In [1]:

```
import os
os.chdir('/Users/hanjudong/work/W2018_HYU_TEXT_MINING_LECTURE_2')
#os.chdir('C:/Users/hanjudong/work/W2018_HYU_TEXT_MINING_LECTURE_2')
os.getcwd()
```

Out[1]:

```
'/Users/hanjudong/work/W2018_HYU_TEXT_MINING_LECTURE_2'
```

Introduction to Text Mining

2018.10.29 Hanyang University

6. Vector Semantics

: Dense Vector Representation

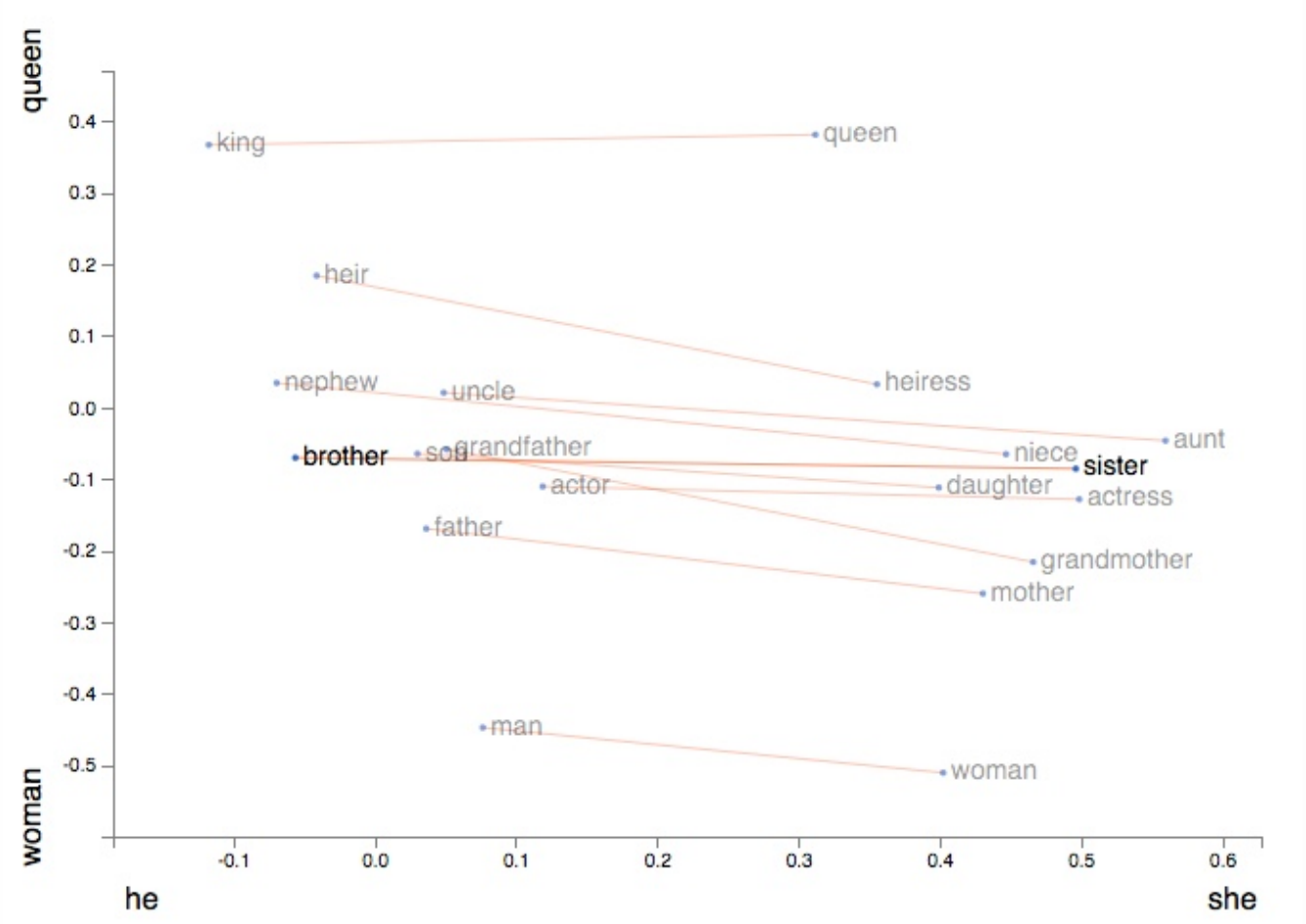
- ***The Curse of Dimensionality***
- ***High-Dimensional Problems:*** $p \gg N$

The Elements of Statistical Learning: Data Mining, Inference, and Prediction 2E -Chapter 18 참고

Example) - 노벨경제학상, 물리학상 연설문 : N = 18, p = 15,594

-기업의 부도율 : 시군구지역별(230)x자산규모별(10)x매출액 규모별(10)x표준산업코드별(1,197) => 27,531,000 segments
그런데, 우리나라 기업의 수는?

king - man + woman = queen



7. 장난감 모델

7-1. 기업에 대한 뉴스를 통해 구축한 언어 모델

- 2008년 12월 ~ 2016년 9월 주요 경제지의 온라인 뉴스 제목과 기업을 연결

- 뉴스 발표일로부터 1년 이내 기업에 부도가 발생하면 부실기업 뉴스(=1)로 그렇지 않으면 정상기업 뉴스(=0)로 구분

구분	뉴스건수	분류별 구성비	기업 수	전체 단어 수	중복 제거 단어수
정상기업 뉴스	958,297	95.97%	13,873사	10,956,581	74,723
부실기업 뉴스	40,237	4.03%	1,069사	444,556	14,896
총계	998,534	100.00%	14,372사	11,398,137	75,939

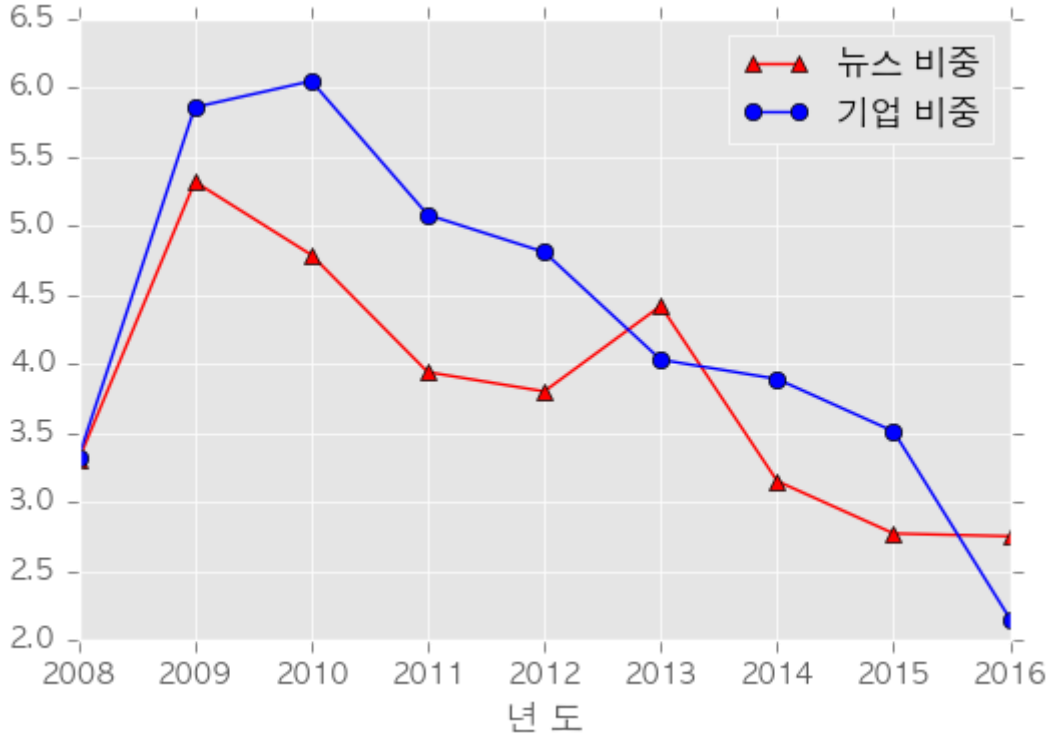
- 년도별 정상/부실 기업 뉴스 구성

년도	정상 뉴스 수	부실 뉴스 수	합계	부실 뉴스 구성비
2008	13,357	456	13,813	3.30%
2009	156,033	8,775	164,808	5.32%
2010	155,932	7,850	163,782	4.79%
2011	115,610	4,740	120,350	3.94%
2012	140,945	5,565	146,510	3.80%
2013	99,096	4,582	103,678	4.42%
2014	95,163	3,092	98,255	3.15%
2015	104,580	2,984	107,564	2.77%
2016	77,581	2,193	79,774	2.75%
합계	958,297	40,237	998,534	4.03%

- 년도별 정상/부실 기업 구성

년도	정상 기업 수	부실 기업 수	합계	부실 기업 구성비
2008	2,008	69	2,077	3.32%
2009	5,043	314	5,357	5.86%
2010	4,952	319	5,271	6.05%
2011	4,352	233	4,585	5.08%
2012	4,985	252	5,237	4.81%
2013	4,644	195	4,839	4.03%
2014	5,044	204	5,248	3.89%
2015	4,999	182	5,181	3.51%
2016	4,602	101	4,703	2.15%

- 년도별 부실 기업과 부실 기업 뉴스 비중



- Word2Vec 방법으로 뉴스로부터 한국어를 학습한 toy_dense_model 을 불러와 이리저리 살펴봅시다.

주의 : 예시는 Cherry picking 결과입니다.

In [169]:

```
import pickle
import pprint
from konlpy.tag import Twitter
pos_tagger = Twitter()

def tokenize(doc):
    return['/'.join(t)for t in pos_tagger.pos(doc, norm=True, stem=True)]

from gensim.models import Doc2Vec
from gensim.models.doc2vec import TaggedDocument
from collections import namedtuple
```

In [170]:

```
toy_dense_model = Doc2Vec.load('Doc2Vec(dbow+w,d400,n5,hs,w5,mc5,s0.001,t8).model')
```

```
//anaconda/lib/python3.5/site-packages/gensim/models/doc2vec.py:531: U
serWarning: The parameter `iter` is deprecated, will be removed in 4.
0.0, use `epochs` instead.
  warnings.warn("The parameter `iter` is deprecated, will be removed i
n 4.0.0, use `epochs` instead.")
//anaconda/lib/python3.5/site-packages/gensim/models/doc2vec.py:535: U
serWarning: The parameter `size` is deprecated, will be removed in 4.
0.0, use `vector_size` instead.
  warnings.warn("The parameter `size` is deprecated, will be removed i
n 4.0.0, use `vector_size` instead.")
```

In [176]:

```
toy_dense_model?
```

In [171]:

```
pprint.pprint(toy_dense_model.wv.most_similar('매출/Noun'))
```

```
[('액/Noun', 0.5762389898300171),
 ('이익/Noun', 0.4915822148323059),
 ('익/Noun', 0.47872430086135864),
 ('영업/Noun', 0.4716307520866394),
 ('순이익/Noun', 0.4596134424209595),
 ('순익/Noun', 0.4572048783302307),
 ('사상/Noun', 0.43765807151794434),
 ('연매출/Noun', 0.4212663173675537),
 ('달성/Noun', 0.4144240915775299),
 ('당다/Verb', 0.37733155488967896)]
```

In [172]:

```
pprint.pprint(toy_dense_model.wv.most_similar('부실/Noun'))
```

```
[('여신/Noun', 0.44594645500183105),
 ('PF/Alpha', 0.39537885785102844),
 ('NPL/Alpha', 0.3409172296524048),
 ('충당금/Noun', 0.3234788775444031),
 ('부동산/Noun', 0.3188086748123169),
 ('채권/Noun', 0.3164753317832947),
 ('공적자금/Noun', 0.3102889955043793),
 ('징후/Noun', 0.30630940198898315),
 ('정리/Noun', 0.30606162548065186),
 ('예보/Noun', 0.3011745512485504)]
```

In [173]:

```
pprint.pprint(toy_dense_model.wv.most_similar(positive=['금리/Noun','상승/Noun'],negat
```

```
('하락/Noun', 0.3773341774940491)
```

In [174]:

```
pprint.pprint(toy_dense_model.wv.most_similar(positive=['금리/Noun','상승/Noun'],negat
```

```
('하락/Noun', 0.39649254083633423)
```

In [175]:

```
pprint.pprint(toy_dense_model.wv.most_similar(positive=['금리/Noun','하락/Noun'],negat
```

```
('상승/Noun', 0.41907522082328796)
```

In [166]:

```
pprint.pprint(toy_dense_model.wv.most_similar(positive=['금리/Noun','보합/Noun'],negat
```

```
[('p/Alpha', 0.372775673866272),
 ('등락/Noun', 0.3579190969467163),
 ('초반/Noun', 0.33119136095046997),
 ('중반/Noun', 0.32870686054229736),
 ('힘겨루기/Noun', 0.3249766230583191),
 ('인하/Noun', 0.31609907746315),
 ('혼/Noun', 0.3082966208457947),
 ('금리인하/Noun', 0.303908109664917),
 ('기준금리/Noun', 0.30262070894241333),
 ('1570/Number', 0.2987695038318634)]
```

In [167]:

```
pprint.pprint(toy_dense_model.wv.most_similar(negative=['농심/Noun','라면/Noun'],posit
```

```
('약품/Noun', 0.22793251276016235)
```

In [168]:

```
pprint.pprint(toy_dense_model.wv.most_similar(negative=['한미/Noun','약품/Noun'],posit
```

```
('마트/Noun', 0.21412473917007446)
```

- 2차원 공간에 시각화 해봅니다.

In [227]:

```
# 2차원 공간에 시각화하고 싶은 문자열을 써봅니다. word tags는 무시해도 되나 값은 넣어 줍니다.
vizdt = [
         ['대형 공사계약 상한가', '2'], ['대형 매출계약 상한가 외국인 투자자 유입', '2'],
         ['업황 악화로 인한 영업부진 지속 안정성 악화', '2'],['대표이사 배임 횡령 주가 급락', '2'],
         ['법정관리와 워크아웃 기로에서 우왕좌왕', '2'],
         ['신약 개발 성공 FDA 승인 실패', '2'] ,['신약 개발 FDA 승인 성공 기대감 증대', '2']
        ]
# 시각화시키고 싶은 문자열을 한국어 품사태깅을 하여 토큰화 합니다.
doc_all   = [(tokenize(row[0]),row[1]) for row in vizdt ]
# Gensim 라이브러리가 원하는 형태로 자료를 변형시켜 줍시다.
TaggedDocumnet = namedtuple("TaggedDocument","words tags")
qlist = [TaggedDocument(a,b) for a,b in doc_all]

# toy_dense_model에 시각화 시키고 싶은 문자열을 input으로 넣어서 400 dim의 vector를 얻습니다.
viz_x = []
for doc in qlist:
    toy_dense_model.random.seed(1234)
    viz_x.append(toy_dense_model.infer_vector(doc.words))
# 입력한 문자열의 갯수와 toy_dense_model을 통해 생성한 vector의 크기를 확인해 봅니다.
print(len(viz_x), len(viz_x[0]))
```

```
7 400
```

In [228]:

```
# 400 dim을 2 dim에 시각화하기 위해 PCA를 통해 축소 시켜봅니다.
import numpy as np
from sklearn.decomposition import PCA
from sklearn.preprocessing import MinMaxScaler

input_x = np.array(viz_x)

# n_components 2로 dim을 지정하고 PCA를 적용합니다.
pca = PCA(n_components=2)
pca.fit(input_x)
trf_x = pca.transform(input_x)

# 시각화를 위해 최대값, 최소값을 0~1로 스케일을 조정합니다.
min_max_scaler = MinMaxScaler()
trf_adj_x = min_max_scaler.fit_transform(trf_x)
print(trf_adj_x)
```

```
[[ 0.37837952  0.95637226]
 [ 0.4320024   1.        ]
 [ 1.          0.        ]
 [ 0.55214983  0.88746619]
 [ 0.43684095  0.72060841]
 [ 0.0473215   0.28082862]
 [ 0.          0.1185436 ]]
```

In [229]:

```
# 이번에는 pandas를 이용해서 시각화를 해봅시다.
import pandas as pd
# PCA를 통해 2차원으로 변형된 데이터를 pandas DataFrame으로 바꿉니다.
value_df = pd.DataFrame(trf_adj_x, columns=['x', 'y'])
# 시각화할 문자열을 DataFrame에 추가합니다.
label = [(row[0]) for row in vizdt ]
label_df = pd.DataFrame(label, columns=['label'])
viz_df = label_df.join(value_df)
```

In [230]:

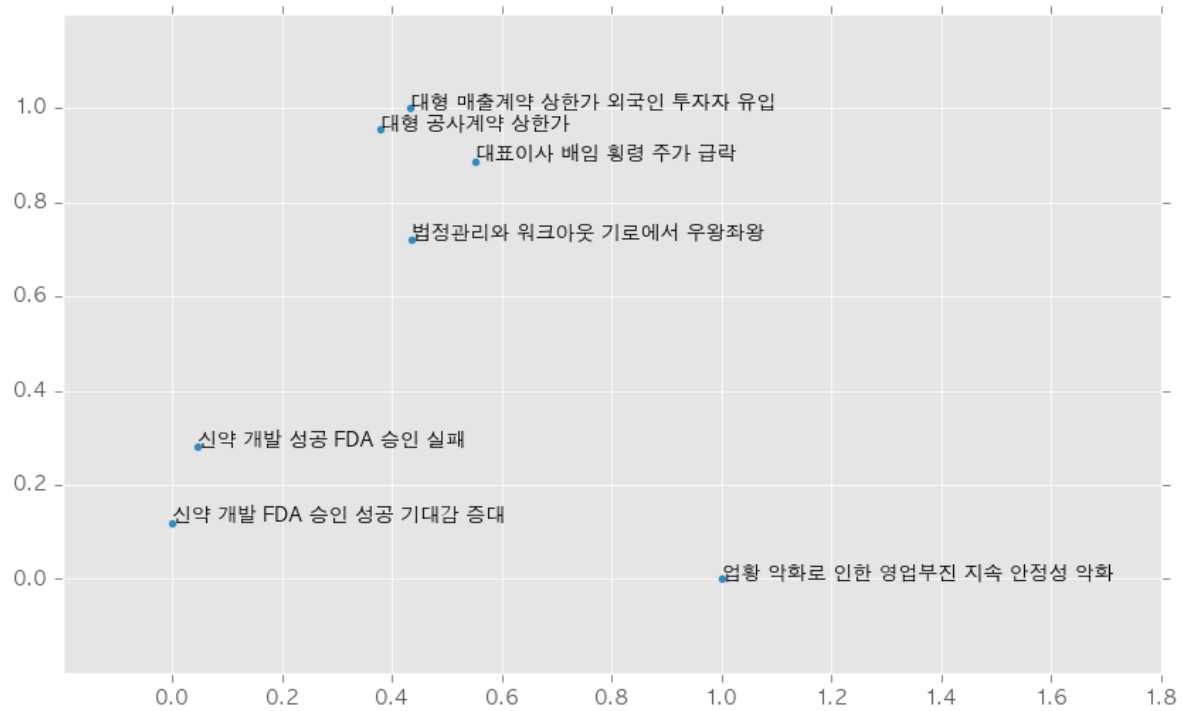
```
viz_df
```

Out[230]:

	label	x	y
0	대형 공사계약 상한가	0.378380	0.956372
1	대형 매출계약 상한가 외국인 투자자 유입	0.432002	1.000000
2	업황 악화로 인한 영업부진 지속 안정성 악화	1.000000	0.000000
3	대표이사 배임 횡령 주가 급락	0.552150	0.887466
4	법정관리와 워크아웃 기로에서 우왕좌왕	0.436841	0.720608
5	신약 개발 성공 FDA 승인 실패	0.047321	0.280829
6	신약 개발 FDA 승인 성공 기대감 증대	0.000000	0.118544

In [246]:

```
%matplotlib inline
import matplotlib ; import matplotlib.pyplot as plt
from matplotlib import font_manager, rc
font_fname = '/Library/Fonts/AppleGothic.ttf' ; font_name = font_manager.FontPropert
# C:/Windows/Fonts/Arial.ttf
rc('font', family=font_name, size=10) ; rc('figure',figsize=(10,6))
plt.style.use('ggplot')
ax = viz_df.plot('x', 'y', kind='scatter') ; viz_df[['x','y','label']].apply(lambda
plt.xlabel("") ; plt.ylabel("")
plt.xticks(np.arange(0,2,0.2)) ; plt.yticks(np.arange(0,1.2,0.2))
plt.show()
```



- toy_dense_model에게 부도 위험을 판별하게 할 수 있을까요?

In [289]:

```
XY_name = {}
XY_name['train_x_Doc2Vec(dbow+w,d400,n5,hs,w5,mc5,s0.001,t8)'] = pickle.load(open('t
XY_name['train_y_Doc2Vec(dbow+w,d400,n5,hs,w5,mc5,s0.001,t8)'] = pickle.load(open('t
train_x = XY_name['train_x_Doc2Vec(dbow+w,d400,n5,hs,w5,mc5,s0.001,t8)']
train_y = XY_name['train_y_Doc2Vec(dbow+w,d400,n5,hs,w5,mc5,s0.001,t8)']
```

In [290]:

```
import time
from sklearn import linear_model
start_time = time.time()
toy_dense_risk_model = linear_model.LogisticRegression(random_state=1234)
toy_dense_risk_model.fit(train_x,train_y)
print("--- %0.2f seconds ---" % (time.time() - start_time))
```

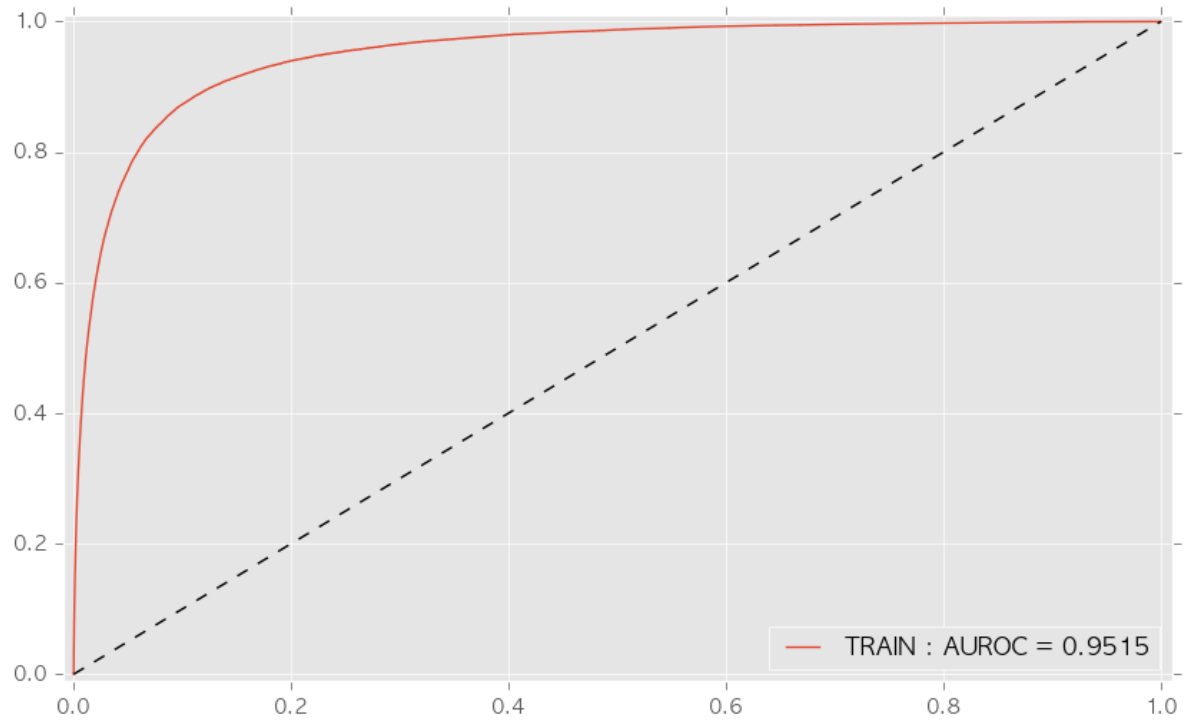
```
--- 51.93 seconds ---
```

In [291]:

```
from sklearn.metrics import roc_curve, auc
x_train = train_x; y_train = np.array([int(y) for y in train_y])
pred_probas_train = toy_dense_risk_model.predict_proba(x_train)[:,1]

fpr_train,tpr_train , _ = roc_curve(y_train, pred_probas_train)
roc_auc_train = auc(fpr_train,tpr_train)

plt.plot(fpr_train,tpr_train,label='TRAIN : AUROC = %.4f' %roc_auc_train)
plt.plot([0, 1], [0, 1], 'k--'); plt.xlim([-0.01, 1.01]); plt.ylim([-0.01, 1.01]); p
plt.show()
```



In [292]:

```
# 시간이 오래걸리는 결과물은 pickle로 쓰고 다음에 작업할때는 pickle로 불러 옵시다.
import pickle
#with open('toy_dense_risk_model', 'wb') as f:
#    pickle.dump(toy_dense_risk_model, f)
#del XY_name, train_x, train_y, toy_dense_risk_model

with open ('toy_dense_risk_model', 'rb') as f:
    toy_dense_risk_model = pickle.load(f)
```

In [293]:

```
data = [['대형 공사계약', '2'], ['대형 판매계약', '2'], ['대형 매출계약', '2'] ]
doc_all   = [(tokenize(row[0]),row[1]) for row in data ]
TaggedDocumnet = namedtuple("TaggedDocument","words tags")
qlist = [TaggedDocument(a,b) for a,b in doc_all]

test_x = []
for doc in qlist:
    toy_dense_model.random.seed(1234)
    test_x.append(toy_dense_model.infer_vector(doc.words))
print(toy_dense_risk_model.predict_proba(test_x)[:,1])
```

```
[ 0.04501267  0.04485619  0.04245259]
```

In [294]:

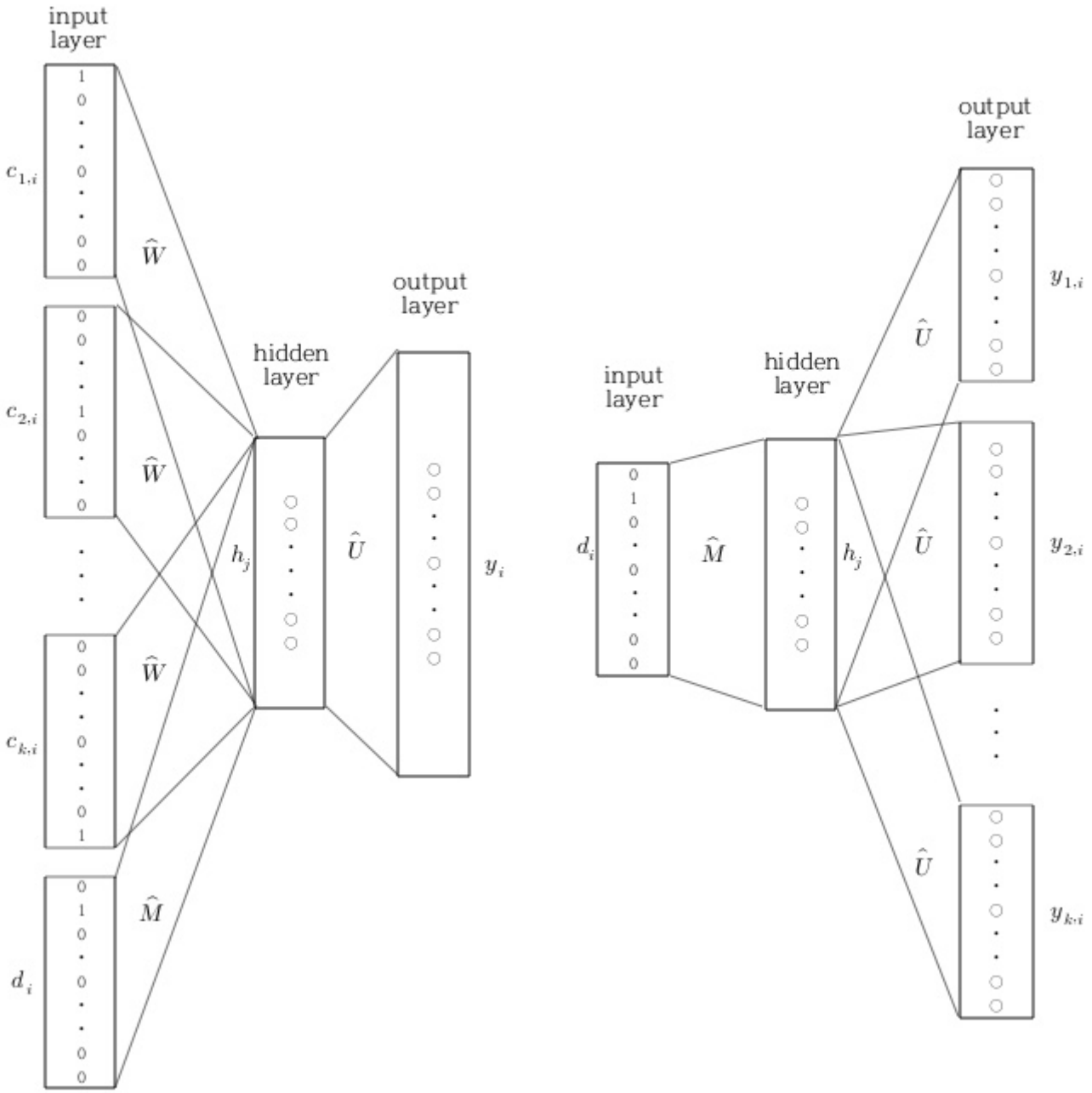
```
data = [ ['영업부진 지속', '2'], ['대표이사 배임 횡령 주가 급락', '2'], ['법정관리와 워크아웃 기로에
doc_all   = [(tokenize(row[0]),row[1]) for row in data ]
TaggedDocumnet = namedtuple("TaggedDocument","words tags")
qlist = [TaggedDocument(a,b) for a,b in doc_all]

test_x = []
for doc in qlist:
    toy_dense_model.random.seed(1234)
    test_x.append(toy_dense_model.infer_vector(doc.words))
print(toy_dense_risk_model.predict_proba(test_x)[:,1])
```

```
[ 0.04876137  0.06677477  0.08226433]
```

8. Word2Vec & Doc2Vec

8.1 Word2Vec & Doc2Vec 설계



- Doc2Vec : -PV-DM(distributed memeory) - PV-DBOW(distributed bag of words)
- Word2Vec : -CBOW(continuous bag of words) - Skip-gram

- $c \in \mathbb{R}^{1 \cdot V}$: $word_i$ 의 주변 맥락 단어 벡터(one-hot-encoding), k개 까지 고려, V는 단어사전의 크기
- $d \in \mathbb{R}^{1 \cdot N}$: 문서의 id 벡터(one-hot-encoding), N은 문서의 개수
- $h \in \mathbb{R}^{1 \cdot P}$: hidden layer 벡터(= $c \cdot \hat{W}$, = $d \cdot \hat{M}$), P는 hyper-parameter로 문서를 표현하는 벡터 크기
- $y \in \mathbb{R}^{1 \cdot V}$: output layer로 단어 등장에 대한 예측 확률값 벡터 $\approx Logit(\hat{U} \cdot h)$
- $\hat{W} \in \mathbb{R}^{V \cdot P}$: 단어 input layer -> hidden layer 가중치 행렬
- $\hat{M} \in \mathbb{R}^{N \cdot P}$: 문서 input layer -> hidden layer 가중치 행렬
- $\hat{U} \in \mathbb{R}^{P \cdot V}$: hidden layer -> output layer 가중치 행렬, 즉 단어별로는 $P \cdot 1$의 벡터가 할당

8.2 개념 및 계산

$$\arg\max_{W,M,U} \Pi P(w_t|c,d)$$

where

$$P(w_t|c,d) = \frac{1}{1+e^{-(c\cdot(WorM)\cdot U)}}$$

c 은 one-hot-encoding 임으로

$$= \frac{1}{1+e^{-((WorM)\cdot U)}}$$

- 문맥 단어 c 혹은 문서 d 가 주어졌을 때, 타겟 단어 w_t 가 나타날 확률을 극대화하는 $\hat{W}, \hat{M}, \hat{U}$를 찾는데, 그것은 ($\hat{W}, \hat{M}$) 과 $\hat{U}$간 내적(inner product)을 최대로 하는 벡터이다.

- $\hat{W}, \hat{M}, \hat{U}$ 이 단어, 혹은 문서를 표현하는 벡터가 된다. $\hat{U}$만을 활용하기도 하고, $(\hat{W}or\hat{M}) + \hat{U}$로 이용하거나 두 벡터를 쌓아서(concatenate) 활용하기도 한다.

- 함께 잘 등장하지 않을 확률을 최소화하는 항을 추가하는데, 이를 negative sampling이라고 한다. 이때 극대화 해야하는 함수는 다음과 같다.

$$L = log\frac{1}{1+e^{-(c\cdot(WorM)\cdot U)}} + \sum_{i=1}^{k} log\frac{1}{1+e^{(n_i\cdot(WorM)\cdot U)}}$$

where n_i : negative words, w_t와 함께 잘 등장하지 않는 단어로 Randomly sampling한다.

- 일반적인 인공신경망(neural network)과 마찬가지로 초기값으로 임의의 $\hat{W}, \hat{M}, \hat{U}$를 부여한 후 확률적 경사하강법(stochastic gradient descent)를 이용하여 목적함수를 극대화하는 파라미터를 찾는다. 즉, 예측값과 실제값의 차이에 일정한 learning rate만큼 조정하여 back-propagation을 통해 파라미터를 조정한다.

- word vector는 모든 문서에 대해 공유된다. 즉 같은 단어는 각각 다른 문서에서 같은 벡터값을 갖는다. 반대로 문서 벡터는 다른 문서간에는 공유되지 않는다.

- 파라미터를 모델에 추가적인 문서, 단어가 등장했을 때, 기존에 추정된 파라미터는 고정한 채로, 새로운 문서나 단어를 나타내는 one-hot-encoding을 추가하고 다시 확률적 경사하강법을 이용하여 새로운 문서, 단어를 표현하는 벡터를 얻는다.

- 추정 결과를 평가하는 방법은 기존에 전문가들이 별도로 구축해놓은 유사도 데이터 셋과 비교하는 방법이 있다. 이를테면

In [300]:

```
from IPython.display import HTML
HTML('<iframe src=https://ronxin.github.io/wevi/ width=1000 height=1100></iframe>')
```

Out[300]:

wevi: word embedding visual inspe

Everything you need to know about this tool (http://bit.ly/wevi-help) - Source code (http://bit

Control Panel

Config:

```
{"hidden_size":5,"random_state":1,"learning_rate":0.2}
```

Training data (context|target):

```
king|kindom,queen|kindom,king|palace,queen|palace,king|royal,queen|royal,king|George,queen|Mary,man|rice,woman|rice,man|farmer,woman|farmer,man|house,w
```

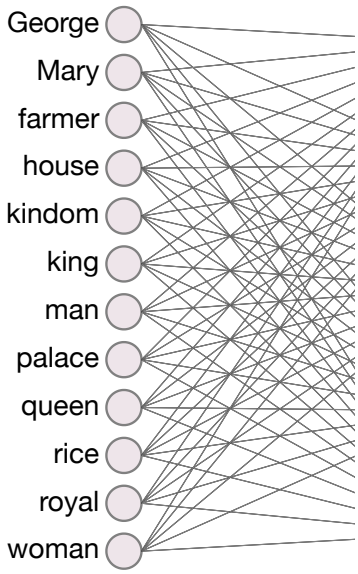
Presets: King and queen

Update and Restart

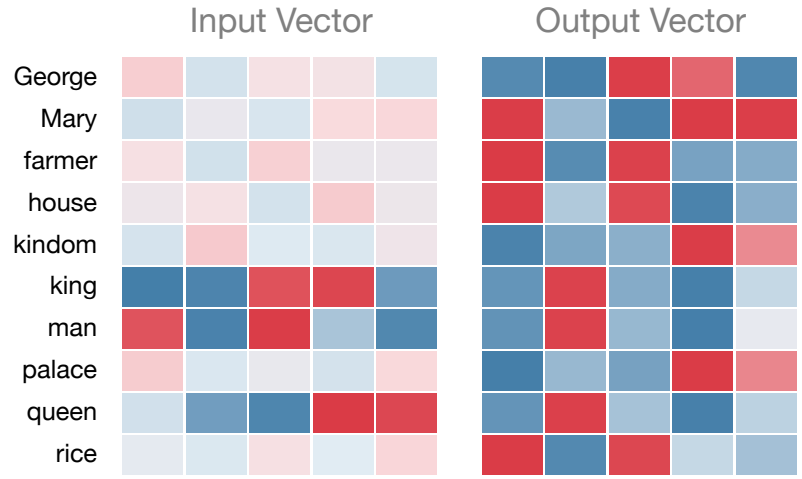
Update Learning Rate

Next | 20 | 100 | 500 | PCA

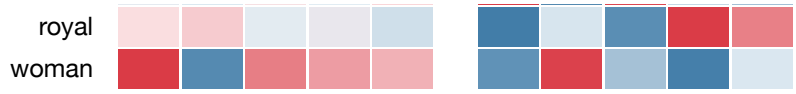
Neurons



Weight Matrices



Vectors



8.3 Embedding방식으로 편향된 시각을 학습하는 경우 : 윤리적 이슈

- 성/인종 차별적 Embedding : man - computer programmer + woman = homemaker

- gender와 관련한 고정관념에 해당하는 부분을 제거하는 방법, 혹은 historical 변화 과정을 추적하거나, 고정관념이 없는 문서의 가중치를 이용하는 방법 등이 연구되고 있음.

- 그 외에도 데이터를 통한 작업 후 이 것이 어떻게 활용되고 어떠한 영향을 미칠 것인지는 염두에 두어야함.

9. Gensim을 이용해 볼 만한 주제들

In [303]:

```
from IPython.display import HTML
HTML('<iframe src=https://radimrehurek.com/gensim/apiref.html width=1000 height=110(
```

Out[303]:

- topic_coherence.probability_estimation – Probability estimation modu
- topic_coherence.segmentation – Segmentation module
- topic_coherence.text_analysis – Analyzing the texts of a corpus to accu
- scripts.package_info – Information about gensim package
- scripts.glove2word2vec – Convert glove format to word2vec
  - How to use
  - Command line arguments
- scripts.make_wikicorpus – Convert articles from a Wikipedia dump to vect
- scripts.word2vec_standalone – Train word2vec on text file CORPUS
- scripts.make_wiki_online – Convert articles from a Wikipedia dump
- scripts.make_wiki_online_lemma – Convert articles from a Wikipedia dum
- scripts.make_wiki_online_nodebug – Convert articles from a Wikipedia du
- scripts.word2vec2tensor – Convert the word2vec format to Tensorflow 2D
  - How to use
  - Command line arguments
- scripts.segment_wiki – Convert wikipedia dump to json-line format
  - How to use
  - Command line arguments
- parsing.porter – Porter Stemming Algorithm
  - Examples:
- parsing.preprocessing – Functions to preprocess raw text
  - Examples:
  - Data:
- summarization.bm25 – BM25 ranking function
  - Data:
- summarization.commons – Common graph functions
- summarization.graph – Graph
- summarization.keywords – Keywords for TextRank summarization algorithm
  - Data:
- summarization.mz_entropy – Keywords for the Montemurro and Zanette ent
- summarization.pagerank_weighted – Weighted PageRank algorithm
- summarization.summarizer – TextRank Summariser
  - Data
- summarization.syntactic_unit – Syntactic Unit class
- summarization.textcleaner – Summarization pre-processing
  - Data
- viz.poincare – Visualize Poincare embeddings

gensim

Home | Tutorials | Install | Support |

Text Rank를 이용한 extraction 방식의 문서 요약

In [306]:

```
text = '''Rice Pudding - Poem by Alan Alexander Milne
 What is the matter with Mary Jane?
 She's crying with all her might and main,
 And she won't eat her dinner - rice pudding again -
 What is the matter with Mary Jane?
 What is the matter with Mary Jane?
 I've promised her dolls and a daisy-chain,
 And a book about animals - all in vain -
 What is the matter with Mary Jane?
 What is the matter with Mary Jane?
 She's perfectly well, and she hasn't a pain;
 But, look at her, now she's beginning again! -
 What is the matter with Mary Jane?
 What is the matter with Mary Jane?
 I've promised her sweets and a ride in the train,
 And I've begged her to stop for a bit and explain -
 What is the matter with Mary Jane?
 What is the matter with Mary Jane?
 She's perfectly well and she hasn't a pain,
 And it's lovely rice pudding for dinner again!
 What is the matter with Mary Jane?'''
```

In [308]:

```
news = '''종합부동산세 인상안 발표 후 서울 집값 상승세가 수개월 만에 재점화하고 있다. 내리막세였던 서울 강
19일 한국감정원에 따르면 7월 셋째주 서울 아파트 매매가격이 3주 연속 상승하며 올해 누적으로 4.05% 상승했다.
지난 4월 양도세 중과 시행 후 하락세로 돌아섰던 강남 4구 아파트 매매는 15주 만에 처음으로 반등했다. 세 번째
조용했던 은평구도 최근 GTX(수도권 광역급행철도) 개통과 재개발 기대감이 시너지를 내며 7월 셋째주 0.22% 올라
반면 지방 부동산시장은 `마이너스 늪`으로 더 깊이 빠져들고 있다. 올 들어 7월 셋째주까지 지방 아파트 가격은 2.
지방 부동산 맹주였던 부산도 올해 2.12% 하락한 것으로 나타났다. 보유세 인상안 발표 후 지방 집값이 더 하락할
'''
```

In [311]:

```
from gensim.summarization.summarizer import summarize

print(summarize(text, word_count= 10))
print(summarize(news))
```

```
And she won't eat her dinner - rice pudding again -
19일 한국감정원에 따르면 7월 셋째주 서울 아파트 매매가격이 3주 연속 상승하며 올해 누적으로
4.05% 상승했다.
지난 4월 양도세 중과 시행 후 하락세로 돌아섰던 강남 4구 아파트 매매는 15주 만에 처음으로
반등했다.
조용했던 은평구도 최근 GTX(수도권 광역급행철도) 개통과 재개발 기대감이 시너지를 내며 7월 셋
째주 0.22% 올라 25개 자치구 중 두 번째로 높은 집값 상승률을 기록했다.
```

In [319]:

```
from gensim.summarization.keywords import get_graph
from gensim.summarization.pagerank_weighted import pagerank_weighted
from gensim.summarization.pagerank_weighted import build_adjacency_matrix
graph = get_graph(news)
# result will looks like {'good': 0.70432858653171504, 'hell': 0.051128871128006126,
result = pagerank_weighted(graph)
pprint.pprint(result)
```

```
{'gtx': 0.060900821397220598,
 '가격은': 0.10682736863216327,
 '가까이': 0.012612679361388959,
 '가팔라지고': 0.012612679361388985,
 '강남구': 0.10974710471582214,
 '강남권': 0.094562854243724001,
 '강남에선': 0.10239122551018501,
 '강화까지': 0.099746921218232226,
 '개통과': 0.10471147263163276,
 '거래됐다': 0.065975130591174963,
 '거래였던': 0.094562854243723946,
 '거세지고': 0.072611379950255336,
 '것으로': 0.14087489215637441,
 '것이란': 0.13846580283066123,
 '곳곳에선': 0.063099573376405732,
 '광역급행철도': 0.10627341279675866,
 '급매들이': 0.059733598198018462,
 '기대감이': 0.11144001934323768,
 '기록했다': 0.14766876335634405,
 '기록만': 0.094562854243724154
```

In [323]:

```
print(build_adjacency_matrix(graph).todense())
```

```
[[ 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.]]
```

참고문헌

1) 파이썬 일반

- 점프 투 파이썬 (https://wikidocs.net/book/1 (https://wikidocs.net/book/1))
- Coding The Matrix: Linear Algebra Through Computer Science Application (http://codingthematrix.com/ (http://codingthematrix.com/))
- Computationa Physics (http://www-personal.umich.edu/~mejn/cp/ (http://www-personal.umich.edu/~mejn/cp/))

2) Graph 및 Network

- Networks: An Introduction (http://www-personal.umich.edu/~mejn/networks-an-introduction/ (http://www-personal.umich.edu/~mejn/networks-an-introduction/))

3) Vector semantics

- Introduction to Machine Learning with Python (https://github.com/amueller/introduction_to_ml_with_python (https://github.com/amueller/introduction_to_ml_with_python)) (http://www.kyobobook.co.kr/product/detailViewKor.laf?ejkGb=KOR&mallGb=KOR&barcode=9788968483394&orderClick=LAG&Kc= (http://www.kyobobook.co.kr/product/detailViewKor.laf?ejkGb=KOR&mallGb=KOR&barcode=9788968483394&orderClick=LAG&Kc=))
- Speech and Language Processing (3rd ed. draft) (https://web.stanford.edu/~jurafsky/slp3/ (https://web.stanford.edu/~jurafsky/slp3/))

In [ ]: