In [1]:

```
from IPython.display import HTML

HTML('''<script>
code_show=true;
function code_toggle() {
 if (code_show){
 $('div.input').hide();
 } else {
 $('div.input').show();
 }
 code_show = !code_show
}
$( document ).ready(code_toggle);
</script>
<form action="javascript:code_toggle()"><input type="submit" value="Click here to to
```

Out[1]:

Click here to toggle on/off the raw code.

In [2]:

```
import os
os.chdir('/Users/hanjudong/work/W2018_HYU_TEXT_MINING_LECTURE')
#os.chdir('C:/Users/hanjudong/work/W2018_HYU_TEXT_MINING_LECTURE')
os.getcwd()
```

Out[2]:

```
'/Users/hanjudong/work/W2018_HYU_TEXT_MINING_LECTURE'
```

Introduction to Text Mining

2018.10.22 Hanyang University

머신러닝이란 무엇인가?

Text Mining

> **- NLP : Natural Language Processing**
>
> **- NLU : Natural Language Understanding**
>
> **- NLG : Natural Language Generating**

1. Motivation

- Kensho : 자연어 처리와 금융을 결합한 성공 사례

> “골드만삭스의 직원들은 금융분석프로그램 켄쇼(Kensho)를 사용한 후 놀라움 감추지 못했다. 그들이 일주일 동안 매달리거나 사람을 고용해 처리하던 일을 켄쇼는 순식간에 해냈기 때문이다.”
>
> 뉴욕타임스는 2016년 2월 27일(현지시간) 세계적인 금융투자기업 골드만삭스가 이용하는 ‘켄쇼’ 프로그램을 한면에 걸쳐 다루며 “로봇이 월스트리트를 침공(Invading)했다”고 보도했다. 로봇이 인간의 일자리를 빠르게 대체하고 있으며, 금융 · 투자와 같은 전문직도 예외는 아니라는 것이다.
>
> 시리아 내전이 경제에 미치는 영향을 파악하기 위해 켄쇼의 검색 엔진에 ‘시리아 내전 격화(Escalations in The Syrian war)’를 입력하면 켄쇼는 불과 몇 분 안에 미국과 아시아의 주가 변동, 천연가스와 유가의 움직임, 심지어 캐나다 달러의 환율 변화 등 다양한 정보를 일목요연하게 정리해 보여준다.
>
> 켄쇼테크놀로지의 창업자 대니얼 나들러(32)는 “50만달러의 연봉을 받는 전문 애널리스트가 40시간이 걸쳐 하는 작업을 켄쇼는 수분 내 처리할 수 있다”고 말했다.

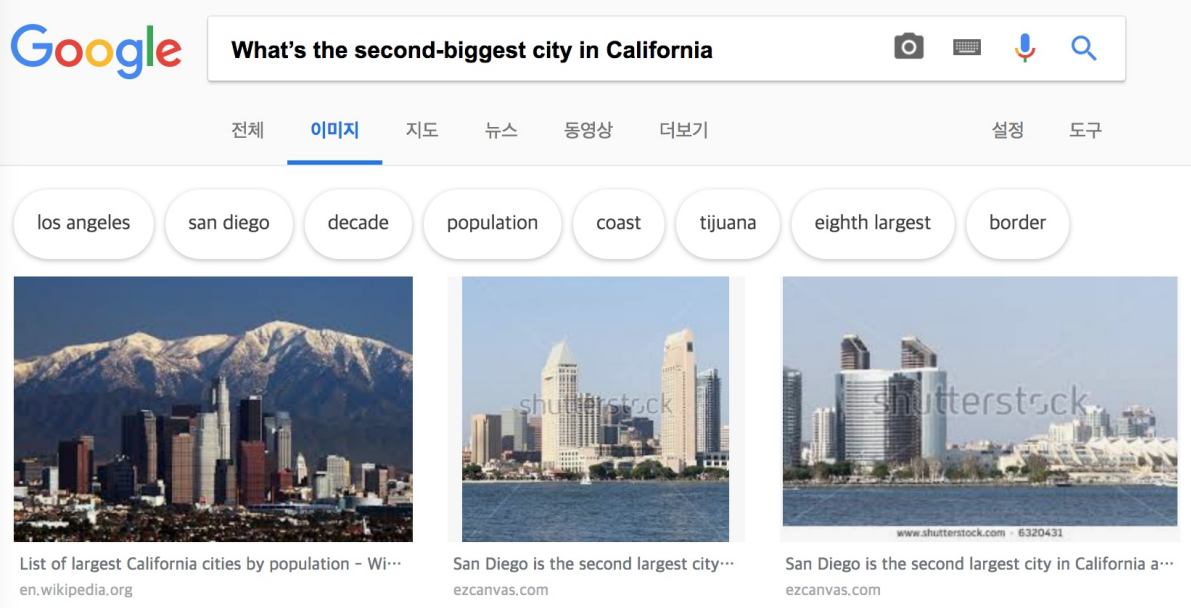
- Kensho : 2018년 3월 S&P글로벌에 인수

> 미국 신용평가회사 S&P글로벌이 5억5000만달러(한화 5880억원)를 투자해 인공지능(AI)으로 금융정보를 분석하는 켄쇼 테크놀로지를 인수했다고 월스트리트저널이 6일(현지시간) 보도했다.
>
> 매사추세츠주 케임브리지에 위치한 켄쇼는 인공지능(AI)을 이용해 금융기관에 데이터분석 자료를 제공한다.
>
> 더글러스 피터슨 S&P 대표는 “켄쇼의 직관적 플랫폼과 정교한 알고리즘, 머신러닝 기능은 월스트리트와 기술 분야에 빠르게 자리 잡았다”면서 “이번 인수로 S&P글로벌은 핀테크 진화에 참여하는 것이 아니라 선도한다는 강력한 의지를 보여줄 것”이라고 말했다.
>
> 켄쇼는 머신러닝 기술을 이용해 구조화되지 않은 웹정보를 체계적 데이터로 변환하고, 해당 데이터와 주식시장 간 상관관계 등을 빠르고 정확하게 이해할 수 있도록 한다. 회사는 이처럼 인공지능 기반 다양한 분석 시스템을 글로벌 금융기관에 제공하고 있다.
>
> 켄쇼는 2014년 1월 구글벤처스 등 주요 투자자로부터 1000만달러를 투자받았고, 같은 해 11월에도 골드만삭스 등으로부터 1500만달러를 투자받은 바 있다.

In [3]:

```
#@hidden_cell
from IPython.display import Image
Image(filename='isecond.jpeg')
#https://spectrum.ieee.org/robotics/artificial-intelligence/machinelearning-maestro-
```

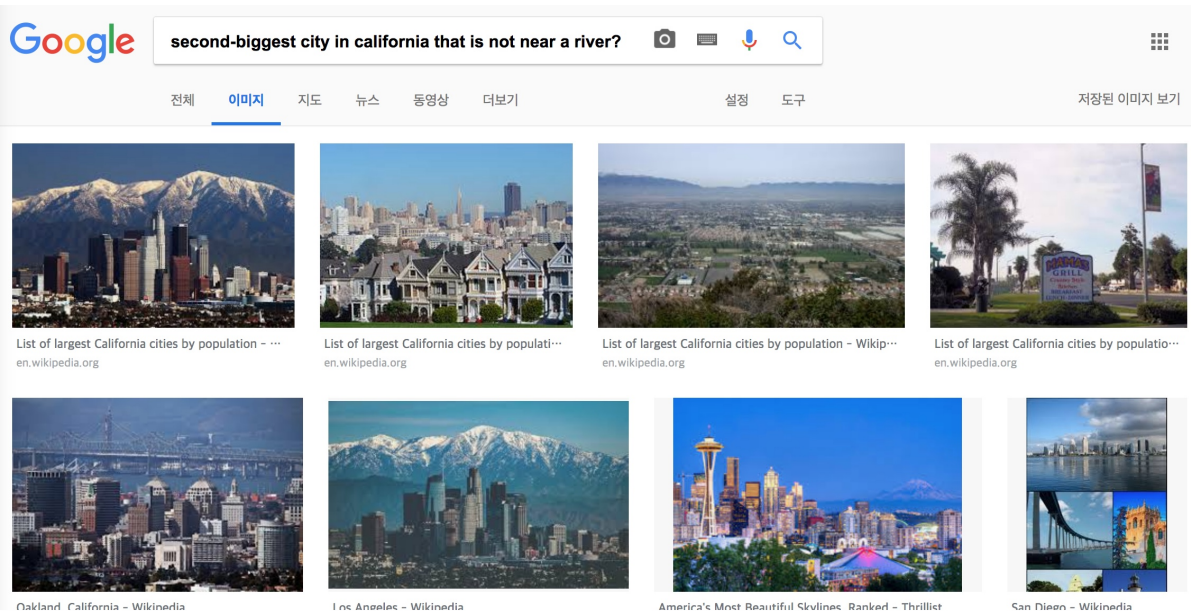
Out[3]:



In [4]:

```
from IPython.display import Image
Image(filename='inot.jpeg')
```

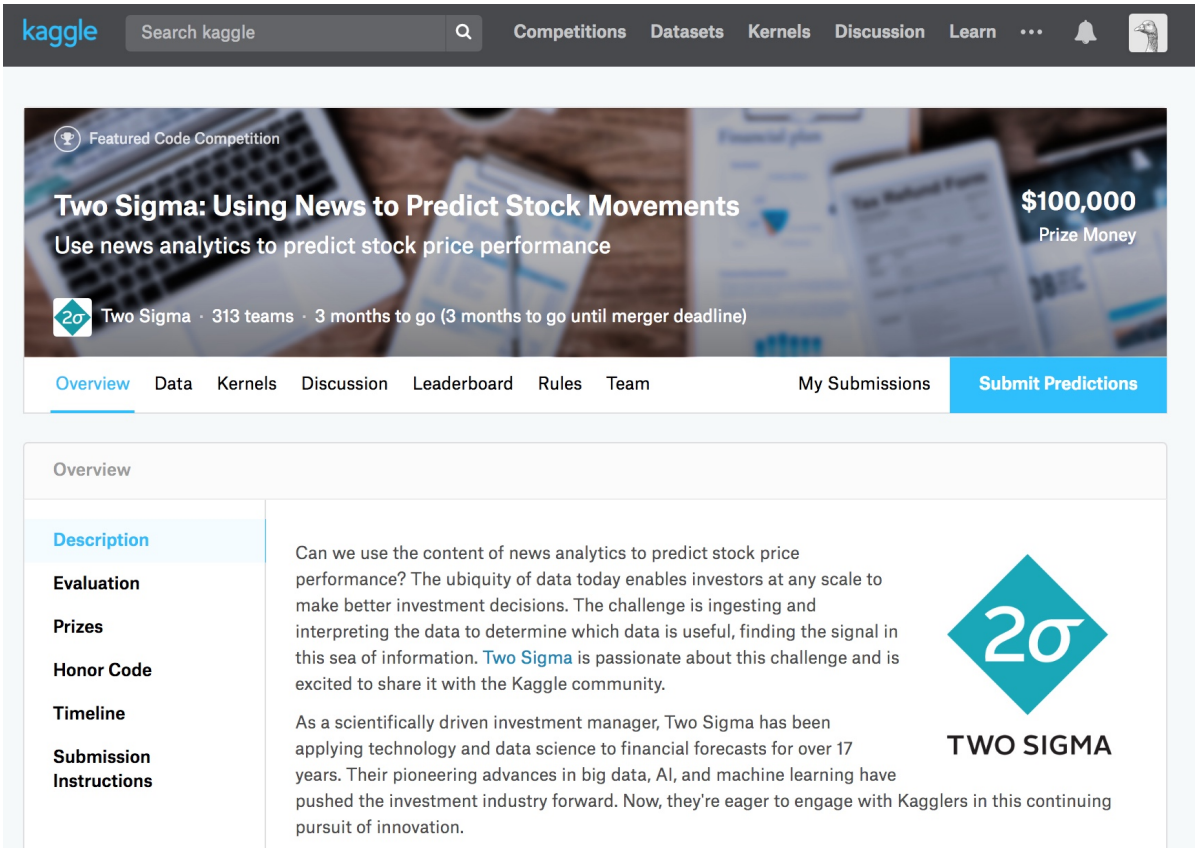
Out[4]:



In [5]:

```
from IPython.display import Image
Image(filename='ikaggle_news.jpeg')
#https://www.kaggle.com/c/two-sigma-financial-news
```

Out[5]:



2. Viewpoints

In [6]:

```
from IPython.display import HTML
HTML('<iframe src=https://ko.wikipedia.org/wiki/%ED%97%88%EB%B2%84%ED%8A%B8_%EC%82%A
```

Out[6]:

WIKIPEDIA

허버트 사이먼

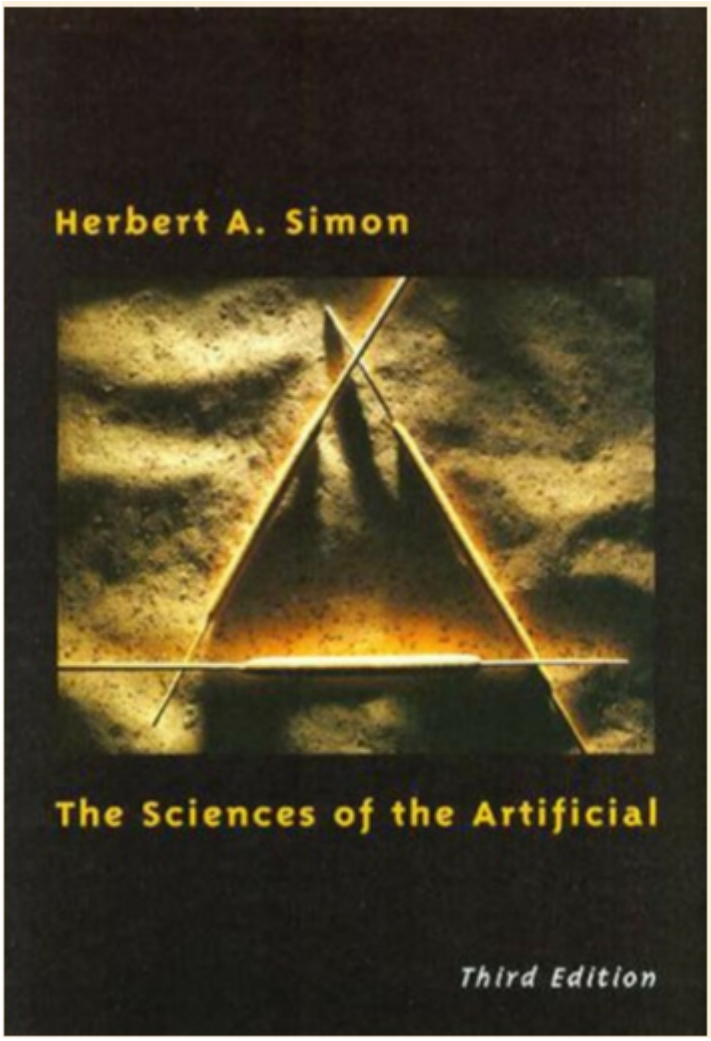
위키백과, 우리 모두의 백과사전.

허버트 알렉산더 사이먼(Herbert Alexander Simon, 1916년 6월 15일 ~ 2001년 2월 9일)은 독일계 미국 한된 상황에서의 의사 결정 모델에 관한 이론으로 1978년 노벨 경제학상을 수상한 미국의 심리학자/경 인지과학자다. 그는 인간 인지능력의 한계(제한적 합리성)라는 관점을 가지고 주류 경제학이 가정하는 대해 그 체계를 비판한 최초의 사회학자였다. 그가 처음 합리성에 의문을 제기한 당시에는 그의 논점 념적 단계에 머물렀고, 모델화가 어려웠기 때문에 대다수의 경제학자들에게 인정받지 못했다. 사이먼 후에 경제학과 심리학이 결합하는 행동 경제학으로 꽃을 피우게 된다. 또한 그는 디지털 컴퓨터는 단 조작 기계라기보다 '범용 목적의 상징(기호)조작체계'(general purpose symbol manipulation system) 계로 간주할 수 있다고 주장하였다

In [7]:

```
Image(filename='isimon1.jpeg')
```

Out[7]:



The Psychology of Thinking: Embedding Artifice in Nature

We watch an ant make his laborious way across a wind- and wave-molded beach. He moves ahead, angles to the right to ease his climb up a steep dunelet, detours around a pebble, stops for a moment to exchange information with a compatriot. Thus he makes his weaving, halting way back to his home. So as not to anthropomorphize about his purposes, I sketch the path on a piece of paper. It is a sequence of irregular, angular segments—not quite a random walk, for it has an underlying sense of direction, of aiming toward a goal.

.....

Viewed as a geometric figure, the ant's path is irregular, complex, hard to describe. But its complexity is really a complexity in the surface of the beach, not a complexity in the ant. An ant, viewed as a behaving system, is quite simple. The apparent complexity of its behavior over time is largely a reflection of the complexity of the environment in which it finds itself.

.....

At the level of cells or molecules ants are demonstrably complex, ***but these microscopic details of the inner environment may be largely irrelevant to the ant's behavior in relation to the outer environment. That is why an automaton, though completely different at the microscopic level, might nevertheless simulate***

the ant's gross behavior.

Simon, Herbert A.. The Sciences of the Artificial (The MIT Press) (pp. 51-52). The MIT Press. Kindle Edition.

Problem Solving as Change in Representation

That representation makes a difference is a long-familiar point. We all believe that arithmetic has become easier since Arabic numerals and place notation replaced Roman numerals, although I know of no theoretic treatment that explains why.

That representation makes a difference is evident for a different reason. All mathematics exhibits in its conclusions only what is already implicit in its premises, as I mentioned in a previous chapter. Hence all mathematical derivation can be viewed simply as change in representation, making evident what was previously true but obscure.

This view can be extended to all of problem solving—solving a problem simply means representing it so as to make the solution transparent. If the problem solving could actually be organized in these terms, the issue of representation would indeed become central. But even if it cannot— if this is too exaggerated a view—a deeper understanding of how representations are created and how they contribute to the solution of problems will become an essential component in the future theory of design.

Simon, Herbert A.. The Sciences of the Artificial (The MIT Press) (p. 132). The MIT Press. Kindle Edition.

갈릴레오 갈릴레이의 격언 2개:
Discourses and Mathematical Demonstrations Relating to Two New Sciences(1638년)

-One, description first, explanation second – that is, the "how" precedes the "why";

-Two, description is carried out in the language of mathematics; namely, equations.

-Do not attempt to answer such questions in the qualitative and slippery nuances of human language; say it in the form of mathematical equations

In [8]:

```
Image(filename='idiscourses.jpeg')
```

Out[8]:

통계학을 떠받치는 일곱기둥 이야기

1. 자료집계(Aggregation) : 새로운 정보를 얻기 위해 개별성에서 오는 정보를 버린다.

2. 정보측정(Information Measurement) : 정보의 총합은 관찰 수의 제곱근에 비례한다.

3. 가능도(Likelihood) : 자료를 이용한 추론에 확률의 잣대를 적용한다.

4. 상호비교(Intercomparison) : 외부 표준에 준거하지 않고 자료 내부 측면에서 비교하여 지식을 얻는다.

5. 회귀(Regression) : 인과관계 추론의 도구

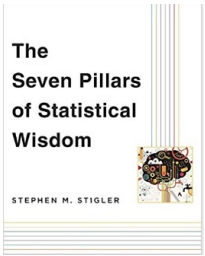
6. 설계(Design) : 랜덤화에 기반한 실험설계

7. 잔차(Residual) : 모형간 비교와 고차원 탐색의 출발점

In [9]:

```
Image(filename='istigler.jpeg', width=100, height=100)
```

Out[9]:



3. 텍스트를 수치화된 벡터로 나타내기

직관적 이해를 위해

One-hot encoding(vector)

- I = [1,0,0]
- play = [0,1,0]
- tennis = [0,0,1]

Bag of Words : term existance

	I	play	tennis
"I play tennis."	1	1	1
"I play."	1	1	0
"I, I play."	1	1	0

Bag of Words : term frequency

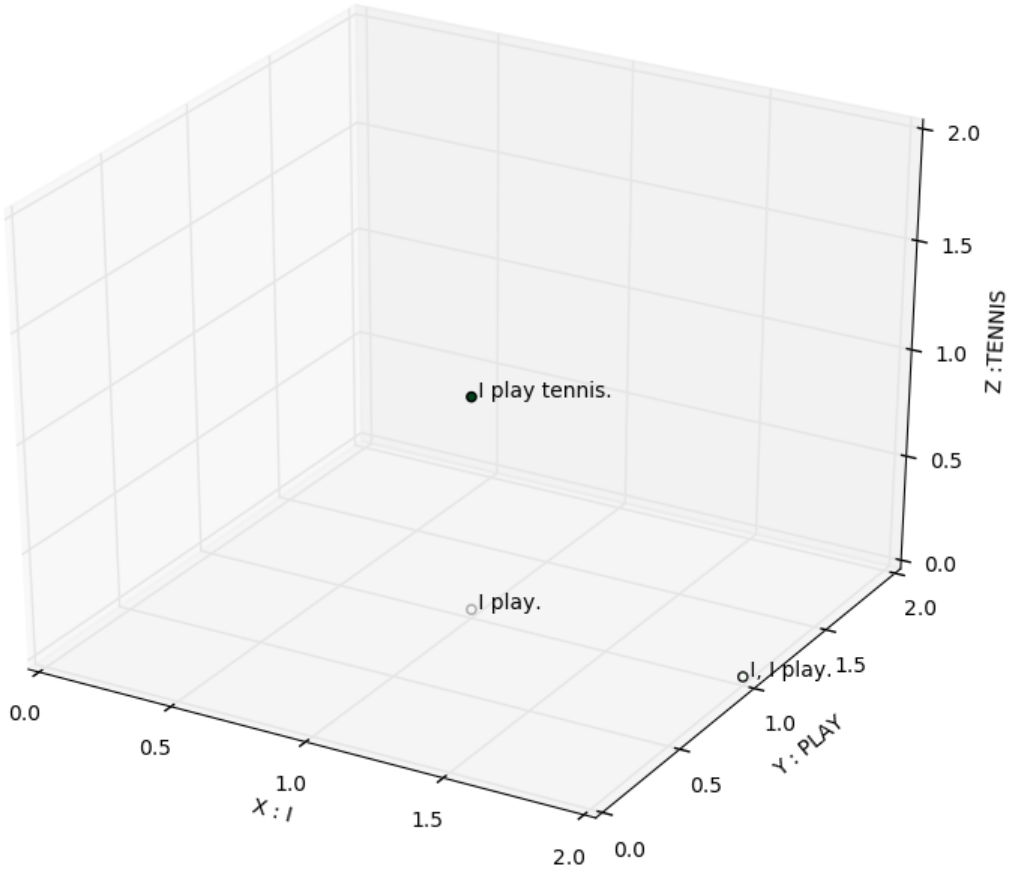
	I	play	tennis
"I play tennis."	1	1	1
"I play."	1	1	0
"I, I play."	2	1	0

In [10]:

```
%matplotlib inline
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
import numpy as np

I_play_tennis = [1,1,1]
I_play        = [1,1,0]
I_I_play      = [2,1,0]
doc_array = np.transpose(np.array([I_play_tennis, I_play, I_I_play]))

def draw():
    plt.rcParams['figure.figsize'] = [10, 8]
    ax = plt.axes(projection='3d')

    ax.set_xlabel('X : I')
    ax.set_ylabel('Y : PLAY')
    ax.set_zlabel('Z :TENNIS');
    ax.set_xlim(0, 2); ax.set_ylim(0, 2); ax.set_zlim(0, 2)

    xdata = doc_array[0]
    ydata = doc_array[1]
    zdata = doc_array[2]

    ax.text(1,1,1, " I play tennis.", fontsize=10)
    ax.text(1,1,0, " I play.", fontsize=10)
    ax.text(2,1,0, " I, I play.", fontsize=10)

    ax.scatter3D(xdata, ydata, zdata, c=zdata, cmap='Greens');
```

In [11]:

```
draw()
```



4. 용어 정리

- 단어(Word) : 분리하여 자립적으로 쓸수 있는 말이나 이에 준하는 말

- 문장(sentence) : 생각이나 감정을 말과 글로 표현할 때 완결된 내용을 나타내는 최소의 단위

- 문단(paragraph) : 단락

- 문서(document) : 문장 혹은 문단으로 구성된 글

- 토큰(token) : 의미를 가질 수 있는 부분(단어: N-gram, 문장 등)

- 품사(pos : part of speech) : 단어를 기능, 형태에 따라 나눈 갈래(명사, 대명사, 수사, 조사, 형용사, 관형사, 부사 , 감탄사)

- 형태소(morpheme) : 뜻을 가진 가장 작은 말의 단위

5. Lexicosemantics : word meanings and their relationships

1. Lemmas and Sense : 표제어와 그 의미

mouse (N) -- mice
1) any of numerous small rodents... 2) a hand-operated device that controls a cursor

Issue => 하나의 단어가 다른 의미를 가진다. 또한 단수/복수형, 동사원형/부정사 등 그 표현이 다르다.

2. Relationships between words or sense

1) synonyms(동의어)

- couch / sofa, car / automobile

2) principle of contrast(단어의 모양이 다르면 의미도 다르다.)

- water / H_2O

3) antonyms(반의어)

- long / short, big / little, hot / cold

Issue => 반의어/동의어는 단어가 실제 사용된 의미에 따라 달라질 수 가 있어, 기계적으로 나누기 어렵다.

3. Word Similarity : 동의어는 아니나 비슷함이 있는 단어

- cat VS. dog / computer VS. mouse / dark VS. heavy / light VS. heavy

ex) SimLex-999 dataset

- vanish VS. disappear : 9.8
- muscle VS. bone : 3.65
- hoel VS. agreement : 0.3

Issue => 하나 하나 언어 의미의 전문가가 비슷함의 정도를 부여 해야하고, 신조어 등에 대응하기 어렵다.

4. Word Relatedness(association) : 비슷하지는 않지만 연상이 되고 관계가 있는 단어

- cup VS. coffee / scalpel VS. surgeon : 커피를 마실 때 컵을 쓰고, 수술을 할때 해부용 칼을 쓴다.

Simantic field

- hospitals : surgeon, scalpel, nurse, doctor
- restaurants : water, menu, plate, food, chef
- house : door, roof, kitchen, family, bed

5. Semantic Frame and Roles : 특정한 상황 속에서 사용되는 단어들

- Commercial transaction : Buy and Sell

  "Sam bought the book from Ling."
  = "Ling sold the book to Sam."

6. Taxonomic Relations : 상하위 포함관계

- vehicle VS. car / fruit VS. mango / furniture VS. chair / mammal VS. dog

7. Connotation : affective meaning, 정서적 의미

- 단어의 정서적 의미를 몇가지로 나누어 값을 부여함

Valence : 즐거움의 정도 , Arousal : 자극의 정도 , Dominance : 통제할 수 있는 수준

	Valence	Arousal	Dominance
courageous	8.05	5.5	7.38
music	7.67	5.57	6.5
life	6.68	5.59	5.58
heartbreak	2.45	5.65	3.58
cub	6.71	3.95	4.24

"자언어 어렵다. 컴퓨터가 다룰 수 있는 다른 방법이 없을까?"

5. Vector Semantics

"*the meaning of a word is its use in the language - Wittgenstein*"

완벽하지는 않지만 현재까지는 최선

- 분포적 가정(distributinal hypothesis)에 기반

유사한 의미를 가진 단어는 같은 맥락에서 등장한다.

- Vector representing(embedding)을 목표

비슷한 의미를 갖는 단어/문장/문서는 벡터 공간상 가깝도록, 상이한 의미를 갖는 것들은 벡터 공간상 멀리 떨어지도록 표현한다.

- Co-occurrence matrix를 활용

단어와 단어가 맥락 속에서 얼마나 함께 등장하게 되는지 표현

5.1 Cosine similarity and Pearson correlation

1) Cosine similarity

- 벡터의 내적(inner product or dot product)

$$inner(v, w) : \langle v, w \rangle : \vec{v} \cdot \vec{w} = \sum_{i=1}^{N} v_i w_i = v_1 w_1 + v_2 w_2 + v_3 w_3 + \ldots + v_N w_N$$

일반적인 자연어 처리 영역에서 $v_i, w_i \geq 0$

*** 두개의 벡터의 내적 값이 크다는 것은 같은 차원에서 함께 큰 값을 갖고 있다는 것, 반대로 내적 값이 작다는 것은 다른 차원에 값을 갖고 있다는 것. 예를 들면 직교하는 두개의 벡터의 내적은 "0" 임.~> one hot encoding으로 나타낸 단어들 간의 내적 값은 모두 0**

*** 서로 다른 내적값 끼리 그 크기를 비교하기 위해서는 vector length로 normalize가 필요함.**

- vector lenth of $\vec{v}$: L_2 norms

$$||\vec{v}|| = \sqrt{\sum_{i=1}^{N} v_i^2}$$

- normalize된 내적값은 두 벡터간 각도의 코사인 값

$$\frac{\vec{v} \cdot \vec{w}}{||\vec{v}|| \cdot ||\vec{w}||} = \cos\theta$$

- Cosine similarity

$$CosSim(\vec{v}, \vec{w}) = \frac{\vec{v} \cdot \vec{w}}{||\vec{v}|| \cdot ||\vec{w}||} = \frac{\sum_{i=1}^{N} v_i w_i}{\sqrt{\sum_{i=1}^{N} v_i^2} \sqrt{\sum_{i=1}^{N} w_i^2}}$$

2) Peaerson correlation

- for a population

$$\rho_{X,Y} = \frac{\operatorname{cov}(X, Y)}{\sigma_X \sigma_Y}$$

where:

- cov is the covariance
- σ_X is the standard deviation of X
- σ_Y is the standard deviation of Y

$$\operatorname{cov}(X, Y) = \operatorname{E}[(X - \mu_X)(Y - \mu_Y)]$$

the formula for ρ can also be written as

$$\rho_{X,Y} = \frac{\operatorname{E}[(X - \mu_X)(Y - \mu_Y)]}{\sigma_X \sigma_Y}$$

The formula for ρ can be expressed in terms of uncentered moments. Since

- $\mu_X = \operatorname{E}[X]$

- $\mu_Y = \mathrm{E}[Y]$
- $\sigma_X^2 = \mathrm{E}[(X - \mathrm{E}[X])^2] = \mathrm{E}[X^2] - [\mathrm{E}[X]]^2$
- $\sigma_Y^2 = \mathrm{E}[(Y - \mathrm{E}[Y])^2] = \mathrm{E}[Y^2] - [\mathrm{E}[Y]]^2$
- $\mathrm{E}[(X - \mu_X)(Y - \mu_Y)] = \mathrm{E}[(X - \mathrm{E}[X])(Y - \mathrm{E}[Y])] = \mathrm{E}[XY] - \mathrm{E}[X]\,\mathrm{E}[Y]$

$$\rho_{X,Y} = \frac{\mathrm{E}[XY] - \mathrm{E}[X]\,\mathrm{E}[Y]}{\sqrt{\mathrm{E}[X^2] - [\mathrm{E}[X]]^2}\,\sqrt{\mathrm{E}[Y^2] - [\mathrm{E}[Y]]^2}}$$

- for a sample

$$r_{x,y} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

- n is the sample size
- x_i, y_i are the individual sample points indexed with i
- $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$ (the sample mean); and analogously for $\bar{y}$

3) Relationship between Cosine similarity & Peaerson correlation

$$z_x = \frac{x - \bar{x}}{s_x}$$

- 표준화시킨 x => z_x 의 평균은 0, 표준편차는 1

따라서 Cosine similarity는 표준화된(standardized or zero-centered) 두개의 벡터의 Peaerson correlation 이다.

참고 : Brendan T. O'Connor의 블로그 => 내적, 코사인유사도, 상관계수, OLS추정계수 간 관계
https://brenocon.com/blog/2012/03/cosine-similarity-pearson-correlation-and-ols-coefficients/
(https://brenocon.com/blog/2012/03/cosine-similarity-pearson-correlation-and-ols-coefficients/)

5.2 Term-document matrix

*** 행(row): 단어(vocabulary), 열(column): 문서(document), 원소(element): 문서별로 단어의 등장 횟수**

*** d_i 는 $|V|$-dimensional space상의 한 점으로 생각할 수 있다.**

*** 행렬의 크기 :** $|V| \cdot |D|$

- $|V|$: 단어의 갯수
- $|D|$: 문서의 갯수

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	1	8	15

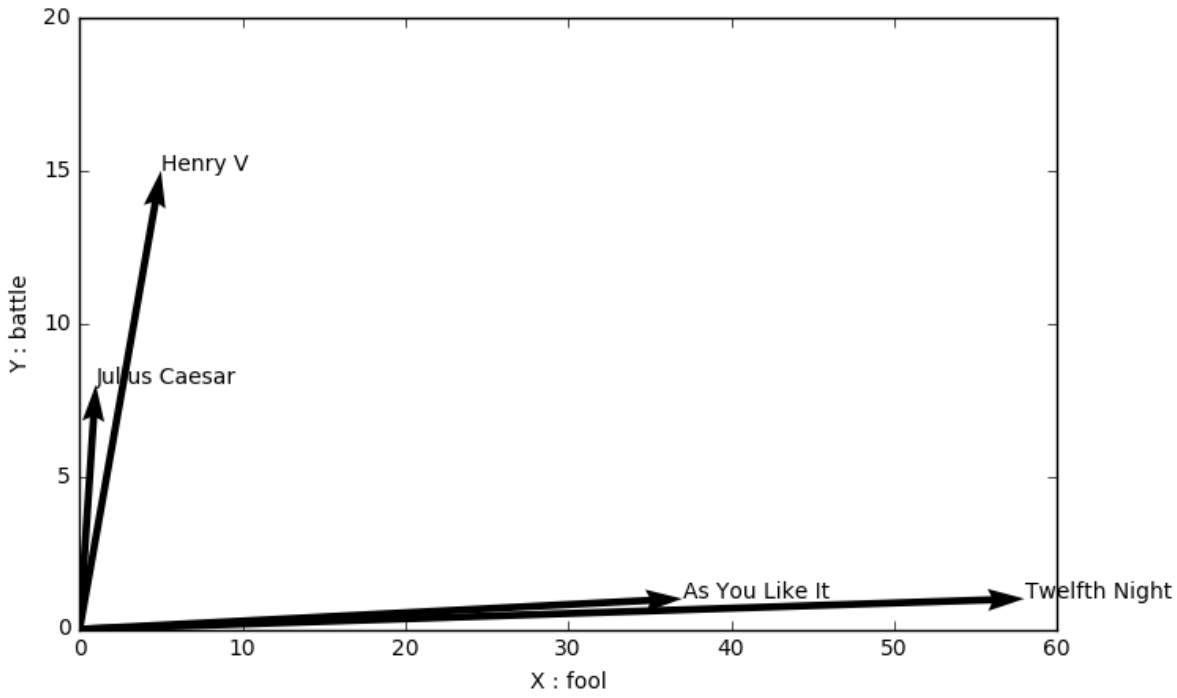
	As You Like It	Twelfth Night	Julius Caesar	Henry V
soldier	2	2	12	36
fool	37	58	1	5
clown	5	117	0	0

In [12]:

```
As_You_Like_It = [0,0,37,1]
Twelfth_Night  = [0,0,58,1]
Julius_Caesar  = [0,0,1,8]
Henry_V        = [0,0,5,15]
doc_array = np.array([As_You_Like_It, Twelfth_Night, Julius_Caesar, Henry_V])

def draw():
    plt.rcParams['figure.figsize'] = [8, 5]
    ax = plt.axes()

    ax.set_xlabel('X : fool'); ax.set_ylabel('Y : battle')
    ax.set_xlim(0, 60); ax.set_ylim(0, 20)

    X_0, Y_0, X, Y = zip(*doc_array)

    ax.text(5,15, "Henry V", fontsize=10) ; ax.text(1,8, "Julius Caesar", fontsize=1
    ax.text(37,1, "As You Like It", fontsize=10) ; ax.text(58,1, "Twelfth Night", fo

    ax = plt.gca()
    ax.quiver(X_0, Y_0, X, Y, angles='xy', scale_units='xy', scale=1)
    plt.show()
```

In [13]:

```
draw()
```



5.2 Term-Term matrix

*** 행(row): 단어(vocabulary), 열(column): 단어(vocabulary), 원소(element): 단어와 단어가 함께 등장하는 횟수**

*** 대칭행렬(symmetric matrix)**

*** 행렬의 크기 :** $|V| \cdot |V|$

- $|V|$: 단어의 갯수

	battle	soldier	fool	clown
battle	1	3	37	5
soldier	3	2	12	36
fool	37	12	1	5
clown	5	36	5	0

n-word window으로 제한하여 구성할 수도 있다.

*** 행(row): 단어(vocabulary), 열(column): n-word window에 등장하는 단어(vocabulary), 원소(element): 단어와 단어가 함께 등장하는 횟수**

*** 비대칭행렬(unsymmetric matrix)**

*** 행렬의 크기 :** $|V| \cdot |V_{window}|$

- $|V|$: 단어의 갯수

sugar, a sliced lemon, a tablespoonful of ***apricot*** jam, a pinch each of,
their enjoyment. Cautiously she sampled her first ***pineapple*** and another fruit whose taste she likened
well suited to programming on the digital ***computer.*** In finding the optimal R-stage policy from
for the purpose of gathering data and ***information*** necessary for the study authorized in the

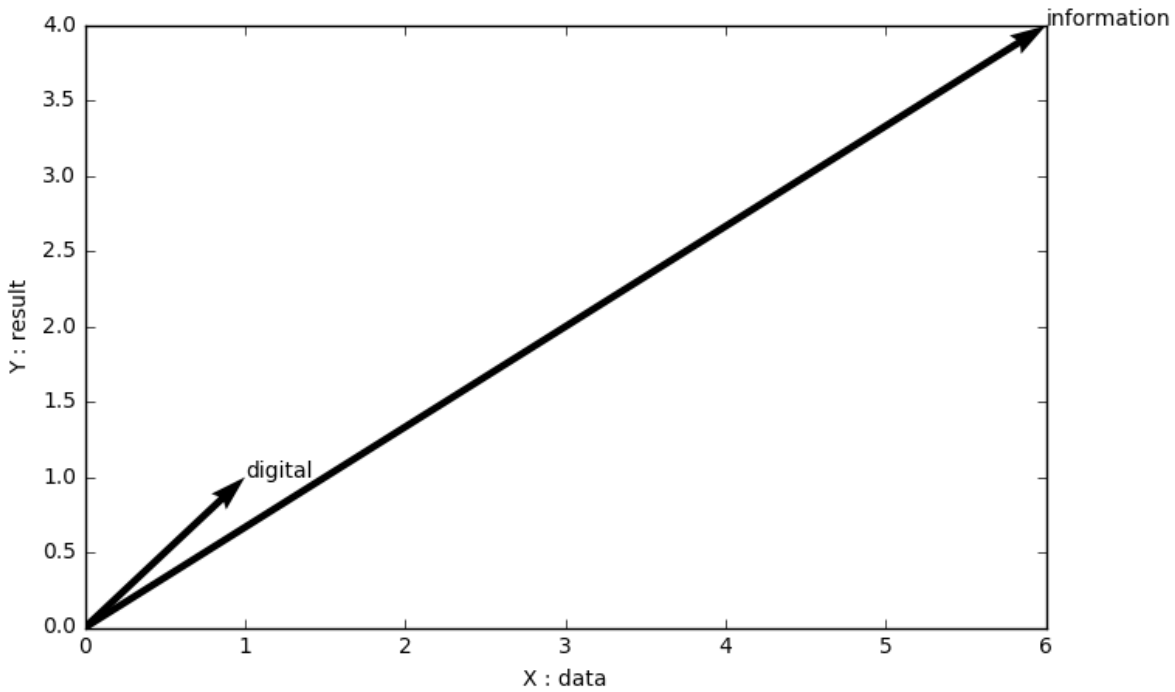
	aardvark	...	computer	data	pinch	result	sugar	...
apricot	0	...	0	0	1	0	1	
pineapple	0	...	0	0	1	0	1	
digital	0	...	2	1	0	1	0	
information	0	...	1	6	0	4	0	

In [14]:

```
digital = [0,0,1,1]
information  = [0,0,6,4]

doc_array = np.array([digital, information])

def draw():
    plt.rcParams['figure.figsize'] = [8, 5]
    ax = plt.axes()

    ax.set_xlabel('X : data'); ax.set_ylabel('Y : result')
    ax.set_xlim(0, 6); ax.set_ylim(0, 4)

    X_0, Y_0, X, Y = zip(*doc_array)

    ax.text(1,1, "digital", fontsize=10); ax.text(6,4, "information", fontsize=10)

    ax = plt.gca()
    ax.quiver(X_0, Y_0, X, Y, angles='xy', scale_units='xy', scale=1)
    plt.show()
```

In [15]:

```
draw()
```



5.3 TF-IDF

> **Term Frequency-Inverse Document Frequency :**
>
> **빈도로 나타낸 가중치를 조정해보자.**

1) motivation : the, it, they 같이 어떠한 단어들과도 함께 자주 등장하는 단어들의 가중치를 줄이고 싶다.

해결방안 1) 단어들을 선정해서(stop-words) 지워버리고 만든다.

해결방안 2) 문서들에서 공통적으로 많이 나오는 단어는 그 빈도만큼 가중치를 줄인다. 공통적으로 많이 나오는 단어는 중요도가 덜할 것이다.

$$TF_{w,d} \cdot IDF_w = log(1 + Frequency_{w,d}) \cdot log(\frac{N}{Document\,Frequency_w})$$

- $Frequency_{w,d}$; : 문서 d에서 단어 w의 갯수
- N : 전체 문서의 갯수
- $Document\,Frequency_w$: 단어 w를 포함하고 있는 문서의 갯수

2) TF-IDF의 활용

- word similarity : paraphrase, tracking changes in word meaning, 각기 다른 문서집합에서 같은 단어의 의미 발견

- documets similarity : 다차원 벡터의 평균 개념인 centroid 값을 활용하기도 함: $\cos(d_1, d_2)$

$$d_i = \frac{w_1 + w_2 + \ldots + w_k}{k}$$

- d_i : centroid document vector
- w_k : word vector
- k : 단어의 갯수

참고 : Term-document matrix, Term-Term matrix를 기반으로 문서와 단어를 표현하는 방법들은 행렬의 크기가 큰 대신에 대부분의 원소가 0인 희소행렬(sparse matrix)인 경우가 많습니다. 행렬의 크기를 줄이고 보다 추상화된 표현을 찾기 위해 SVD(singular value decomposition) 등 요인화(factorize) 하는 방법을 이용하여 행렬의 크기를 축소하고 조밀행렬(dense matrix)로 표현하기도 합니다.

5.4 Gensim을 이용한 TF-IDF 실습

1) 파일 읽기 및 전처리

In [16]:

```
import os
import glob
import codecs
import re
import time

econ_flist = glob.glob('econ_nobel/*.txt')
phys_flist = glob.glob('phys_nobel/*.txt')
print(econ_flist)
print(phys_flist)
```

```
['econ_nobel/deaton-lecture.txt', 'econ_nobel/fama-lecture.txt', 'econ
_nobel/hansen-lecture.txt', 'econ_nobel/hart-lecture.txt', 'econ_nobe
l/holmstrom-lecture.txt', 'econ_nobel/shiller-lecture.txt', 'econ_nobe
l/tirole-lecture.txt']
['phys_nobel/akasaki-lecture.txt', 'phys_nobel/amano-lecture.txt', 'ph
ys_nobel/englert-lecture.txt', 'phys_nobel/haldane-lecture.txt', 'phys
_nobel/haroche-lecture.txt', 'phys_nobel/higgs-lecture.txt', 'phys_nob
el/kajita-lecture.txt', 'phys_nobel/kosterlitz-lecture.txt', 'phys_nob
el/mcdonald-lecture.txt', 'phys_nobel/nakamura-lecture.txt', 'phys_nob
el/wineland-lecture.txt']
```

In [17]:

```
def doctolist(flst, file, category):
        doclist = [category]
        rm_el = re.compile(r'-+page\s\d+-+', re.UNICODE)
        rm_ch = re.compile(r'[^a-zA-Z|0-9]', re.UNICODE)
        rm_sp = re.compile(r'\s+', re.UNICODE)
        with codecs.open(file, 'r' , encoding='utf-8', errors='ignore') as f:
            doclist.append(file.replace(".txt",""))
            data = [f.read().lower().replace(',',' ').replace('\ufeff','').replace(
            data[0] = re.sub(rm_el,"",data[0],count=0)
            data[0] = re.sub(rm_ch," ",data[0],count=0)
            data[0] = re.sub(rm_sp," ",data[0],count=0)
            doclist.append(data[0])
        return doclist
```

In [18]:

```
econ_txt = [doctolist(econ_flist, file, 'econ_nobel') for file in econ_flist]
phys_txt = [doctolist(econ_flist, file, 'phys_nobel') for file in phys_flist]
all_txt = econ_txt + phys_txt
print(len(econ_txt))
print(len(econ_txt[0]))
print(len(phys_txt))
print(len(phys_txt[0]))
print(len(all_txt))
print(len(all_txt[0]))
print(econ_txt[0])
```

```
7
3
11
3
18
3
['econ_nobel', 'econ_nobel/deaton-lecture', 'measuring and understandi
ng behavior welfare and povertyprize lecture december 8 2015 by angus
deatonwoodrow wilson school princeton university usa 1 introductionthe
work cited by the prize committee spans many years covers areas of eco
nomics that are not always grouped together and involves many differen
t col laborators yet like the committee i believe that the work has an
underlying unity it concerns wellbeing what was once called welfare an
d uses market and survey data to measure the behavior of individuals a
nd groups and to make inferences about wellbeing often little more tha
n counting is involved as in the estimation of the fraction of the pop
ulation whose spending is below a cutoff or the calculation of the fra
ction of newborn children who die before their first birthday measurem
ent even without understanding of mechanisms can be of great importanc
```

2) tokenize, pos_tag

In [19]:

```
import nltk
import pandas as pd
#nltk.download('punkt')
#nltk.download('maxent_treebank_pos_tagger')
#nltk.download('averaged_perceptron_tagger')
from gensim.models import TfidfModel
from gensim.corpora import Dictionary
```

In [20]:

```
# 시간이 오래걸릴 것 같은 실행은 실행시간을 확인합시다.
start_time = time.time()
doc_pos = [(lst_i[0], lst_i[1], nltk.pos_tag(nltk.word_tokenize(lst_i[2]))) for lst_
doc_non = [(lst_i[0], lst_i[1], nltk.word_tokenize(lst_i[2])) for lst_i in all_txt]
print("--- %0.2f seconds ---" % (time.time() - start_time))
print(len(doc_pos))
print(len(doc_non))
```

```
--- 13.38 seconds ---
18
18
```

In [21]:

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

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

3) TF-IDF 계산

In [22]:

```
tok_list = [(raw[2]) for raw in doc_non]
dct = Dictionary(tok_list)
corpus_as_bow = [dct.doc2bow(x) for x in tok_list]
model = TfidfModel(corpus_as_bow)
#Dictionary(tok_list).keys()
#Dictionary(tok_list).token2id
#dct.keys()
#dct.token2id
model_trained_list = [model[lst] for lst in corpus_as_bow]
```

4) OUTPUT을 위한 정리

In [23]:

```
tf_idf_pd  = pd.DataFrame(data={'doc_seq':[], 'word_id':[] , 'tf_idf':[], })
for i in range(len(model_trained_list)):
    tmp_pd = pd.DataFrame(model_trained_list[i], columns = ['word_id','tf_idf'])
    tmp_pd['doc_seq'] = i
    tf_idf_pd = tf_idf_pd.append(tmp_pd)
tf_idf_pd['word_id'] = tf_idf_pd['word_id'].astype('int')
```

In [24]:

```
token2id_pd = pd.DataFrame.from_dict(dct.token2id, orient='index')
# 컬럼이름 바꾸기
token2id_pd.rename(columns={0: 'word_id'}, inplace=True)
# index의 내용을 컬럼으로 가져가기
token2id_pd['word'] = token2id_pd.index
# index reset하기
token2id_pd.reset_index(level=0, inplace=True)
# columns 삭제, drop 하기
token2id_pd.drop(['index'], axis = 1, inplace=True)
# columns 타입 바꾸기
token2id_pd['word_id'] = token2id_pd['word_id'].astype('int')
```

In [25]:

```
tf_idf_pd_id = pd.merge(tf_idf_pd, token2id_pd, how='left', on = 'word_id')
# 컬럼 순서 바꾸기
tf_idf_pd_id = tf_idf_pd_id[['doc_seq','word_id','word','tf_idf']]
```

In [26]:

```
doc_label = [(lst_i[0], lst_i[1]) for lst_i in all_txt]
doc_label_pd = pd.DataFrame(doc_label, columns = ['doc_field','doc_author'])
doc_label_pd['doc_seq'] = doc_label_pd.index
# 컬럼 순서 바꾸기
doc_label_pd = doc_label_pd[['doc_seq','doc_field','doc_author']]
tf_idf_pd_id= pd.merge(doc_label_pd, tf_idf_pd_id, how='left', on = 'doc_seq')
```

In [27]:

```
tf_idf_pd_id[-10:]
```

Out[27]:

	doc_seq	doc_field	doc_author	word_id	word	tf_idf
37779	17	phys_nobel	phys_nobel/wineland-lecture	15945	wesenberg	0.006438
37780	17	phys_nobel	phys_nobel/wineland-lecture	15946	wieman	0.006438
37781	17	phys_nobel	phys_nobel/wineland-lecture	15947	winelandnational	0.006438
37782	17	phys_nobel	phys_nobel/wineland-lecture	15948	wings	0.006438
37783	17	phys_nobel	phys_nobel/wineland-lecture	15949	withp	0.006438
37784	17	phys_nobel	phys_nobel/wineland-lecture	15950	withspread	0.006438
37785	17	phys_nobel	phys_nobel/wineland-lecture	15951	wolfgang	0.006438
37786	17	phys_nobel	phys_nobel/wineland-lecture	15952	younge	0.006438
37787	17	phys_nobel	phys_nobel/wineland-lecture	15953	zis	0.006438
37788	17	phys_nobel	phys_nobel/wineland-lecture	15954	zitterbewegung	0.006438

5) 결과 파일로 쓰기

In [28]:

```
excel_filename = "nobel_tfidf.xlsx"
with pd.ExcelWriter(excel_filename, engine='xlsxwriter') as writer:
    tf_idf_pd_id.to_excel(writer, header = True, encoding='utf-8')
```

중간과제

1) 노벨 경제학 / 물리학 상 수상자의 수상기념 연설이 담긴 파일(econ_flist, phys_flist)을 가지고 TF-IDF를 구해봅니다.

2) 구해진 TF-IDF값의 특징들을 요약해서 기술합니다.

3) 파악한 TF-IDF값의 특징들에 근거하여 경제학분야와 물리학 분야를 잘 구분할 수 있을지 판단해보고 그 근거를 기술합니다.

4) 경제학 분야내에 author별 특징을 잘 구분할수 있을지 판단해 보고 그 근거를 기술합니다.

***과제 취지 :**

***1)** 아직 머신러닝 알고리즘을 배우기 전 입니다. 하지만 알고리즘의 성능은 많은 부분 **input** 데이터가 결정합니다. **input** 데이터를 보고 어떤 알고리즘을 적용하면 좋을지, 좋은 결과가 나올만한 작업인지 판단할 수 있는 능력 또한 중요합니다. 단순히 **input** 데이터는 주어진 것으로 생각하고 알고리즘의 결과만 기다리면 우연히 결과가 좋기를 바라는 것과 다를 바가 없습니다. 추후에 알고리즘의 성능과 분석가의 기대를 비교해 나가면서 분석을 발전시킬 수 있을 것입니다.*

***2)** 텍스트 파일의 용량 문제 등으로 학생들이 가진 **pc**로 구동하는데 다소 버거울 수 있습니다. 이 때 데이터 규모를 작게 만들어 코딩하고, 성능이 좋은 **pc**에서 최소한의 비용으로 짧은 시간 안에 작업을 끝내보는 노력을 해봅시다.*

In [ ]: