In [20]:

```
import numpy as np
```

In [21]:

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

In [22]:

```
X = X.reshape(5,-1)
X
```

Out[22]:

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

In [23]:

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

Out[23]:

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

In [24]:

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

Out[24]:

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

In [25]:

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

Out[25]:

```
(1.4142135623730951, 1.4142135623730951)
```

In [26]:

```
#X.dot(X.T)

'''
6 x 6 행렬  // 처음에 5 * 6 => min(M,N) = K
'''
X.T.dot(X) #내적(단어별)
np.linalg.norm(X, axis=1) #길이
np.linalg.norm(X.T, axis=0) #길이 (위와 동일)
```

Out[26]:

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

In [27]:

```
np.linalg.norm(X, axis=1) * np.linalg.norm(X.T, axis=0)
```

Out[27]:

```
array([2., 1., 2., 3., 2.])
```

In [28]:

```
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.         0.5        0.40824829 0.        ]
 [0.         1.         0.70710678 0.         0.        ]
 [0.5        0.70710678 1.         0.40824829 0.        ]
 [0.40824829 0.         0.40824829 1.         0.40824829]
 [0.         0.         0.         0.40824829 1.        ]]
```

In [29]:

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

X = X.reshape(5,-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.         0.40824829 0.57735027 0.40824829 0.57735027 0.        ]
 [0.40824829 1.         0.         0.         0.         0.        ]
 [0.57735027 0.         1.         0.         0.         0.        ]
 [0.40824829 0.         0.         1.         0.70710678 0.70710678]
 [0.57735027 0.         0.         0.70710678 1.         0.        ]
 [0.         0.         0.         0.70710678 0.         1.        ]]
```

In [30]:

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

'''
Sigma = singular value <- svd
'''
U.shape, Sigma.shape, Vt.shape
```

Out[30]:

```
((5, 5), (5,), (5, 6))
```

In [31]:

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

```
[2.16250096 1.59438237 1.27529025 1.         0.39391525]
[[ 1. -0.  0. -0. -0.]
 [-0.  1.  0.  0. -0.]
 [ 0.  0.  1.  0.  0.]
 [-0.  0.  0.  1.  0.]
 [-0. -0.  0.  0.  1.]]
[[1.]]
0.7830849702365396
```

In [32]:

```
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))
```

```
[[ 0.9523 -0.4722 -0.7263  0.5774 -0.0971]
 [ 0.2797 -0.5285  0.7486 -0.     -0.2865]
 [ 1.0283 -0.8149  0.4689 -0.      0.242 ]
 [ 1.5203  0.5589 -0.1975 -0.5774 -0.0629]
 [ 0.568   1.0312  0.5287  0.5774  0.0341]]


[[ 1.619  0.605  0.44   0.966  0.703  0.263]
 [-0.457 -0.843 -0.296  0.997  0.351  0.647]
 [-0.357  0.955 -0.569  0.26  -0.155  0.415]
 [-0.    -0.    0.577  0.    -0.577  0.577]
 [ 0.208 -0.113 -0.246 -0.073 -0.16   0.087]]
```

In [33]:

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


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

In [34]:

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

```
(5, 2)
[[ 0.95225185 -0.47221518]
 [ 0.2797116  -0.52845914]
 [ 1.02833465 -0.81491313]
 [ 1.52028211  0.55894647]
 [ 0.56803026  1.03116165]]
```

In [35]:

```
#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.     0.812  0.978  0.688  0.043]
 [ 0.812  1.     0.916  0.134 -0.548]
 [ 0.978  0.916  1.     0.521 -0.166]
 [ 0.688  0.134  0.521  1.     0.755]
 [ 0.043 -0.548 -0.166  0.755  1.   ]]


[[1.    0.    0.5   0.408 0.   ]
 [0.    1.    0.707 0.    0.   ]
 [0.5   0.707 1.    0.408 0.   ]
 [0.408 0.    0.408 1.    0.408]
 [0.    0.    0.    0.408 1.   ]]
```

In [36]:

```
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("Wn")

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.          0.81176374  0.97807901  0.68755734  0.0431365 ]
 [ 0.81176374  1.          0.91557488  0.13408432 -0.54842575]
 [ 0.97807901  0.91557488  1.          0.52128036 -0.16584937]
 [ 0.68755734  0.13408432  0.52128036  1.          0.75511302]
 [ 0.0431365  -0.54842575 -0.16584937  0.75511302  1.        ]]


[[1.         0.         0.5        0.40824829 0.        ]
 [0.         1.         0.70710678 0.         0.        ]
 [0.5        0.70710678 1.         0.40824829 0.        ]
 [0.40824829 0.         0.40824829 1.         0.40824829]
 [0.         0.         0.         0.40824829 1.        ]]
```

In [37]:

```
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[37]:

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

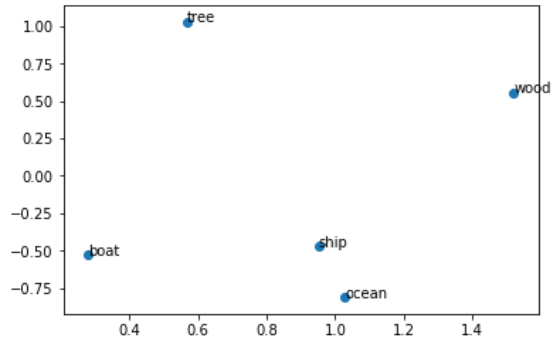
In [38]:

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

for i, txt in enumerate(["ship", "boat", "ocean", "wood", "tree"]):
    plt.text(USigma[i,0], USigma[i,1], txt)
plt.show
#워드 임베딩 < - 다차원 공간의 단어를 2차원에 맵핑
```

Out[38]:

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



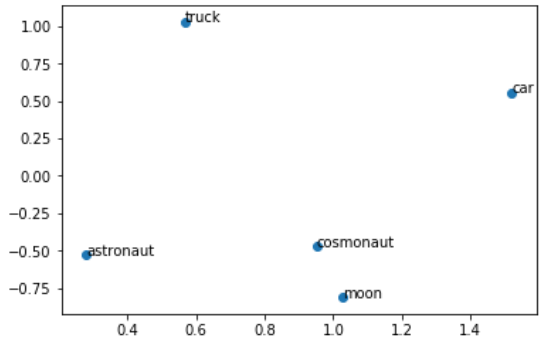
In [39]:

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

for i, txt in enumerate(["cosmonaut", "astronaut", "moon", "car", "truck"]):
    plt.text(USigma[i,0], USigma[i,1], txt)
plt.show
#워드 임베딩 < - 다차원 공간의 단어를 2차원에 맵핑
```

Out[39]:

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



In [40]:

```
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), 3))

print("₩n")

SVt = np.diag(Sigma[:2]).dot(Vt[:2]) #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), 3)) # - 는 덜 유사? 반대? 애매함
```

```
[[1.    0.408 0.577 0.408 0.577 0.   ]
 [0.408 1.    0.    0.    0.    0.   ]
 [0.577 0.    1.    0.    0.    0.   ]
 [0.408 0.    0.    1.    0.707 0.707]
 [0.577 0.    0.    0.707 1.    0.   ]
 [0.    0.    0.    0.707 0.    1.   ]]


[[ 1.     0.782  0.95   0.474  0.74   0.111]
 [ 0.782  1.     0.937 -0.178  0.159 -0.533]
 [ 0.95   0.937  1.     0.176  0.494 -0.205]
 [ 0.474 -0.178  0.176  1.     0.943  0.927]
 [ 0.74   0.159  0.494  0.943  1.     0.75 ]
 [ 0.111 -0.533 -0.205  0.927  0.75   1.   ]]
```