In [1]:

```
import numpy as np
```

In [6]:

```
words = ['I', 'like', 'enjoy', 'deep', 'learning', 'NLP', 'flying', '.']
```

In [7]:

```
X = np.array([[0,2,1,0,0,0,0,0],
              [2,0,0,1,0,1,0,0],
              [1,0,0,0,0,0,1,0],
              [0,1,0,0,1,0,0,0],
              [0,0,0,1,0,0,0,1],
              [0,1,0,0,0,0,0,1],
              [0,0,1,0,0,0,0,1],
              [0,0,0,0,1,1,1,0]])

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

In [8]:

```
_Sigma = np.diag(Sigma[:2])
_Sigma
```

Out[8]:

```
array([[2.75726275, 0.        ],
       [0.        , 2.678248  ]])
```

In [10]:

```
len(words), U.shape, Sigma.shape, V.shape, _Sigma.shape
```

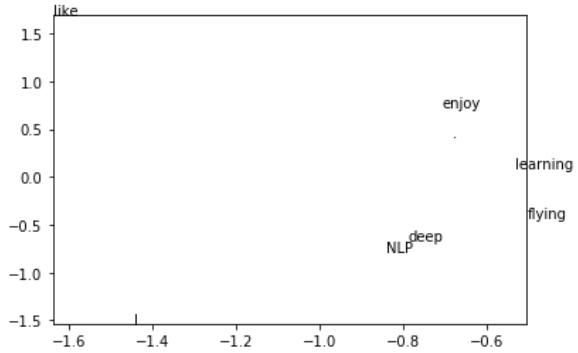
Out[10]:

```
(8, (8, 8), (8,), (8, 8), (2, 2))
```

In [12]:

```
import matplotlib.pyplot as plt

US = U[:,:2].dot(_Sigma)
SV = _Sigma.dot(V[:2])

for t, _repr in zip(words, US):
    print(t, _repr)
    plt.text(_repr[0], _repr[1], t)
plt.xlim(US[:,0].min(), US[:,0].max())
plt.ylim(US[:,1].min(), US[:,1].max())
plt.show()
```

```
I [-1.44515015 -1.53425886]
like [-1.63902195  1.68761941]
enjoy [-0.70661477  0.73388691]
deep [-0.78757738 -0.66397017]
learning [-0.53253583  0.09065737]
NLP [-0.8413365  -0.78737543]
flying [-0.50317243 -0.4312723 ]
. [-0.68076383  0.42116725]
```



In [13]:

```
query = US[words.index("deep")]
US.shape, query.shape
```

Out[13]:

```
((8, 2), (2,))
```

In [14]:

```
result = np.linalg.norm(US - np.repeat(query, len(words)).reshape(2, -1).T, axis=1)

for dist,w in zip(result, words):
    print(w, dist)
```

```
I 1.0907815298565604
like 2.5009860841516702
enjoy 1.400199757357459
deep 0.0
learning 0.7965606768052836
NLP 0.13460646241428836
flying 0.367470376518128
. 1.0903817475676898
```

In [8]:

```
'''
distance vs angle
euc, cos
'''
```

In [15]:

```
US.shape, query.shape
```

Out[15]:

```
((8, 2), (2,))
```

In [22]:

```
queryMat = np.repeat(query, len(words), axis=0).reshape(-1, 8).T

for t, dist in zip(words, np.linalg.norm(US-queryMat, axis=1)):
    if t != 'deep':
        print(t,dist)

sorted({t:dist for t,dist in zip(words, np.linalg.norm(US-queryMat, axis=1))}.W
      items(), key=lambda x:x[1])[0]
```

```
I 1.0907815298565604
like 2.5009860841516702
enjoy 1.400199757357459
learning 0.7965606768052836
NLP 0.13460646241428836
flying 0.367470376518128
. 1.0903817475676898
```

Out[22]:

```
('deep', 0.0)
```

In [17]:

```
documents = [
    'king is a strong man',
    'queen is a wise woman',
    'boy is a young man',
    'girl is a young woman',
    'prince is a young king',
    'princess is a young queen',
    'man is strong',
    'woman is pretty',
    'prince is a boy will be king',
    'princess is a girl will be queen'
]

stop_words = ["is", "a", "will", "be"]
```

In [18]:

```
_documents = list()
Vocabulary = list()

for document in documents:
    _termList = list()
    for term in document.lower().split():
        if term not in stop_words:
            _termList.append(term)
            Vocabulary.append(term)
    _documents.append(_termList)

Vocabulary = list(set(Vocabulary))
```

In [19]:

```
len(Vocabulary), (12,12), _documents
```

Out[19]:

```
(12,
 (12, 12),
 [['king', 'strong', 'man'],
  ['queen', 'wise', 'woman'],
  ['boy', 'young', 'man'],
  ['girl', 'young', 'woman'],
  ['prince', 'young', 'king'],
  ['princess', 'young', 'queen'],
  ['man', 'strong'],
  ['woman', 'pretty'],
  ['prince', 'boy', 'king'],
  ['princess', 'girl', 'queen']])
```

In [20]:

```
V = len(Vocabulary)
X = np.zeros((V, V))
```

In [21]:

```
Vocabulary[0], X[0], Vocabulary.index('strong')
```

Out[21]:

```
('man', array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.]), 10)
```

In [23]:

```
'''
V = ['king', 'strong', 'man', 'queen', 'wise', 'woman']

['king', 'strong', 'man'],
  ['queen', 'wise', 'woman']

king 행, strong 열 1
strong 행, king 열 1, man 열 1

t번째 -> 행
t-1번째 열, t+1번째 열
'''
```

Out[23]:

```
"WnV = ['king', 'strong', 'man', 'queen', 'wise', 'woman']WnWn['king', 'strong',
'man'],Wn  ['queen', 'wise', 'woman']WnWnking 행, strong 열 1Wnstrong 행, king 열
1, man 열 1WnWnt번째 -> 행Wnt-1번째 열, t+1번째 열Wn"
```

In [25]:

```
WINDOW = 1
for document in _documents:
    for v in range(len(document) - WINDOW):
        i = Vocabulary.index(document[v])
        j = Vocabulary.index(document[v+WINDOW])
        X[i][j] += 1
        X[j][i] += 1
```

In [26]:

```
print(X.shape)
print(Vocabulary, X)
Vocabulary.index("strong")
```

```
(12, 12)
['man', 'king', 'boy', 'princess', 'girl', 'woman', 'young', 'wise', 'queen', 'pri
nce', 'strong', 'pretty'] [[0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 2. 0.]
 [0. 0. 1. 0. 0. 0. 1. 0. 0. 0. 1. 0.]
 [0. 1. 0. 0. 0. 0. 1. 0. 0. 1. 0. 0.]
 [0. 0. 0. 0. 1. 0. 1. 0. 0. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 1. 0. 1. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 1. 1. 0. 0. 0. 1.]
 [1. 1. 1. 1. 1. 1. 0. 0. 1. 1. 0. 0.]
 [0. 0. 0. 0. 0. 1. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 0. 1. 0. 1. 1. 0. 0. 0. 0.]
 [0. 0. 1. 0. 0. 0. 1. 0. 0. 0. 0. 0.]
 [2. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0.]]
```

Out[26]:

```
10
```

In [27]:

```
X
```

Out[27]:

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

In [28]:

```
import pandas as pd

pd.DataFrame(X, index=Vocabulary, columns=Vocabulary, dtype=int)
```

Out[28]:

	man	king	boy	princess	girl	woman	young	wise	queen	prince	strong	pret
man	0	0	0	0	0	0	1	0	0	0	2	
king	0	0	1	0	0	0	1	0	0	0	1	
boy	0	1	0	0	0	0	1	0	0	1	0	
princess	0	0	0	0	1	0	1	0	0	0	0	
girl	0	0	0	1	0	0	1	0	1	0	0	
woman	0	0	0	0	0	0	1	1	0	0	0	
young	1	1	1	1	1	1	0	0	1	1	0	
wise	0	0	0	0	0	1	0	0	1	0	0	
queen	0	0	0	0	1	0	1	1	0	0	0	
prince	0	0	1	0	0	0	1	0	0	0	0	
strong	2	1	0	0	0	0	0	0	0	0	0	
pretty	0	0	0	0	0	1	0	0	0	0	0	

In [30]:

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

In [31]:

```
_Sigma = np.diag(Sigma[:2])
_Sigma
```

Out[31]:

```
array([[3.69381462, 0.        ],
       [0.        , 3.01139938]])
```

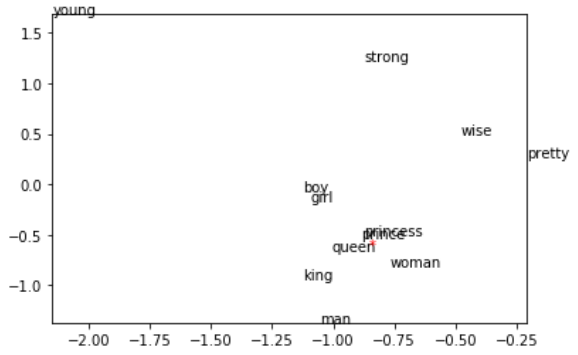
In [32]:

```
US = U[:,:2].dot(_Sigma)
SV = _Sigma.dot(V[:2])
```

In [33]:

```
prince = US[Vocabulary.index('prince')]
boy = US[Vocabulary.index('boy')]
girl = US[Vocabulary.index('girl')]

query = prince - boy + girl

result = np.linalg.norm(US - np.repeat(query, len(Vocabulary)).reshape(2,-1).T, axis=1)

for t,point in zip(Vocabulary, US):
    plt.text(point[0], point[1], t, color="k")#color="r")
    print(t, point)

# for t,point in zip(Vocabulary, SV):
#     plt.text(point[0], point[1], t, color="b")

plt.text(query[0], query[1], "*", color = "r")
plt.xlim(US[:,0].min(), US[:,0].max())
plt.ylim(US[:,1].min(), US[:,1].max())
plt.show()
```

```
man [-1.05550031 -1.37047342]
king [-1.12313423 -0.9419351 ]
boy [-1.12538134 -0.06751978]
princess [-0.87691625 -0.50135907]
girl [-1.09145195 -0.17128992]
woman [-0.76778117 -0.81011387]
young [-2.14771411  1.68108232]
wise [-0.48047132  0.48947834]
queen [-1.00699081 -0.6639009 ]
prince [-0.88610171 -0.53581818]
strong [-0.87555418  1.22298025]
pretty [-0.2078559   0.26901575]
```



In [34]:

```
US[Vocabulary.index("king")] - US[Vocabulary.index("man")] + US[Vocabulary.index("queen")]
```

Out[34]:

```
array([-1.07462473, -0.23536258])
```

In [35]:

```
V = len(Vocabulary)

queryMat = np.repeat(query, V, axis=0).reshape(-1, V).T

for t,dist in zip(Vocabulary, np.linalg.norm(US-queryMat, axis=1)):
    if t not in ["prince", "boy", "girl"]:
        print(t, dist)
```

```
man 0.7586404384776863
king 0.40599745966580897
princess 0.14042643920783562
woman 0.19026516127548104
young 2.6578075027492596
wise 1.188677055870589
queen 0.15671587984852015
strong 1.8627153168575021
pretty 1.1138693793791246
```

In [36]:

```
'https://www.slideshare.net/ssuser06e0c5/i-64267027'
```

Out[36]:

```
'https://www.slideshare.net/ssuser06e0c5/i-64267027'
```

In [37]:

```
'''
Input = WINDOW*2 단어 필요 => 벡터표현(one-hot)
[WINDOW*2, len(V)], center_wowrd[1, len(V)]
WIBDOW * 2의 문맥단어, 단어 => CBoW(keras)
'''
```

Out[37]:

```
'WnInput = WINDOW*2 단어 필요 => 벡터표현(one-hot)Wn[WINDOW*2, len(V)], center_wow
rd[1, len(V)]WnWIBDOW * 2의 문맥단어, 단어 => CBoW(keras)Wn'
```

In [38]:

```
onehot = np.diag(np.ones(V))
onehot[0]
```

Out[38]:

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

In [39]:

```
word2idx = lambda t:Vocabulary.index(t)
idx2word = lambda i:Vocabulary[i]
idx2vec = lambda i:onehot[i]
```

In [40]:

```
word2idx("man"), idx2vec(word2idx("man"))
```

Out[40]:

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

In [41]:

```
def getPair(D, W=2):
    pairList = list()
    for termList in D:
        for i in range(len(termList)):
            s = i - W
            e = i + W + 1
            for j in range(s,e):
                if -1 < j < len(termList) and j != i:
                    pairList.append((termList[j], termList[i]))
    return pairList
```

In [42]:

```
_documents[0]
```

Out[42]:

```
['king', 'strong', 'man']
```

In [43]:

```
inputList = list()
outputList = list()

for pair in getPair(_documents, 2):
    inputList.append(idx2vec(word2idx(pair[0])))
    outputList.append(idx2vec(word2idx(pair[1])))
#     _input = idx2vec(word2idx(pair[0]))
#     _output = idx2vec(word2idx(pair[1]))
#     print(_input)
#     print(_output)
#     break
```

In [44]:

```
len(inputList), len(outputList)
```

Out[44]:

```
(52, 52)
```

In [45]:

```
np.array(inputList).shape
```

Out[45]:

```
(52, 12)
```

In [ ]: