 |
| player | 0.824985 | 0.565101 |
| release | 0.824985 | 0.565101 |
| research | 0.824985 | 0.565101 |
| security | 0.824985 | 0.565101 |
| software | 0.34263 | 0.93906 |
| take | 0.824985 | 0.565101 |
| target | 0.34263 | 0.93906 |
| time | 0.34263 | 0.93906 |
| update | 0.824985 | 0.565101 |
| user | 0.824985 | 0.565101 |
| worst | 0.34263 | 0.93906 |

The correlation of each term with query terms adobe and security are:

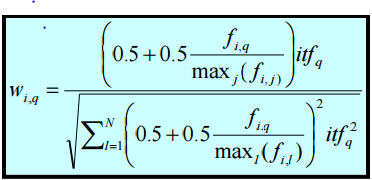
Cu,v = ku.kv

|  |  |  |
| --- | --- | --- |
| Terms (t) | cadobe,t | Csecurity,t |
| access | 0.958238 | 0.813328 |
| allow | 0.940533 | 0.999939 |
| attack | 0.995225 | 0.940533 |
| bug | 0.940533 | 0.999939 |
| crash | 0.940533 | 0.999939 |
| customer | 0.958238 | 0.813328 |
| data | 0.958238 | 0.813328 |
| discover | 0.940533 | 0.999939 |
| emergency | 0.940533 | 0.999939 |
| flash | 0.940533 | 0.999939 |
| hacker | 0.958238 | 0.813328 |
| machine | 0.940533 | 0.999939 |
| manage | 0.958238 | 0.813328 |
| million | 0.958238 | 0.813328 |
| near | 0.958238 | 0.813328 |
| personal | 0.958238 | 0.813328 |
| player | 0.940533 | 0.999939 |
| release | 0.940533 | 0.999939 |
| research | 0.940533 | 0.999939 |
| software | 0.958238 | 0.813328 |
| take | 0.940533 | 0.999939 |
| target | 0.958238 | 0.813328 |
| time | 0.958238 | 0.813328 |
| update | 0.940533 | 0.999939 |
| user | 0.940533 | 0.999939 |
| worst | 0.958238 | 0.813328 |

The query vector is:

q = wadobe, q \* k2 + wsecurity,q \* k21

Using formula



fadobe,q  = 1, fsecurtiy,q  = 1

wadobe,q = (0.5+0.5(1)/1)idfq/√((0.5+0.5(1)/1)idfq2 +0.5+0.5(1)/1)idfq2) = 1/√2 = 0.707

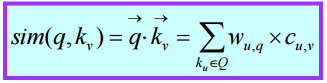
wsecurity,q = (0.5+0.5(1)/1)idfq/√((0.5+0.5(1)/1)idfq2 +0.5+0.5(1)/1)idfq2) = 1/√2 = 0.707

q = 0,707k2 + 0.707k21

kadobe = (wadobe,,3,wadobe,4) = (2.42, 2.42)

ksecurity = (wsecurity,3, wsecurity,4) = (3.395, 1.697)

Using formula



For example:

Sim(q,access) = wadobe,q\*cadobe,access +

wsecurity,q\*csecurity,access

=0.707\*0.958238 + 0.707\*0.813328

=1.252497

The similarities of each term with the query are:

|  |  |
| --- | --- |
| Terms (t) | Sim(q,t) |
| access | 1.252497 |
| allow | 1.371914 |
| attack | 1.368581 |
| bug | 1.371914 |
| crash | 1.371914 |
| customer | 1.252497 |
| data | 1.252497 |
| discover | 1.371914 |
| emergency | 1.371914 |
| flash | 1.371914 |
| hacker | 1.252497 |
| machine | 1.371914 |
| manage | 1.252497 |
| million | 1.252497 |
| near | 1.252497 |
| personal | 1.252497 |
| player | 1.371914 |
| release | 1.371914 |
| research | 1.371914 |
| software | 1.252497 |
| take | 1.371914 |
| target | 1.252497 |
| time | 1.252497 |
| update | 1.371914 |
| user | 1.371914 |
| worst | 1.252497 |

There are many terms with top rank = 1.371914

Expanding the query by picking any 2 top ranked terms

Expanded query is **adobe security bug crash**

**APPENDIX**



*Code used to resolve the problems in the final exam.*