Q1:

TermID	term	DocID
T1	windows	D1, D3, D6, D7, D8
T2	Azure	D1, D4
T3	Cloud	D1, D4, D5
T4	Microsoft	D1, D2, D4
T5	Chairman	D2
T6	Bill	D2
T7	Gates	D2, D3, D4, D7
T8	retirement	D2
T9	morning	D2, D5
T10	Success(successful)	D3, D4
T11	attributed	D3
T12	sunrise	D5
T13	garden	D5, D8
T14	wooden	D6, D7
T15	frames	D6
T16	trees	D8
T17	flowers	D8

[1]:
```
import numpy as np
from sklearn.preprocessing import normalize
from numpy import linalg
from numpy.linalg import inv
import matplotlib.pyplot as plt
```

[2]:
```
# Question a)
A = np.array([[1,0,1,0,0,1,1,1],
 ↪[1,0,0,1,0,0,0,0],[1,0,0,1,1,0,0,0],[1,1,0,1,0,0,0,0],
                 
 ↪[0,1,0,0,0,0,0,0],[0,1,0,0,0,0,0,0],[0,1,1,1,0,0,1,0],[0,1,0,0,0,0,0,0],
                 
 ↪[0,1,0,0,1,0,0,0],[0,0,1,1,0,0,0,0],[0,0,1,0,0,0,0,0],[0,0,0,0,1,0,0,0],
                 
 ↪[0,0,0,0,1,0,0,1],[0,0,0,0,0,1,2,0],[0,0,0,0,0,1,0,0],[0,0,0,0,0,0,0,1],[0,0,0,0,0,0,0,1]])
A
```

[2]:
```
array([[1, 0, 1, 0, 0, 1, 1, 1],
       [1, 0, 0, 1, 0, 0, 0, 0],
       [1, 0, 0, 1, 1, 0, 0, 0],
       [1, 1, 0, 1, 0, 0, 0, 0],
       [0, 1, 0, 0, 0, 0, 0, 0],
       [0, 1, 0, 0, 0, 0, 0, 0],
       [0, 1, 1, 1, 0, 0, 1, 0],
       [0, 1, 0, 0, 0, 0, 0, 0],
       [0, 1, 0, 0, 1, 0, 0, 0],
       [0, 0, 1, 1, 0, 0, 0, 0],
       [0, 0, 1, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 1, 0, 0, 0],
       [0, 0, 0, 0, 1, 0, 0, 1],
       [0, 0, 0, 0, 0, 1, 2, 0],
       [0, 0, 0, 0, 0, 1, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 1],
       [0, 0, 0, 0, 0, 0, 0, 1]])
```

[3]:
```
A_norm = normalize(A, axis=0)
A_norm
```

[3]:
```
array([[0.5       , 0.        , 0.5       , 0.        , 0.        ,
        0.57735027, 0.40824829, 0.5       ],
       [0.5       , 0.        , 0.        , 0.4472136 , 0.        ,
        0.        , 0.        , 0.        ],
       [0.5       , 0.        , 0.        , 0.4472136 , 0.5       ,
        0.        , 0.        , 0.        ],
       [0.5       , 0.40824829, 0.        , 0.4472136 , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.40824829, 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.40824829, 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.40824829, 0.5       , 0.4472136 , 0.        ,
        0.        , 0.40824829, 0.        ],
       [0.        , 0.40824829, 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.40824829, 0.        , 0.        , 0.5       ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.5       , 0.4472136 , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.5       , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.5       ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.5       ,
        0.        , 0.        , 0.5       ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.57735027, 0.81649658, 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.57735027, 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.5       ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.5       ]])
```

[4]:
```
# Question b)
U, s, Vh = np.linalg.svd(A_norm, full_matrices=True)
U.shape, s.shape, Vh.shape, A_norm.shape
```

[4]:
```
((17, 17), (8,), (8, 8), (17, 8))
```

[5]:
```
U
```

[5]:
```
array([[-0.57095876, -0.38420785, -0.15767464,  0.1996219 , -0.10879471,
        -0.17151627, -0.34296701,  0.39754535, -0.01845838,  0.11417005,
        -0.32350869,  0.00990825, -0.03174627, -0.14291744, -0.04150248,
        -0.04165452, -0.04165452],
```

```
[-0.24576484,  0.26368332,  0.05985029,  0.36621512,  0.24056984,
 -0.14855945,  0.10484259, -0.02437994,  0.57827043, -0.01473762,
  0.07260805,  0.28072451,  0.09914366,  0.14370107,  0.35458472,
 -0.18158086, -0.18158086],
[-0.30416853,  0.39265773, -0.23305036,  0.19086574,  0.33429677,
  0.23442813,  0.02588033,  0.03525202, -0.31848058,  0.39684824,
  0.27298021, -0.26479953, -0.14978161, -0.16368568,  0.12122159,
  0.11501792,  0.11501792],
[-0.30974153,  0.3752339 ,  0.14111185,  0.04375227,  0.18574352,
 -0.39454532, -0.00342276, -0.01407721, -0.24133147, -0.49628068,
 -0.02207957, -0.02583323,  0.08238422,  0.16290205, -0.43430383,
  0.10821745,  0.10821745],
[-0.06397669,  0.11155058,  0.08126156, -0.32246285, -0.05482632,
 -0.24598587, -0.10826535,  0.01030273,  0.07560658,  0.44783526,
  0.22030904,  0.36429972,  0.5224951 , -0.24500387, -0.16411119,
  0.15819538,  0.15819538],
[-0.06397669,  0.11155058,  0.08126156, -0.32246285, -0.05482632,
 -0.24598587, -0.10826535,  0.01030273, -0.24820225,  0.19124277,
 -0.24939393, -0.07188668,  0.0203887 ,  0.57120621,  0.5369081 ,
  0.09227539,  0.09227539],
[-0.40301335,  0.10572149,  0.37844935, -0.23475005, -0.3114008 ,
  0.16836822,  0.32837147, -0.07769607, -0.07263378, -0.23360134,
 -0.09188951, -0.01631462, -0.01105593, -0.45669924,  0.33403899,
  0.00525869,  0.00525869],
[-0.06397669,  0.11155058,  0.08126156, -0.32246285, -0.05482632,
 -0.24598587, -0.10826535,  0.01030273, -0.11527057,  0.15266289,
  0.27876188,  0.0670862 , -0.42368852, -0.03090081, -0.12863413,
 -0.49077472, -0.49077472],
[-0.12238037,  0.240525  , -0.21163909, -0.49781222,  0.0389006 ,
  0.1370017 , -0.18722761,  0.06993468,  0.60183149, -0.0618589 ,
 -0.13570791, -0.31735138, -0.19052358, -0.00150433, -0.14389794,
  0.1268278 ,  0.1268278 ],
[-0.23689109,  0.13389938,  0.22945899,  0.20077663, -0.34532436,
  0.32427083, -0.03512084, -0.42990896,  0.05417539,  0.3477714 ,
 -0.23161918,  0.02622287, -0.02069034,  0.3137818 , -0.37554148,
 -0.04691321, -0.04691321],
[-0.12479915, -0.02721982,  0.1276831 ,  0.06833512, -0.37252892,
  0.25861926, -0.32718181,  0.26640608,  0.03691677, -0.22834011,
  0.64701738, -0.0198165 ,  0.06349254,  0.28583487,  0.08300497,
  0.08330904,  0.08330904],
[-0.05840368,  0.12897441, -0.29290065, -0.17534937,  0.09372692,
  0.38298757, -0.07896225,  0.05963196, -0.17370222, -0.17815959,
 -0.14706512,  0.7512722 , -0.20216614,  0.07053052,  0.00530531,
  0.04656166,  0.04656166],
[-0.14011258,  0.05190718, -0.58813124, -0.11811105, -0.18059535,
  0.10411881,  0.14934869, -0.12982231, -0.10964869, -0.15682975,
  0.00979282, -0.16912129,  0.54247133,  0.09465949,  0.01737104,
```

```
 -0.28840739, -0.28840739],
[-0.33292338, -0.52221339,  0.11952726, -0.27278898,  0.41344109,
  0.15302752,  0.4148799 , -0.00943139,  0.04554608,  0.05971564,
  0.2076991 ,  0.00320319,  0.0214011 ,  0.29980834, -0.14626825,
  0.01819791,  0.01819791],
[-0.12863223, -0.24275644, -0.01593035, -0.04666133,  0.23594132,
 -0.027139  , -0.52863542, -0.69325173, -0.0270877 , -0.1738857 ,
  0.11580959, -0.01311143,  0.01034517, -0.1568909 ,  0.18777074,
  0.02345661,  0.02345661],
[-0.0817089 , -0.07706724, -0.29523059,  0.05723833, -0.27432227,
 -0.27886876,  0.22831094, -0.18945427,  0.06405354,  0.02132985,
  0.15685794,  0.07960652, -0.25536253,  0.02412897,  0.01206572,
  0.66503095, -0.33496905],
[-0.0817089 , -0.07706724, -0.29523059,  0.05723833, -0.27432227,
 -0.27886876,  0.22831094, -0.18945427,  0.06405354,  0.02132985,
  0.15685794,  0.07960652, -0.25536253,  0.02412897,  0.01206572,
 -0.33496905,  0.66503095]])
```

[6]:
```
S = np.zeros((17, 8))
S[:8,:8] = np.diag(s)
S
```

[6]:
```
array([[1.63573344, 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 1.25616758, 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 1.06374115, 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.92521088, 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.87391752,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.74861232, 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.53664546, 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.38303801],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
```

```
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ]])
```

[7]:
```
V = Vh.T
V
```

[7]:
```
array([[-0.43730648,  0.25767545, -0.08919598,  0.43257977,  0.37292731,
        -0.32072202, -0.20093979,  0.51475337],
       [-0.25633618,  0.34323775,  0.211737  , -0.7307958 , -0.11736408,
        -0.45106878, -0.14231562,  0.00966651],
       [-0.40827629, -0.0683853 ,  0.27164353,  0.12644879, -0.65111911,
         0.38721113, -0.35116127,  0.20408731],
       [-0.40998873,  0.45256386,  0.24208388,  0.27399955,  0.05316151,
         0.10989732,  0.35046607, -0.59639316],
       [-0.19106572,  0.32402696, -0.62314095, -0.3244703 ,  0.1638192 ,
         0.57341844, -0.08474947,  0.04568261],
       [-0.3644374 , -0.52817638, -0.02935093, -0.07477535,  0.3571372 ,
        -0.03518937, -0.49136515, -0.45993183],
       [-0.40926794, -0.42994027,  0.17647573, -0.25623594,  0.18998261,
         0.16518732,  0.62012901,  0.32079642],
       [-0.26730796, -0.19361873, -0.62809785,  0.10591505, -0.47947008,
        -0.41752919,  0.24504406, -0.14513637]])
```

[8]:
```
# Question c)
U2 = U[:,:2]
U2
```

[8]:
```
array([[-0.57095876, -0.38420785],
       [-0.24576484,  0.26368332],
       [-0.30416853,  0.39265773],
       [-0.30974153,  0.3752339 ],
       [-0.06397669,  0.11155058],
       [-0.06397669,  0.11155058],
       [-0.40301335,  0.10572149],
       [-0.06397669,  0.11155058],
       [-0.12238037,  0.240525  ],
       [-0.23689109,  0.13389938],
       [-0.12479915, -0.02721982],
       [-0.05840368,  0.12897441],
       [-0.14011258,  0.05190718],
```

```
       [-0.33292338, -0.52221339],
       [-0.12863223, -0.24275644],
       [-0.0817089 , -0.07706724],
       [-0.0817089 , -0.07706724]])
```

[9]:
```
S2 = np.zeros((17, 8))
S2[:2,:2] = np.diag(s[:2])
S2
```

[9]:
```
array([[1.63573344, 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 1.25616758, 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ]])
```

[10]:
```
V2 = V[:2,:]
V2
```

[10]:
```
array([[-0.43730648,  0.25767545, -0.08919598,  0.43257977,  0.37292731,
        -0.32072202, -0.20093979,  0.51475337],
       [-0.25633618,  0.34323775,  0.211737  , -0.7307958 , -0.11736408,
        -0.45106878, -0.14231562,  0.00966651]])
```

[11]:
```
Vh2 = V2.T
Vh2
```

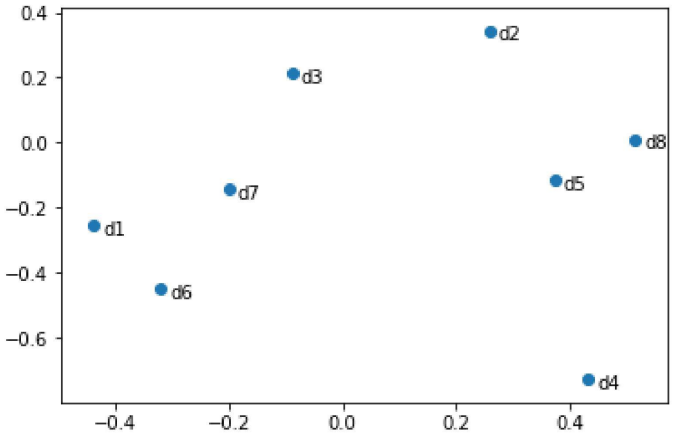
[11]:
```
array([[-0.43730648, -0.25633618],
       [ 0.25767545,  0.34323775],
       [-0.08919598,  0.211737  ],
       [ 0.43257977, -0.7307958 ],
       [ 0.37292731, -0.11736408],
       [-0.32072202, -0.45106878],
       [-0.20093979, -0.14231562],
       [ 0.51475337,  0.00966651]])
```

[12]:
```
A2 = np.dot(np.dot(U, S2), Vh)
A2
```

[12]:
```
array([[ 0.28405465,  0.07374503,  0.41430882,  0.16448272,  0.02205826,
         0.59527481,  0.58973203,  0.34309472],
       [ 0.26114968,  0.21673941,  0.14147813,  0.31472076,  0.18413711,
        -0.02844215,  0.02211877,  0.04332693],
       [ 0.34467372,  0.29683708,  0.16940259,  0.4272096 ,  0.2548869 ,
        -0.0791981 , -0.00843881,  0.03749478],
       [ 0.34302037,  0.2916613 ,  0.17462118,  0.42104166,  0.24953659,
        -0.06431558,  0.00470226,  0.04416932],
       [ 0.08187069,  0.07492189,  0.03314305,  0.1063209 ,  0.06539947,
        -0.03587342, -0.01741651,  0.0008424 ],
       [ 0.08187069,  0.07492189,  0.03314305,  0.1063209 ,  0.06539947,
        -0.03587342, -0.01741651,  0.0008424 ],
       [ 0.32250254,  0.21456587,  0.26006305,  0.33037601,  0.16898685,
         0.17010142,  0.21270085,  0.15050208],
       [ 0.08187069,  0.07492189,  0.03314305,  0.1063209 ,  0.06539947,
        -0.03587342, -0.01741651,  0.0008424 ],
       [ 0.16539472,  0.15501956,  0.06106751,  0.21880974,  0.13614926,
        -0.08662937, -0.04797409, -0.00498975],
       [ 0.21279321,  0.15706049,  0.14670084,  0.23498808,  0.12853754,
         0.0523768 ,  0.08627153,  0.07101266],
       [ 0.08046033,  0.04059178,  0.08568304,  0.06821998,  0.02792446,
         0.09245533,  0.098248  ,  0.06118809],
       [ 0.08352403,  0.08009767,  0.02792446,  0.11248884,  0.07074979,
        -0.05075595, -0.03055758, -0.00583215],
       [ 0.11702639,  0.08112939,  0.08911255,  0.12347305,  0.06491764,
         0.04908498,  0.06576495,  0.04863873],
       [ 0.06911381, -0.08556569,  0.26719652, -0.07360709, -0.10850824,
```

```
  0.54494022,  0.5049121 ,  0.27258042],
[ 0.01343653, -0.05073268,  0.10675821, -0.05174116, -0.05860792,
  0.23774413,  0.21722044,  0.11528638],
[ 0.03350235,  0.00103172,  0.06118809,  0.01098421, -0.00583215,
  0.09984093,  0.09632253,  0.05447088],
[ 0.03350235,  0.00103172,  0.06118809,  0.01098421, -0.00583215,
  0.09984093,  0.09632253,  0.05447088]])
```

[13]:
```
# plot document in 2D
plt.scatter(Vh2[:,0], Vh2[:,1])
for i, xy in enumerate(zip(Vh2[:,0], Vh2[:,1])):
    plt.annotate("d%s" % str(i+1), xy=xy, xytext=(5, -5), textcoords='offset␣
 ↪points')
```



[14]:
```
# Question d)
q1 = np.array([1,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0])
q1 = q1 / np.linalg.norm(q1)
q2 = np.array([1,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0])
q2 = q2 / np.linalg.norm(q2)
```

[15]:
```
S2_inv = inv(S2[:2,:2])
S2_inv
```

[15]:
```
array([[0.61134655, 0.        ],
       [0.        , 0.79607213]])
```

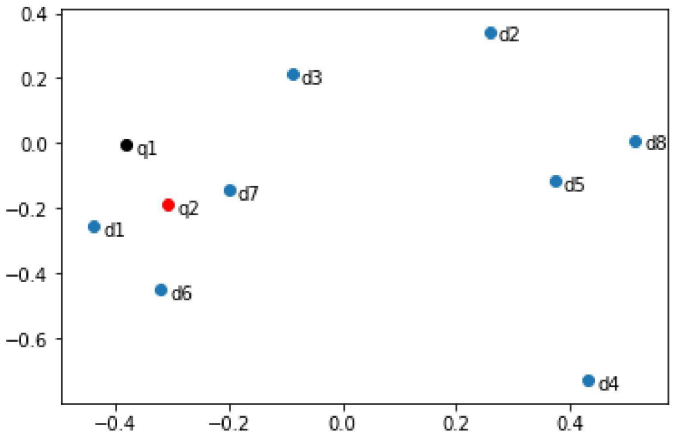
[16]:
```
q1_2D = np.dot(np.dot(q1, U2), S2_inv)
q1_2D
```

[16]:
```
array([-0.38071554, -0.00505151])
```

[17]:
```
q2_2D = np.dot(np.dot(q2, U2), S2_inv)
q2_2D
```

[17]:
```
array([-0.30738711, -0.18705471])
```

[18]:
```
# query q1 & q2 in 2D
plt.scatter(Vh2[:,0], Vh2[:,1])
for i, xy in enumerate(zip(Vh2[:,0], Vh2[:,1])):
    plt.annotate("d%s" % str(i+1), xy=xy, xytext=(5, -5), textcoords='offset␣
 ↪points')
plt.scatter(q1_2D[0], q1_2D[1], c='black')
plt.annotate("q1", xy=(q1_2D[0], q1_2D[1]), xytext=(5, -5), textcoords='offset␣
 ↪points')
plt.scatter(q2_2D[0], q2_2D[1], c='red')
plt.annotate("q2", xy=(q2_2D[0], q2_2D[1]), xytext=(5, -5), textcoords='offset␣
 ↪points')
plt.show()
```



[19]:
```
# Question e)
q1_A2_inner = np.inner(A2.T, q1)
```

```
q1_A2_top3 = np.argsort(-q1_A2_inner) + 1
print("top 3 documents for q1 in the reduced vector space: ", q1_A2_top3[0:3])
```

```
top 3 documents for q1 in the reduced vector space:  [1 7 3]
```

[20]:
```
q1_Anorm_inner = np.inner(A_norm.T, q1)
q1_Anorm_top3 = np.argsort(-q1_Anorm_inner) + 1
print("top 3 documents for q1 in the original vector space: ", q1_Anorm_top3[0:
 ↪3])
```

```
top 3 documents for q1 in the original vector space:  [1 6 3]
```

[21]:
```
q2_A2_inner = np.inner(A2.T, q2)
q2_A2_top3 = np.argsort(-q2_A2_inner) + 1
print("top 3 documents for q2 in the reduced vector space: ", q2_A2_top3[0:3])
```

```
top 3 documents for q2 in the reduced vector space:  [7 6 3]
```

[22]:
```
# Question e)
q2_Anorm_inner = np.inner(A_norm.T, q2)
q2_Anorm_top3 = np.argsort(-q2_Anorm_inner) + 1
print("top 3 documents for q2 in the original vector space: ", q2_Anorm_top3[0:
 ↪3])
```

```
top 3 documents for q2 in the original vector space:  [8 6 1]
```

f)

‘success’ in D3 and ‘successful’ in D4 are synonymy, so put them in 1 term T10 ‘success’, this term occurs in D3 and D4.

‘windows’ has 2 meanings here: opening window in wall, a Microsoft system. In D1-D3, ‘windows’ means system, in D6-D8, ‘windows’ means window in the wall. The meaning of ‘windows’ in two queries is different, q1 has the same meaning with D1-D3, q2 has the same meaning with D6-D8. For q1 with LSI in this example, the top 3 documents in reduced or original vector space has similar result, D1 and D3 is correct, D6 and D7 are incorrect. For q2 with LSI, in reduced vector space it fails to get D8 for top 3 documents, but it gets D6 and D7, so Rank-2 approximation might reduce accuracy but increase the speed and reduce memory.

‘gates’ also have two different meanings, one is surname of Bill Gates, the other is the barrier.

Q2:

[1]:
```
import numpy as np
from sklearn.preprocessing import normalize
```

[2]:
```
def cal_Q(A):
    A_norm = normalize(A, axis=0)
    A_T = A_norm.T
    N = 8
    outer_sum = 0
    for i in range(8):
        inner_sum = 0
        for k in range(8):
            if (i != k):
                inner_sum += np.inner(A_T[i], A_T[k])
        outer_sum += inner_sum
    Q = 1/(N*(N-1))*outer_sum
    return Q
```

[3]:
```
A = [[]]*17
Q = [0]*17
dv = [0]*17
```

[4]:
```
# Q_origin is the average similarity value between document pairs in 8␣
↪documents with all 17 terms in Q1
A_origin = np.array([[1,0,1,0,0,1,1,1],␣
↪[1,0,0,1,0,0,0,0],[1,0,0,1,1,0,0,0],[1,1,0,1,0,0,0,0],
                ␣
↪[0,1,0,0,0,0,0,0],[0,1,0,0,0,0,0,0],[0,1,1,1,0,0,1,0],[0,1,0,0,0,0,0,0],
                ␣
↪[0,1,0,0,1,0,0,0],[0,0,1,1,0,0,0,0],[0,0,1,0,0,0,0,0],[0,0,0,0,1,0,0,0],
                ␣
↪[0,0,0,0,1,0,0,1],[0,0,0,0,0,1,2,0],[0,0,0,0,0,1,0,0],[0,0,0,0,0,0,0,1],[0,0,0,0,0,0,0,1]])
Q_origin = cal_Q(A_origin)
Q_origin
```

[4]:
```
0.2252868290095076
```

```
[5]: # Q[j](j = 0,...,16) are the average similarity values after
     # removing term j to the documents of a collection
     # term discrimination values as dv[j] = Q[j] - Q_origin
     for i in range(17):
         A[i] = np.delete(A_origin, i, 0)
         Q[i] = cal_Q(A[i])
         dv[i] = Q[i] - Q_origin
     dv
```

```
[5]: [-0.06793478754577989,
      0.009792828347518184,
      -0.006936524682353784,
      -0.003259704827657417,
      0.003900253751872196,
      0.003900253751872196,
      -0.02251759493643843,
      0.003900253751872196,
      -8.776183966965889e-05,
      0.008158108560873845,
      0.010211678479045166,
      0.005125731298228325,
      0.0003011401731182195,
      0.030032393441576183,
      0.012626906806902621,
      0.006866489917376539,
      0.006866489917376539]
```

```
[6]: indexes = np.argsort(-np.array(dv)) + 1
     select_term = indexes[:2]
     select_term
```

```
[6]: array([14, 15])
```

According to the result, choosing ‘wooden’ and ‘frames’ as index terms is better.

Q3:

Strategy 1: click > skip above
page 2 > page 1, page 8 > page 3, page 8 > page 4, page 8 > page 5, page 8 > page 6, page 8 > page 7
Strategy 2: last click > skip above
page 8 > page 1, page 8 > page 3, page 8 > page 4, page 8 > page 5, page 8 > page 6, page 8 > page 7
Strategy 3: click > earlier click
page 8 > page 2
Strategy 4: last click > skip previous
page 2 > page 1, page 8 > page 7
Strategy 5: click > no-click next
page 2 > page 3, page 8 > page 9

Above all:
page 8 > page 2 > (page 1, page 3)
page 8 > (page 4, page 5, page 6, page 7, page 9)

Q4:

1. Pages that cited from many places have a high score in Page Rank, when random selecting websites people have a high possibility to select such high score pages.
2. Pages that have a citation from important websites like Wikipedia have a high

score in Page Rank, on the same way, they are worth looking at.

3. This pattern is similar to recommended system, people random surfer is actually based on Page Rank algorithm. People have small possibility to pick low Page Rank score pages even if they are also worth looking at.