In [3]:

```
import numpy as np
```

In [109]:

```
X = np.array([1,1,0,0,0,0,0,
              0,0,1,1,1,0,0,
              0,1,0,0,0,1,0,
              0,0,1,0,0,0,1,
              0,0,0,1,1,0,1,
              1,0,1,0,1,1,1])
```

In [110]:

```
X = X.T.reshape(7,-1)
X
```

Out[110]:

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

In [111]:

```
X[0].reshape(1,-1).shape, X[0].reshape(1,-1).T.shape
```

Out[111]:

```
((1, 6), (6, 1))
```

In [112]:

```
Xi = X[[0]]
Xi.dot(Xi.T), Xi
```

Out[112]:

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

In [113]:

```
np.linalg.norm(Xi), np.linalg.norm(Xi.T)
```

Out[113]:

```
(1.4142135623730951, 1.4142135623730951)
```

In [114]:

```
X.T.dot(X) #내적(단어별)
np.linalg.norm(X, axis=1) #길이
np.linalg.norm(X.T, axis=0)
```

Out[114]:

```
array([1.41421356, 1.73205081, 1.        , 1.41421356, 1.        ,
       2.        , 2.        ])
```

In [115]:

```
len1 = np.linalg.norm(X, axis=1).reshape(1,-1) #큰 매트릭스로 복원하기
len2 = np.linalg.norm(X.T, axis=0).reshape(-1,1)
print(X.dot(X.T) / (len1 * len2))
```

```
[[1.         0.         0.         0.5        0.         0.35355339
  0.35355339]
 [0.         1.         0.57735027 0.40824829 0.57735027 0.57735027
  0.8660254 ]
 [0.         0.57735027 1.         0.         1.         0.
  0.5       ]
 [0.5        0.40824829 0.         1.         0.         0.70710678
  0.70710678]
 [0.         0.57735027 1.         0.         1.         0.
  0.5       ]
 [0.35355339 0.57735027 0.         0.70710678 0.         1.
  0.75      ]
 [0.35355339 0.8660254  0.5        0.70710678 0.5        0.75
  1.        ]]
```

In [116]:

```
X = np.array([1,1,0,0,0,0,0,
              0,0,1,1,1,0,0,
              0,1,0,0,0,1,0,
              0,0,1,0,0,0,1,
              0,0,0,1,1,0,1,
              1,0,1,0,1,1,1])

X = X.reshape(7,-1)
Xi = X[[0]]
X.T.dot(X)
np.linalg.norm(X.T, axis=1)
np.linalg.norm(X, axis=0)

len1 = np.linalg.norm(X.T, axis=1).reshape(1,-1) #큰 매트릭스로 복원하기
len2 = np.linalg.norm(X, axis=0).reshape(-1,1)
print(X.T.dot(X) / (len1 * len2))
```

```
[[1.         0.5        0.         0.         0.         0.        ]
 [0.5        1.         0.5        0.25       0.57735027 0.75      ]
 [0.         0.5        1.         0.         0.57735027 0.5       ]
 [0.         0.25       0.         1.         0.57735027 0.5       ]
 [0.         0.57735027 0.57735027 0.57735027 1.         0.8660254 ]
 [0.         0.75       0.5        0.5        0.8660254  1.        ]]
```

In [117]:

```
U, Sigma, Vt = np.linalg.svd(X, full_matrices=False)
U.shape, Sigma.shape, Vt.shape
```

Out[117]:

```
((7, 6), (6,), (6, 6))
```

In [118]:

```
print(Sigma)
print(np.round(U.dot(U.T)))
print(U[[0]].dot(U[[0]].T))
print(np.sum(Sigma[:3]) / np.sum(Sigma))
```

```
[3.26630358 1.84095977 1.32307523 0.81963238 0.63492329 0.3415775 ]
[[ 1.  0. -0. -0. -0.  0. -0.]
 [ 0.  1. -0.  0. -0. -0. -0.]
 [-0. -0.  1. -0.  0.  0. -0.]
 [-0.  0. -0.  1.  0.  0. -0.]
 [-0. -0.  0.  0.  0.  0. -0.]
 [ 0. -0.  0.  0.  0.  1. -0.]
 [-0. -0. -0. -0. -0. -0.  1.]]
[[1.]]
0.7816642151436408
```

In [119]:

```
U.dot(np.diag(Sigma))
print(np.round(U.dot(np.diag(Sigma)), 4))
print("₩n")
print(np.round(np.diag(Sigma).dot(Vt), 3))
```

```
[[ 0.5482  0.7616 -1.004   0.2677 -0.1915  0.0564]
 [ 1.456  -0.8064  0.2653  0.0129 -0.3616  0.169 ]
 [ 0.3932 -0.7745 -0.4015  0.0791  0.2753  0.0472]
 [ 1.0788  0.6111 -0.1358 -0.641   0.1452  0.1114]
 [ 0.3932 -0.7745 -0.4015  0.0791  0.2753  0.0472]
 [ 1.7209  0.7164  0.5514  0.4033  0.2386  0.0403]
 [ 1.9528 -0.2695 -0.1651 -0.1179 -0.078  -0.2578]]


[[ 0.168  1.623  0.527  1.284  1.571  1.901]
 [ 0.414  0.988  0.389 -1.426 -0.195  0.137]
 [-0.759 -0.57   0.417 -0.531  0.492  0.39 ]
 [ 0.327 -0.107  0.492  0.065  0.364 -0.418]
 [-0.302  0.18   0.376  0.175 -0.317 -0.088]
 [ 0.165 -0.146  0.118  0.016 -0.142  0.184]]
```

In [120]:

```
np.round(U.dot(np.diag(Sigma)).dot(Vt), 3)
print(np.round(U[:,:3].dot(np.diag(Sigma[:3])).dot(Vt[:3])))
print("₩n")
print(X)
```

```
[[ 1.  1. -0.  0. -0.  0.]
 [-0.  0.  0.  1.  1.  1.]
 [ 0. -0. -0.  1.  0.  0.]
 [ 0.  1.  0.  0.  0.  1.]
 [ 0. -0. -0.  1.  0.  0.]
 [-0.  1.  1. -0.  1.  1.]
 [ 0.  1.  0.  1.  1.  1.]]


[[1 1 0 0 0 0]
 [0 0 0 1 1 1]
 [0 0 0 1 0 0]
 [0 1 0 0 0 1]
 [0 0 0 1 0 0]
 [0 1 1 0 1 1]
 [0 1 0 1 1 1]]
```

In [121]:

```
USigma = U.dot(np.diag(Sigma))
USigma = U[:,:2].dot(np.diag(Sigma[:2]))
print(USigma.shape)  #바뀐 차원에서 어떤 값을 가지는가
print(USigma)
```

```
(7, 2)
[[ 0.5482415   0.76155898]
 [ 1.45599552 -0.80637134]
 [ 0.39323422 -0.77450464]
 [ 1.0787931   0.61107101]
 [ 0.39323422 -0.77450464]
 [ 1.72092006  0.71635409]
 [ 1.95284937 -0.2695185 ]]
```

In [122]:

```
#유사도
USigma.shape #코사인..
len1 = np.linalg.norm(USigma, axis=1).reshape(-1,1) #큰 매트릭스로 복원하기
len2 = np.linalg.norm(USigma.T, axis=0).reshape(1,-1)
print(np.round(USigma.dot(USigma.T) / (len1 * len2), 3)) # - 는 덜 유사? 반대? 애매함

print("₩n")

len1 = np.linalg.norm(X, axis=1).reshape(-1,1) #큰 매트릭스로 복원하기
len2 = np.linalg.norm(X.T, axis=0).reshape(1,-1)
print(np.round(X.dot(X.T) / (len1 * len2), 3))
```

```
[[ 1.     0.118 -0.459  0.908 -0.459  0.851  0.468]
 [ 0.118  1.     0.828  0.522  0.828  0.621  0.933]
 [-0.459  0.828  1.    -0.046  1.     0.075  0.57 ]
 [ 0.908  0.522 -0.046  1.    -0.046  0.993  0.795]
 [-0.459  0.828  1.    -0.046  1.     0.075  0.57 ]
 [ 0.851  0.621  0.075  0.993  0.075  1.     0.862]
 [ 0.468  0.933  0.57   0.795  0.57   0.862  1.   ]]


[[1.     0.     0.     0.5    0.     0.354 0.354]
 [0.     1.     0.577  0.408  0.577  0.577 0.866]
 [0.     0.577  1.     0.     1.     0.    0.5  ]
 [0.5    0.408  0.     1.     0.     0.707 0.707]
 [0.     0.577  1.     0.     1.     0.    0.5  ]
 [0.354  0.577  0.     0.707  0.     1.    0.75 ]
 [0.354  0.866  0.5    0.707  0.5    0.75  1.   ]]
```

In [123]:

```
USigma.shape #코사인..
len1 = np.linalg.norm(USigma, axis=1).reshape(-1,1) #큰 매트릭스로 복원하기
len2 = np.linalg.norm(USigma.T, axis=0).reshape(1,-1)
print(USigma.dot(USigma.T) / (len1 * len2)) # - 는 덜 유사? 반대? 애매함

print("₩n")

len1 = np.linalg.norm(X, axis=1).reshape(-1,1) #큰 매트릭스로 복원하기
len2 = np.linalg.norm(X.T, axis=0).reshape(1,-1)
print(X.dot(X.T) / (len1 * len2))
```

```
[[ 1.          0.11790061 -0.45914807  0.90835353 -0.45914807  0.8512693
   0.46780574]
 [ 0.11790061  1.          0.82802996  0.52238177  0.82802996  0.62143445
   0.93282153]
 [-0.45914807  0.82802996  1.         -0.04555394  1.          0.07528951
   0.57036806]
 [ 0.90835353  0.52238177 -0.04555394  1.         -0.04555394  0.99269681
   0.79455402]
 [-0.45914807  0.82802996  1.         -0.04555394  1.          0.07528951
   0.57036806]
 [ 0.8512693   0.62143445  0.07528951  0.99269681  0.07528951  1.
   0.86200063]
 [ 0.46780574  0.93282153  0.57036806  0.79455402  0.57036806  0.86200063
   1.        ]]


[[1.         0.         0.         0.5        0.         0.35355339
  0.35355339]
 [0.         1.         0.57735027 0.40824829 0.57735027 0.57735027
  0.8660254 ]
 [0.         0.57735027 1.         0.         1.         0.
  0.5       ]
 [0.5        0.40824829 0.         1.         0.         0.70710678
  0.70710678]
 [0.         0.57735027 1.         0.         1.         0.
  0.5       ]
 [0.35355339 0.57735027 0.         0.70710678 0.         1.
  0.75      ]
 [0.35355339 0.8660254  0.5        0.70710678 0.5        0.75
  1.        ]]
```

In [124]:

```
len1 = np.linalg.norm(X, axis=0).reshape(1,-1) #큰 매트릭스로 복원하기
len2 = np.linalg.norm(X.T, axis=1).reshape(-1,1)
np.round(X.T.dot(X) / (len1 * len2))
```

Out[124]:

```
array([[1., 0., 0., 0., 0., 0.],
       [0., 1., 0., 0., 1., 1.],
       [0., 0., 1., 0., 1., 0.],
       [0., 0., 0., 1., 1., 0.],
       [0., 1., 1., 1., 1., 1.],
       [0., 1., 0., 0., 1., 1.]])
```

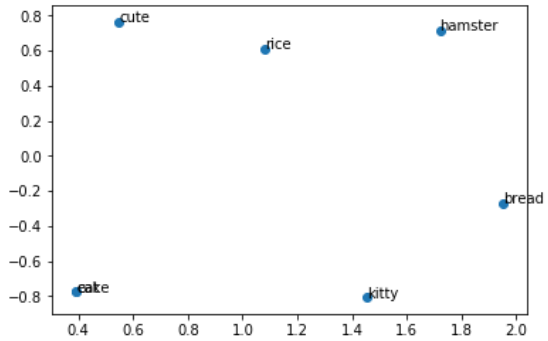
In [125]:

```
import matplotlib.pyplot as plt
plt.scatter(USigma[:,0], USigma[:,1])

for i, txt in enumerate(["cute", "kitty","eat", "rice", "cake", "hamster", "bread"]):
    plt.text(USigma[i,0], USigma[i,1], txt)
plt.show
#워드 임베딩 <- 다차원 공간의 단어를 2차원에 맵핑
```

Out[125]:

```
<function matplotlib.pyplot.show(*args, **kw)>
```



In [126]:

```
len1 = np.linalg.norm(X, axis=0).reshape(1,-1) #큰 매트릭스로 복원하기
len2 = np.linalg.norm(X.T, axis=1).reshape(-1,1)
print(np.round(X.T.dot(X) / (len1 * len2), 7))

print("Wn")

SVt = np.diag(Sigma[:7]).dot(Vt[:7]) #코사인..
len1 = np.linalg.norm(SVt, axis=0).reshape(-1,1) #큰 매트릭스로 복원하기
len2 = np.linalg.norm(SVt.T, axis=1).reshape(1,-1)
print(np.round(SVt.T.dot(SVt) / (len1 * len2), 7)) # - 는 덜 유사? 반대? 애매함
```

```
[[1.        0.5       0.        0.        0.        0.        ]
 [0.5       1.        0.5       0.25      0.5773503 0.75      ]
 [0.        0.5       1.        0.        0.5773503 0.5       ]
 [0.        0.25      0.        1.        0.5773503 0.5       ]
 [0.        0.5773503 0.5773503 0.5773503 1.        0.8660254]
 [0.        0.75      0.5       0.5       0.8660254 1.        ]]


[[ 1.         0.5       -0.        0.        0.        0.       ]
 [ 0.5        1.         0.5       0.25      0.5773503 0.75     ]
 [-0.         0.5        1.       -0.        0.5773503 0.5      ]
 [ 0.         0.25      -0.        1.        0.5773503 0.5      ]
 [ 0.         0.5773503  0.5773503 0.5773503 1.        0.8660254]
 [ 0.         0.75       0.5       0.5       0.8660254 1.       ]]
```

In [127]:

```
SVt
```

Out[127]:

```
array([[ 0.16784769,  1.62287549,  0.52687082,  1.28442236,  1.57051077,
         1.90079027],
       [ 0.41367497,  0.98832447,  0.38911991, -1.42583187, -0.19529799,
         0.13663268],
       [-0.75880269, -0.56950271,  0.41675346, -0.53126826,  0.49246576,
         0.38982211],
       [ 0.32665676, -0.10720965,  0.49206242,  0.06483465,  0.36387857,
        -0.41816517],
       [-0.3016261 ,  0.1800323 ,  0.37586266,  0.1747953 , -0.31653798,
        -0.08783637],
       [ 0.16503247, -0.14577728,  0.11794586,  0.01611808, -0.14222579,
         0.18381856]])
```

In [ ]: