코세라 4번째 강의

Date	
Property	
Weeks	Coursera 4번째 강의

1주차-Consumption

<Theory>

Welcome Video

4가지의 alternative data를 배울것임

1. consumption data

사람들이 무슨 품목을 사는지, 어떤 브랜드에 관심도가 높은지에 대한 데이터 → 주식시장의 매매 결정 과정을 얻을 수 있음

2. open web

인터넷에서 어떻게 경제가 흘러가는지 추출한 데이터로 → 다양한 firm과 sector에 대한 통찰력을 얻을 수 있음.

3. corporate and regulatory disclosures

기업의 재무제표와 규제 당국에 대한 파일 데이터 → 기업의 제정적 상태에 대한 정보를 얻고 다른 기업과의 관계를 얻음

4. media

언론, 디지털 미디어, 사회적 관계망 데이터

What is Consumption data?

- 왜 trader, portfolio managers, researchers 에게 consumption data가 중요한지?

consumption data는 소비자의 행동 방식을 이해하도록 함(어디서 사는지 얼마나 자주 사는지.. 무엇을 사는지...) → geolocation data(지리적 위치 데이터)

이를 통해 경제가 어떻게 돌아가는지 알수 있고

더 나아가 trading과 포트폴리오 관리에 있어 이러한 정보를 사용할 수 있음

- 어떻게 consumption을 측정할 것인가?

historical perspective > 얼마나 많은 사람이 실제로 가게에 가는지.

최근 > 카드 사용내역

ex) 대기업이 제공하는 카드 log는 지리적 단위에서 정보를 통합하는 경향이 있음. → 표본이 크면 세밀함이 떨어질 수 있기도 함

→ 대기업(대표본)과 소기업(소표본) 의 균형을 생각해야함.

- Consumption data의 특성

1. 얻기 어려움 (사생활의 문제, 비용의 문제)
2. 통합하기는 쉬움 (한번 얻고 나면 어떻게 사용할지가 직관적임)

Geolocation and foot-traffic

- Geolocation → 지리 위치 정보

지구 표면 두 지점 사이의 거리를 계산하고 이를 적용하여 food traffic 이 어디에 있는지, 사람들이 어디에 있는지, 그리고 그것들이 특정 관심 지점(Points of interest → POIs)으로부터 얼마나 떨어져 있는지를 이해할 수 있음.

이를 바탕으로 유동인구와 POIs 주변의 traffic 분포를 알 수 있음.

위치 데이터를 활용하려면 지구의 표면에 정확한 위치를 명시할 수 있어야 함. → 이를 위해 위도와 경도를 지정하는 좌표계를 사용

<위도>

위도는 우리가 남에서 북까지의 좌표에 얼마나 떨어져 있는지를 말해줌. 남에서 북까지의 좌표에 얼마나 떨어져 있는지를 말해준다

중간에 있다면(적도) 0.

북극에 있다면, 90도

남극에 있다면 영하 90도.

적도와 북극 사이에 있다면, 45도.

<경도>

우리가 Greenwich에서 (0에서)얼마나 멀리 떨어져 있는지를 말해준다. 동쪽은 180도, 서쪽은 180도

→위도, 경도 좌표로 우리는 실제로 정확한 위치를 지정할 수 있다.

- 두 점 사이의 거리 측정

<유클리드 거리>

원형인 지구에서는 적절치 않음

→거대한 원을 따라 거리를 측정해야하고, 아크사인 함수를 이용하는 해버신 함수로 측정할 것이다.

- 거리 측정해서 뭐할건데?

어디에서 Drop off가 일어나고, 사람들의 흐름, 관심 장소 주변의 음식 traffic에 대해 알 수 있음.

시간 차원과 지리적 차원을 따라 데이터의 분포를 특성화가능

Python시각화 도구도 사용가능.

2주차

Open web

royalty-free

websites/references/government

ex) 고용동향, 기업 간 연결, 산업 간 연결 등→useful insight

Useful insight

-identify the sources

-extract

-textually analyze(비정형 데이터)

-feed into financial market application

Textual analysis

-document classification: 소비자 리뷰 분류 및 개선

-서치엔진(PageRank)

-금융쪽: risk and uncertainty 측정

-terminology: ex) 의료, 법조

Vectorize

벡터화 과정

Aa 텍스트 수집	표준화	토큰화	스테밍(stemming)	차원축소	벡터화
웹 스크래핑, API, 내부 데이터	HTML 태그	문장 분리	어간추출, 형태소 분석	불용어 제거, 다출현 단어	bags of words
html, data warehouse		I, think, to, myself, what, a, wonderful,world	reduction, reductions, reduce, reduced → reduc	and, the, it, they	

```
<html lang="ko" data-dark="false" data-useragent="Mozilla/5
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/83.0.4103.116 S
  ▶<head>…</head>
••• ▼<body> == $0
    ▼<div id="u_skip">
      ▼<a href="#newsstand">
          <span>뉴스스탠드 바로가기</span>
        </a>
      ▼<a href="#themecast">
          <span>주제별캐스트 바로가기</span>
        </a>
      ▼<a href="#timesquare">
          <span>타임스퀘어 바로가기</span>
        </a>
      ▶<a href="#shopcast">…</a>
      ▶<a href="#account">…</a>
    </div>
    ▼<div id="wrap">
      ▶<style type="text/css">…</style>
      ▶<div id="NM_TOP_BANNER" data-clk-prefix="top" class="_
```

차원축소

-local dictionary: 사전에 나와있는 용어로 한정(rule based)

-token reduction: stemming, synonyms

Bags of Words

	12	bedroom	big	brown	dog	house	likes	number	play	small	street
0	0.0	0.0	1.0	1.0	0.0	1.0	0.0	1.0	0.0	0.0	1.0
1	1.0	1.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	1.0	1.0
2	0.0	0.0	0.0	1.0	1.0	0.0	1.0	0.0	1.0	0.0	0.0
3	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0

Why might someone want to exclude numbers from the bag of words approach?

-No significance

BOW → TFIDF

TFIDF(단어 빈도-역문서 빈도)

-term frequency inverse document frequency

$$\mathbf{tf}(t, d) = \frac{f_d(t)}{\max_{w \in d} f_d(w)}$$

$$\mathbf{idf}(t, D) = \ln\left(\frac{|D|}{|\{d \in D : t \in d\}|}\right)$$

$$\mathbf{tfidf}(t, d, D) = \mathbf{tf}(t, d) \cdot \mathbf{idf}(t, D)$$

$$\mathbf{tfidf}'(t, d, D) = \frac{\mathbf{idf}(t, D)}{|D|} + \mathbf{tfidf}(t, d, D)$$

$$f_d(t) := \text{frequency of term t in document d}$$

$$D := \text{corpus of documents}$$

TF: 특정 문서 d에서의 특정 단어 t의 등장 횟수

DF: 특정 단어 t가 등장한 문서의 수

IDF: DF(t)에 반비례

$idf(d,t) = log(\frac{n}{1+df(t)})$

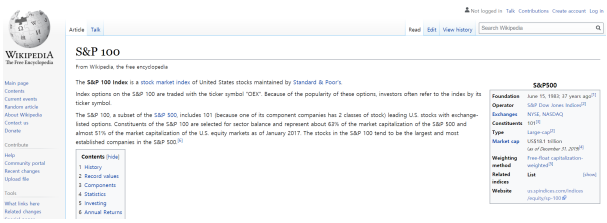
If we are computing inverse document frequency like we did in class, and we have 5 documents, in which a word appears 3, 1, 0, 0, 5 times respectively, what will the IDF term equal?

-ln(5/3)

How will doing a log transform on word counts transform the value of words?

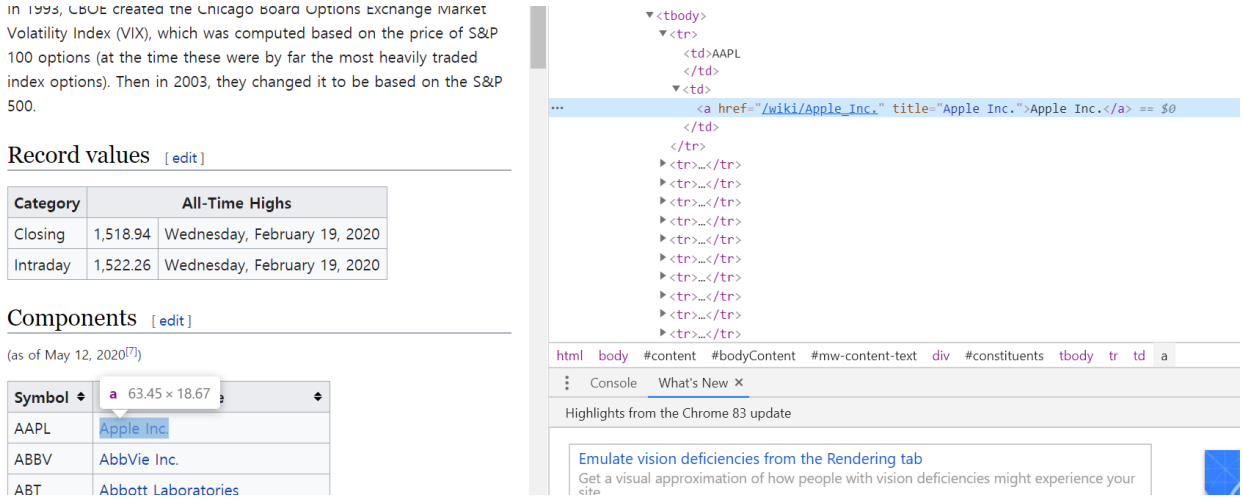
-It will reduce the degree to which more common words are important, but they will still be more important than uncommon words

Lab sessions



```
print(soup.prettify())
```

```
<li id="footer-places-contact">
 <a href="//en.wikipedia.org/wiki/Wikipedia:Contact_us">
  Contact Wikipedia
 </a>
</li>
<li id="footer-places-developers">
 <a href="https://www.mediawiki.org/wiki/Special:MyLanguage/How_to_contribute">
  Developers
 </a>
</li>
<li id="footer-places-statslink">
 <a href="https://stats.wikimedia.org/#/en.wikipedia.org">
  Statistics
 </a>
</li>
<li id="footer-places-cookiestatement">
 <a href="https://foundation.wikimedia.org/wiki/Cookie_statement">
  Cookie statement
 </a>
</li>
```




```
1 apple_element = apple_text_elements[0]
2 #The parent attribute gives you the element which contains the current element
3 print(apple_element)
4 print(apple_element.parent)
```

```
Apple Inc.
<a href="/wiki/Apple_Inc." title="Apple Inc.">Apple Inc.</a>
```

```
1 #Getting the parent of the hyperlink, we are now in the <td> tag
2 print(apple_element.parent.parent)
```

```
<td><a href="/wiki/Apple_Inc." title="Apple Inc.">Apple Inc.</a>
</td>
```

In 1993, CBOE created the Chicago Board Options Exchange Market Volatility Index (VIX), which was computed based on the price of S&P 100 options (at the time these were by far the most heavily traded index options). Then in 2003, they changed it to be based on the S&P 500.

Record values [edit]

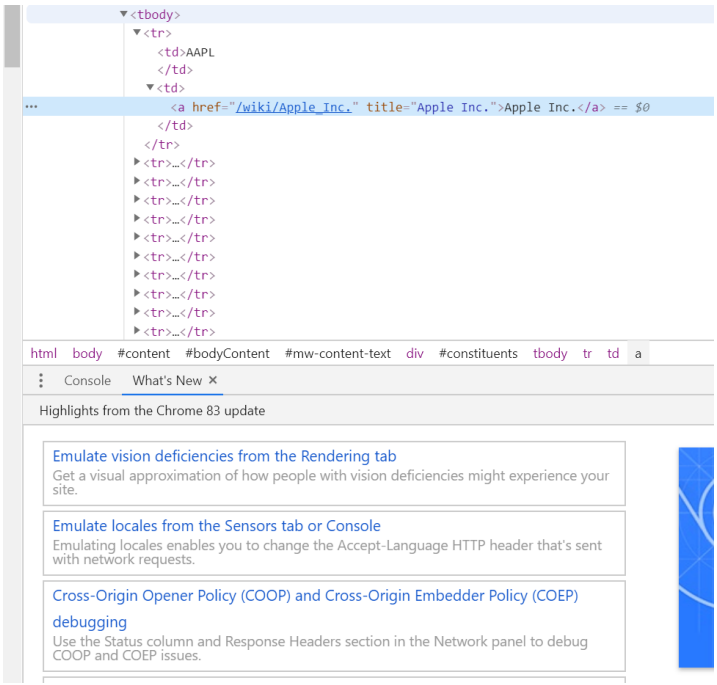
Category	All-Time Highs	
Closing	1,518.94	Wednesday, February 19, 2020
Intraday	1,522.26	Wednesday, February 19, 2020

Components [edit]

(as of May 12, 2020[7])

tbody 292.67 × 2828

	Name
AAPL	Apple Inc.
ABBV	AbbVie Inc.
ABT	Abbott Laboratories
ACN	Accenture
ADBE	Adobe Inc.
AIG	American International Group
ALL	Allstate
AMGN	Amgen Inc.
AMT	American Tower



```
#You can also query for a certain parent with a tag like below
print(apple_element.find_parent('tbody'))
```

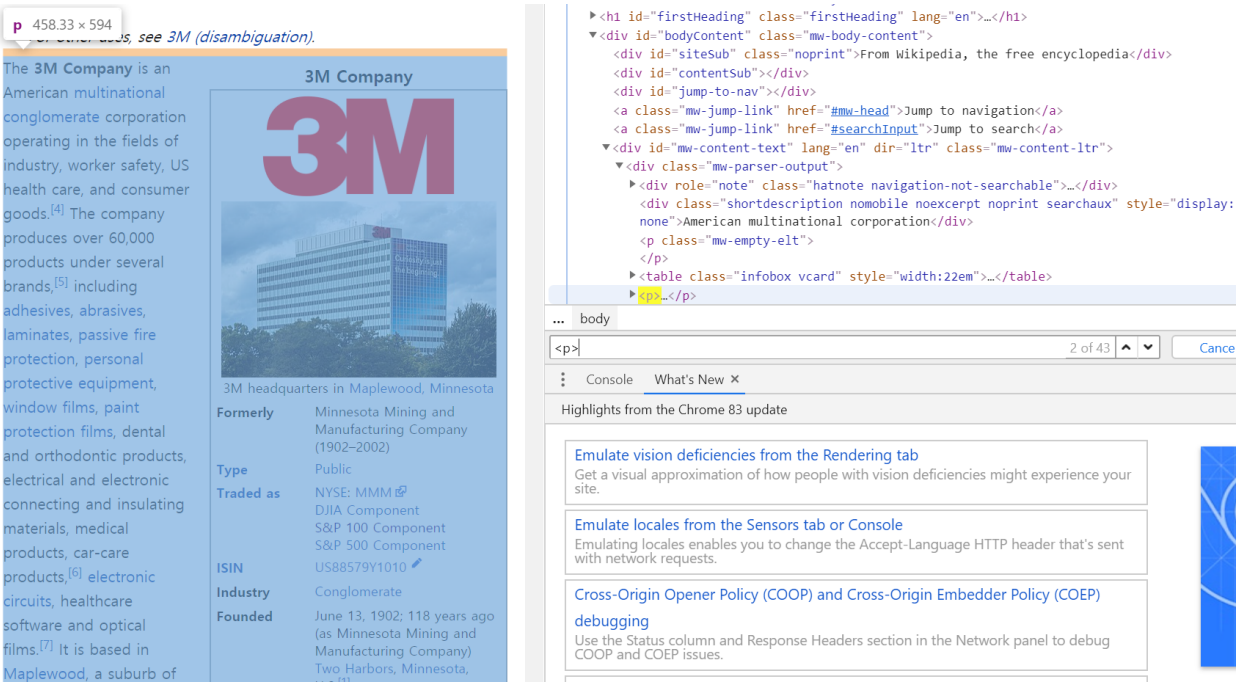
```
<tbody><tr>
<th>Symbol
</th>
<th>Name
</th></tr>
<tr>
<td>AAPL
</td>
<td><a href="/wiki/Apple_Inc." title="Apple Inc.">Apple Inc.</a>
</td></tr>
<tr>
<td>ABBV
</td>
<td><a href="/wiki/AbbVie_Inc." title="AbbVie Inc.">AbbVie Inc.</a>
</td></tr>
<tr>
<td>ABT
</td>
<td><a href="/wiki/Abbott_Laboratories" title="Abbott Laboratories">Abbott Laboratories</a>
</td></tr>
<tr>
<td>ACN
</td>
<td><a href="/wiki/Accenture" title="Accenture">Accenture</a>
</td></tr>
<tr>
<td>ADBE
```

F	Ford Motor Company
FB	Facebook, Inc.
FDX	FedEx
GD	General Dynamics
GE	General Electric
GILD	Gilead Sciences
GM	General Motors
GOOG	Alphabet Inc. (Class C)
GOOGL	Alphabet Inc. (Class A)
GS	Goldman Sachs
HD	Home Depot
HON	Honeywell
IBM	International Business Machines
INTC	Intel Corp.
JNJ	Johnson & Johnson
JPM	JPMorgan Chase & Co.
KHC	Kraft Heinz
KMI	Kinder Morgan
KO	The Coca-Cola Company
LLY	Eli Lilly and Company

GOOG(C) 의결권 X, GOOGL(A) 의결권 1

```
1 #Let's add in the part of the link that comes before each of these
2 links_unique = ["https://en.wikipedia.org/" + link for link in links_unique]
3 print(links_unique)
```

```
['https://en.wikipedia.org//wiki/3M', 'https://en.wikipedia.org//wiki/AT%26T', 'ht
```


p 458.33 × 594
see 3M (disambiguation).
The 3M Company is an American multinational conglomerate corporation operating in the fields of industry, worker safety, US health care, and consumer goods.[4] The company produces over 60,000 products under several brands,[5] including adhesives, abrasives, laminates, passive fire protection, personal protective equipment, window films, paint protection films, dental and orthodontic products, electrical and electronic connecting and insulating materials, medical products, car-care products,[6] electronic circuits, healthcare software and optical films.[7] It is based in Maplewood, a suburb of
3M Company
3M
3M headquarters in Maplewood, Minnesota
Formerly
Minnesota Mining and Manufacturing Company (1902–2002)
Type
Public
Traded as
NYSE: MMM
DJIA Component
S&P 100 Component
S&P 500 Component
ISIN
US88579Y1010
Industry
Conglomerate
Founded
June 13, 1902; 118 years ago (as Minnesota Mining and Manufacturing Company)
Two Harbors, Minnesota,
<h1 id="firstHeading" class="firstHeading" lang="en">…</h1>

From Wikipedia, the free encyclopedia

Jump to navigation
Jump to search

…
American multinational corporation

<table class="infobox vcard" style="width:22em">…</table>
…
body

2 of 43
Cance
Console
What's New
Highlights from the Chrome 83 update
Emulate vision deficiencies from the Rendering tab
Get a visual approximation of how people with vision deficiencies might experience your site.
Emulate locales from the Sensors tab or Console
Emulating locales enables you to change the Accept-Language HTTP header that's sent with network requests.
Cross-Origin Opener Policy (COOP) and Cross-Origin Embedder Policy (COEP) debugging
Use the Status column and Response Headers section in the Network panel to debug COOP and COEP issues.

```
def get_company_text(url):
    r = requests.get(url)
    soup = BeautifulSoup(r.content, 'html.parser')
    paragraphs = soup.find_all("p")
    paragraphs = [paragraph.text for paragraph in paragraphs]
    paragraphs = " ".join(paragraphs)
    pattern = "₩[[0-9]+₩]"
    paragraphs = re.sub(pattern, "", paragraphs)
    stemmer = nltk.stem.SnowballStemmer('english')
    paragraphs = stemmer.stem(paragraphs)
    vectorizer = CountVectorizer(stop_words='english')
    counts = vectorizer.fit_transform([paragraphs])
    counts = pd.Series(counts.toarray()[0],index=vectorizer.get_feature_names())
    return counts
counts = get_company_text(links_unique[0])
print(counts.sort_values(ascending=False))
```

Loading...

```
3m              62
company         30
billion         11
business        11
products        11
minnesota       10
million         10
sales            9
announced        9
2018             7
manufacturing    7
year             7
division         6
product          6
audio            6
sold             6
digital          6
```

	3M	AT%26T	AbbVie_Inc.	Abbott_Laboratories	Accenture	#
00	0	0	0	0	0	
000	4	5	1	1	5	
000005	0	0	0	0	0	
000791	0	0	0	0	0	
000th	0	0	0	0	0	
001	0	0	0	0	0	
002	0	0	0	0	0	
003	0	0	0	0	0	
005	0	0	0	0	0	
006	0	0	0	0	0	
007	0	0	0	0	0	
0090	0	0	0	0	0	
01	0	0	0	0	0	
010	0	0	0	0	0	
010976	0	0	0	0	0	
011	0	0	0	0	0	
0191	0	0	0	0	0	
02	0	0	0	0	0	
021	0	0	0	0	0	
023	0	0	0	0	0	
024	0	0	0	0	0	
025	0	0	0	0	0	
03	0	0	0	0	0	

```
print(counts.sum(axis=1).sort_values(ascending=False).head(100))
```

```
company        3583
million        1336
new            1301
billion        1279
announced      1040
bank            925
united          826
states          813
business        736
000             690
products        680
apple           671
2018            658
largest         642
year            625
american        582
corporation     574
including       573
acquired        562
market          539
2017            535
companies       530
based           519
2016            502
2015            493
world           485
stores          478
time            469
2010            468
2014            466
```

23주차

Lab session: Company Distances and Industry Distances

similarity 구하는 방법 : 거리를 구해서 비교해본다.
BNY Mellon과 JPMorgan 간 거리가 BNY Mellon 과 Facebook 거리보다 가깝다.
(비슷한 업계니까)
ex) 3M 과 다른 기업 간 text에서 찾을 수 있는 combination 개수를 찾는다.

big data → sorting 필요하다

```
#Turn it into a function
def get_company_industries(urls):
    industries_data = []
    for url in urls:
        r = requests.get(url)
        soup = BeautifulSoup(r.content, 'html.parser')
        infobox = soup.find("table", {"class": "infobox"})
        industries = [x.text for x in infobox.find("th", text = "Industry").parent()[1].find_all('a')]
        industries_data.append(industries)
    return industries_data
print(get_company_industries(links_unique[:5]))

#Instead of an array, let's modify to get a dataframe of dummy variables representing what industries each company is tagged with
def get_company_industries(urls):
    industries_data = []
    for url in urls:
        r = requests.get(url)
        soup = BeautifulSoup(r.content, 'html.parser')
```

```
        infobox = soup.find("table", {"class": "infobox"})
        industries = [x.text for x in infobox.find("th", text = "Industry").parent()[1].find_all('a')]
        industries = pd.Series(1, index=industries)
        industries_data.append(industries)
    industries_data = pd.concat(industries_data,axis=1,sort=False).fillna(0)  ## 행방향
    return industries_data
print(get_company_industries(links_unique[:5]))

#And clean up with transposing and putting in the index of tickers
def get_company_industries(urls):
    industries_data = []
    for url in urls:
        r = requests.get(url)
        soup = BeautifulSoup(r.content, 'html.parser')
        infobox = soup.find("table", {"class": "infobox"})
        industries = [x.text for x in infobox.find("th", text = "Industry").parent()[1].find_all('a')]
        industries = pd.Series(1, index=industries)
        industries_data.append(industries)
    industries_data = pd.concat(industries_data,axis=1,sort=False).fillna(0)
    return industries_data
industries = get_company_industries(links_unique)
industries = industries.transpose()
industries.index = index
print(industries)


#Let's see which companies are in financial services
fin_services = industries[industries['financial services'] == 1].index
print(fin_services)

Index(['American_Express', 'American_International_Group', 'Bank_of_America',
       'Capital_One', 'Caterpillar_Inc.', 'Citigroup', 'Goldman_Sachs',
       'JPMorgan_Chase_%26_Co.', 'MasterCard', 'MetLife', 'Morgan_Stanley',
       'PayPal', 'The_Bank_of_New_York_Mellon', 'U.S._Bancorp', 'Visa_Inc.',
       'Wells_Fargo'],
      dtype='object')
```

Application: applying similarity analysis on corporate filings to predict returns
텍스트 분석으로 주식 수익률 계산?
Q. 분기별/연도별 보고서에서 텍스트가 바뀌는 것이 실제로 회사의 변화를 의미하는가
A.

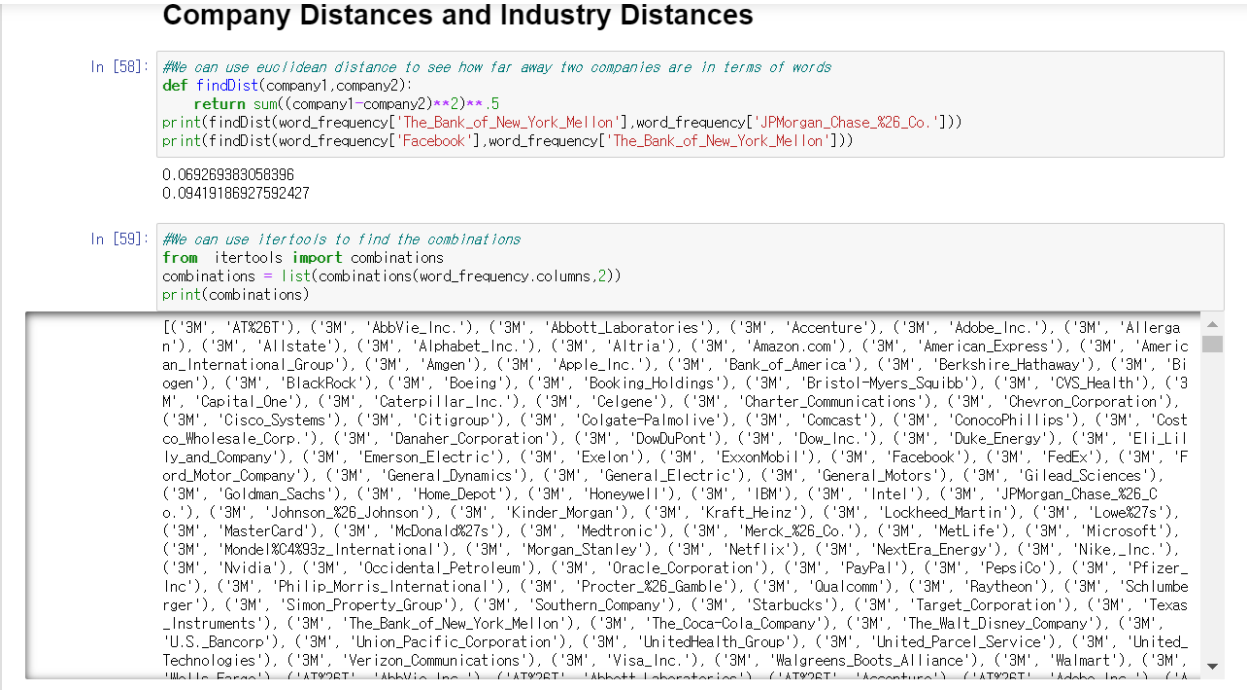
보고서들의 유사성을 확인 (유클리디안 대신 cosine similarity 이용한다.)

결론 (a paper by Cohen, Malloy, and Nguyen 참조)

올해와 전분기 사이에 전혀 변동이 없는 포트폴리오를 사서 작년과 가장 큰 변동폭을 보인 1분기에 주식을 팔면 실제로 상당한 수익률을 보인다.
보고서의 법적인 부분의 변화가 제일 큰 영향을 끼친다.

Lab session: Working with 10-K Data

Lab session: Company Distances and Industry Distances



Company Distances and Industry Distances

In [58]:
```
#We can use euclidean distance to see how far away two companies are in terms of words
def findDist(company1,company2):
    return sum((company1-company2)**2)**.5
print(findDist(word_frequency['The_Bank_of_New_York_Mellon'],word_frequency['JPMorgan_Chase_%26_Co.']))
print(findDist(word_frequency['Facebook'],word_frequency['The_Bank_of_New_York_Mellon']))
```

```
0.069269383058396
0.09419186927592427
```

In [59]:
```
#We can use itertools to find the combinations
from  itertools import combinations
combinations = list(combinations(word_frequency.columns,2))
print(combinations)
```

```
[('3M', 'AT%26T'), ('3M', 'AbbVie_Inc.'), ('3M', 'Abbott_Laboratories'), ('3M', 'Accenture'), ('3M', 'Adobe_Inc.'), ('3M', 'Allerga
n'), ('3M', 'Allstate'), ('3M', 'Alphabet_Inc.'), ('3M', 'Altria'), ('3M', 'Amazon.com'), ('3M', 'American_Express'), ('3M', 'Americ
an_International_Group'), ('3M', 'Amgen'), ('3M', 'Apple_Inc.'), ('3M', 'Bank_of_America'), ('3M', 'Berkshire_Hathaway'), ('3M', 'Bi
ogen'), ('3M', 'BlackRock'), ('3M', 'Boeing'), ('3M', 'Booking_Holdings'), ('3M', 'Bristol-Myers_Squibb'), ('3M', 'CVS_Health'), ('3
M', 'Capital_One'), ('3M', 'Caterpillar_Inc.'), ('3M', 'Celgene'), ('3M', 'Charter_Communications'), ('3M', 'Chevron_Corporation'),
('3M', 'Cisco_Systems'), ('3M', 'Citigroup'), ('3M', 'Colgate-Palmolive'), ('3M', 'Comcast'), ('3M', 'ConocoPhillips'), ('3M', 'Cost
co_Wholesale_Corp.'), ('3M', 'Danaher_Corporation'), ('3M', 'DowDuPont'), ('3M', 'Dow_Inc.'), ('3M', 'Duke_Energy'), ('3M', 'Eli_Lil
ly_and_Company'), ('3M', 'Emerson_Electric'), ('3M', 'Exelon'), ('3M', 'ExxonMobil'), ('3M', 'Facebook'), ('3M', 'FedEx'), ('3M', 'F
ord_Motor_Company'), ('3M', 'General_Dynamics'), ('3M', 'General_Electric'), ('3M', 'General_Motors'), ('3M', 'Gilead_Sciences'),
('3M', 'Goldman_Sachs'), ('3M', 'Home_Depot'), ('3M', 'Honeywell'), ('3M', 'IBM'), ('3M', 'Intel'), ('3M', 'JPMorgan_Chase_%26_C
o.'), ('3M', 'Johnson_%26_Johnson'), ('3M', 'Kinder_Morgan'), ('3M', 'Kraft_Heinz'), ('3M', 'Lockheed_Martin'), ('3M', 'Lowe%27s'),
('3M', 'MasterCard'), ('3M', 'McDonald%27s'), ('3M', 'Medtronic'), ('3M', 'Merck_%26_Co.'), ('3M', 'MetLife'), ('3M', 'Microsoft'),
('3M', 'Mondel%C4%93z_International'), ('3M', 'Morgan_Stanley'), ('3M', 'Netflix'), ('3M', 'NextEra_Energy'), ('3M', 'Nike,_Inc.'),
('3M', 'Nvidia'), ('3M', 'Occidental_Petroleum'), ('3M', 'Oracle_Corporation'), ('3M', 'PayPal'), ('3M', 'PepsiCo'), ('3M', 'Pfizer_
Inc'), ('3M', 'Philip_Morris_International'), ('3M', 'Procter_%26_Gamble'), ('3M', 'Qualcomm'), ('3M', 'Raytheon'), ('3M', 'Schlumbe
rger'), ('3M', 'Simon_Property_Group'), ('3M', 'Southern_Company'), ('3M', 'Starbucks'), ('3M', 'Target_Corporation'), ('3M', 'Texas
_Instruments'), ('3M', 'The_Bank_of_New_York_Mellon'), ('3M', 'The_Coca-Cola_Company'), ('3M', 'The_Walt_Disney_Company'), ('3M',
'U.S._Bancorp'), ('3M', 'Union_Pacific_Corporation'), ('3M', 'UnitedHealth_Group'), ('3M', 'United_Parcel_Service'), ('3M', 'United_
Technologies'), ('3M', 'Verizon_Communications'), ('3M', 'Visa_Inc.'), ('3M', 'Walgreens_Boots_Alliance'), ('3M', 'Walmart'), ('3M',
'Wells_Fargo'), ('AT%26T', 'AbbVie_Inc.'), ('AT%26T', 'Abbott_Laboratories'), ('AT%26T', 'Accenture'), ('AT%26T', 'Adobe_Inc.'), ('A
```

similarity 구하는 방법 : 거리를 구해서 비교해본다.

BNY Mellon과 JPMorgan 간 거리가 BNY Mellon 과 Facebook 거리보다 가깝다.

(비슷한 업계니까)

ex) 3M 과 다른 기업 간 text에서 찾을 수 있는 combination 개수를 찾는다.

In [60]:
```
#Create the distance dataframe
distance = pd.DataFrame(combinations)
distance.columns = ["Company 1", "Company 2"]
#Create the distance for each combination
distance["Distance"] = distance.apply(lambda x: findDist(word_frequency[x["Company 1"]], word_frequency[x["Company 2"]]), axis=1)
print(distance)
```

	Company 1	Company 2	Distance
0	3M	AT%26T	0.067785
1	3M	AbbVie_Inc.	0.086907
2	3M	Abbott_Laboratories	0.082096
3	3M	Accenture	0.099816
4	3M	Adobe_Inc.	0.098538
5	3M	Allergan	0.091710
6	3M	Allstate	0.089795
7	3M	Alphabet_Inc.	0.099336
8	3M	Altria	0.095903
9	3M	Amazon.com	0.084428
10	3M	American_Express	0.098040
11	3M	American_International_Group	0.087494
12	3M	Amgen	0.106996
13	3M	Apple_Inc.	0.084011
14	3M	Bank_of_America	0.093553
15	3M	Berkshire_Hathaway	0.079356
16	3M	Biogen	0.086669
17	3M	BlackRock	0.099466
18	3M	Boeing	0.091513
19	3M	Booking_Holdings	0.107829
20	3M	Bristol-Myers_Squibb	0.077146
21	3M	CVS_Health	0.086509
22	3M	Capital_One	0.093070
23	3M	Caterpillar_Inc.	0.075362
24	3M	Celgene	0.087456
25	3M	Charter_Communications	0.087377
26	3M	Chevron_Corporation	0.079814
27	3M	Cisco_Systems	0.089222

big data → sorting 필요하다

```
#Turn it into a function
def get_company_industries(urls):
    industries_data = []
    for url in urls:
        r = requests.get(url)
        soup = BeautifulSoup(r.content, 'html.parser')
        infobox = soup.find("table", {"class": "infobox"})
        industries = [x.text for x in infobox.find("th", text = "Industry").parent()[1].find_all('a')]
        industries_data.append(industries)
    return industries_data
print(get_company_industries(links_unique[:5]))

#Instead of an array, let's modify to get a dataframe of dummy variables representing what industries each company is tagged with
def get_company_industries(urls):
    industries_data = []
    for url in urls:
        r = requests.get(url)
        soup = BeautifulSoup(r.content, 'html.parser')
        infobox = soup.find("table", {"class": "infobox"})
        industries = [x.text for x in infobox.find("th", text = "Industry").parent()[1].find_all('a')]
        industries = pd.Series(1, index=industries)
        industries_data.append(industries)
    industries_data = pd.concat(industries_data,axis=1,sort=False).fillna(0)  ## 행방향
    return industries_data
print(get_company_industries(links_unique[:5]))


#And clean up with transposing and putting in the index of tickers
def get_company_industries(urls):
    industries_data = []
    for url in urls:
        r = requests.get(url)
        soup = BeautifulSoup(r.content, 'html.parser')
        infobox = soup.find("table", {"class": "infobox"})
        industries = [x.text for x in infobox.find("th", text = "Industry").parent()[1].find_all('a')]
        industries = pd.Series(1, index=industries)
        industries_data.append(industries)
    industries_data = pd.concat(industries_data,axis=1,sort=False).fillna(0)
    return industries_data
industries = get_company_industries(links_unique)
industries = industries.transpose()
```

```
industries.index = index
print(industries)
```

	Conglomerate	Telecommunications	Technology	₩
3M	1.0	0.0	0.0	
AT%26T	0.0	1.0	1.0	
AbbVie_Inc.	0.0	0.0	0.0	
Abbott_Laboratories	0.0	0.0	0.0	
Accenture	0.0	0.0	0.0	
Adobe_Inc.	0.0	0.0	0.0	
Allergan	0.0	0.0	0.0	
Allstate	0.0	0.0	0.0	
Alphabet_Inc.	1.0	0.0	0.0	
Altria	0.0	0.0	0.0	
Amazon.com	0.0	0.0	0.0	
American_Express	0.0	0.0	0.0	
American_International_Group	0.0	0.0	0.0	
Amgen	0.0	0.0	0.0	
Apple_Inc.	0.0	0.0	0.0	
Bank_of_America	0.0	0.0	0.0	
Berkshire_Hathaway	1.0	0.0	0.0	
Biogen	0.0	0.0	0.0	
BlackRock	0.0	0.0	0.0	

```
#Let's see which companies are in financial services
fin_services = industries[industries['financial services'] == 1].index
print(fin_services)
```

```
Index(['American_Express', 'American_International_Group', 'Bank_of_America',
       'Capital_One', 'Caterpillar_Inc.', 'Citigroup', 'Goldman_Sachs',
       'JPMorgan_Chase_%26_Co.', 'MasterCard', 'MetLife', 'Morgan_Stanley',
       'PayPal', 'The_Bank_of_New_York_Mellon', 'U.S._Bancorp', 'Visa_Inc.',
       'Wells_Fargo'],
      dtype='object')
```

In [78]:
```
#Let's check how similar companies are within and outside of the financials industry
print(distance.loc[fin_services_index]["Distance"].mean())
print(distance.loc[fin_services_index2]["Distance"].mean())
```

```
0.08921896425997317
0.099163444188783
```

In [79]:
```
#And check how different industries line up
#First create the base of the dataframe, each combination of industry
from  itertools import combinations
industry_distances = pd.DataFrame(list(combinations(industries.columns,2)))
industry_distances.columns = ["Industry 1", "Industry 2"]
print(industry_distances)
```

	Industry 1	Industry 2
0	conglomerate	telecommunications
1	conglomerate	technology
2	conglomerate	mass media
3	conglomerate	entertainment
4	conglomerate	health care
5	conglomerate	computer software
6	conglomerate	insurance
7	conglomerate	tobacco
8	conglomerate	cloud computing
9	conglomerate	artificial intelligence
10	conglomerate	consumer electronics
11	conglomerate	digital distribution
12	conglomerate	banking
13	conglomerate	financial services
14	conglomerate	biotechnology
15	conglomerate	computer hardware
16	conglomerate	semiconductors

In [86]:
```
##What about the most similar to financial technology?
print(industry_distances[industry_distances["Industry 1"] == 'financial services'].sort_values(by='Distance').dropna())
```

	Industry 1	Industry 2	Distance
123	financial services	entertainment	0.087768
312	financial services	banking	0.089098
258	financial services	artificial intelligence	0.089152
42	financial services	telecommunications	0.089627
337	financial services	consumer goods	0.090286
342	financial services	pharmaceutical	0.090317
336	financial services	oil and gas	0.090589
295	financial services	digital distribution	0.091025
97	financial services	mass media	0.091230
334	financial services	pharmaceuticals	0.091732
332	financial services	aerospace	0.092209
333	financial services	defense	0.092209
238	financial services	cloud computing	0.092336
172	financial services	computer software	0.092457
148	financial services	health care	0.092548
277	financial services	consumer electronics	0.092594
195	financial services	insurance	0.092609
340	financial services	automotive	0.092721
341	financial services	medical equipment	0.093429
331	financial services	semiconductors	0.093476
335	financial services	retail	0.093893
330	financial services	computer hardware	0.094054
344	financial services	video games	0.094163

In [90]:
```
import seaborn as sns
import matplotlib.pyplot as plt
#Plot the heatmap
sns.heatmap(pivot_data)
plt.show()
```

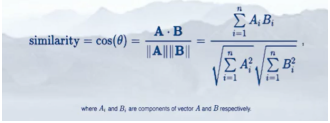
Application: applying similarity analysis on corporate filings to predict returns

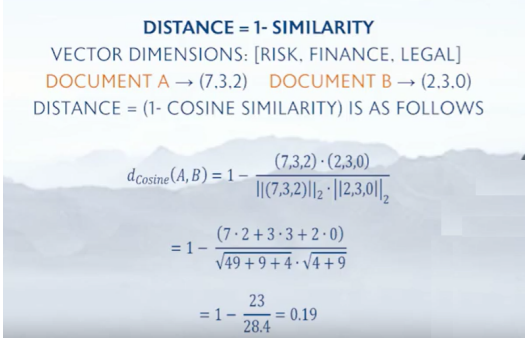
텍스트 분석으로 주식 수익률 계산?

Q. 분기별/연도별 보고서에서 텍스트가 바뀌는 것이 실제로 회사의 변화를 의미하는가

A.

1. 보고서들의 유사성을 확인 (유클리디안 대신 cosine similarity 이용한다.)





2. 결론 (a paper by Cohen, Malloy, and Nguyen 참조)

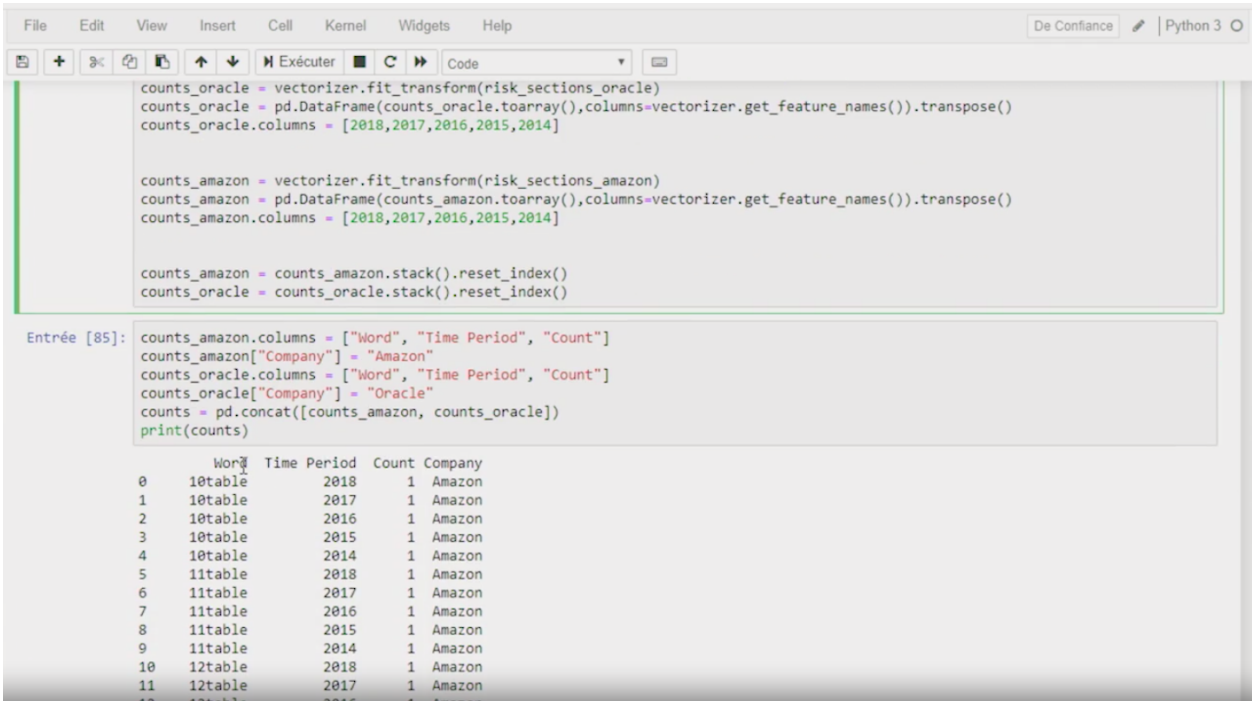
- 올해와 전분기 사이에 전혀 변동이 없는 포트폴리오를 사서 작년과 가장 큰 변동폭을 보인 1분기에 주식을 팔면 실제로 상당한 수익률을 보인다.
- 보고서의 법적인 부분의 변화가 제일 큰 영향을 끼친다.

Lab session

Risk Analysis

Amazon vs Oracle 의 tf-idf 분석

1) 단어 수 벡터화



2) 시간 - 단어 테이블 정렬

```
Entrée [87]: counts = counts.set_index(["Company", "Time Period", "Word"])["Count"].unstack().transpose().fillna(0)
             print(counts)
```

```
Company      Amazon                          Oracle
Time Period    2014 2015  2016  2017 2018    2014 2015 2016 2017 2018
Word
10table         1.0  1.0   1.0   1.0  1.0     0.0  0.0  0.0  0.0  0.0
11table         1.0  1.0   1.0   1.0  1.0     0.0  0.0  0.0  0.0  0.0
12              0.0  0.0   0.0   0.0  0.0     0.0  0.0  0.0  1.0  0.0
12table         1.0  1.0   1.0   1.0  1.0     0.0  0.0  0.0  0.0  0.0
13table         1.0  1.0   1.0   1.0  1.0     0.0  0.0  0.0  0.0  0.0
14              0.0  0.0   0.0   0.0  0.0     0.0  0.0  0.0  0.0  1.0
14table         0.0  0.0   0.0   1.0  0.0     0.0  0.0  0.0  0.0  0.0
15              0.0  0.0   0.0   0.0  0.0     0.0  0.0  0.0  0.0  1.0
16              0.0  0.0   0.0   0.0  0.0     0.0  0.0  0.0  0.0  1.0
17              0.0  0.0   0.0   0.0  0.0     0.0  0.0  0.0  0.0  2.0
18              0.0  0.0   0.0   0.0  0.0     2.0  1.0  1.0  1.0  1.0
19              0.0  0.0   0.0   0.0  0.0     1.0  1.0  0.0  0.0  1.0
1a              3.0  3.0   3.0   3.0  3.0     0.0  0.0  0.0  0.0  0.0
```

3) 단어수 로그화, tf-idf 테이블 생성

```
Entrée [88]: tf_log = np.log(counts + 1)

             n = (counts > 0).sum(axis=1)

             idf = np.log(len(counts.columns) / n)
             tf_idf = tf_log.multiply(idf, axis=0)
             print(tf_idf)
```

```
Company        Amazon                                          Oracle  \
Time Period      2014      2015      2016      2017      2018      2014
Word
10table      0.480453  0.480453  0.480453  0.480453  0.480453  0.000000
11table      0.480453  0.480453  0.480453  0.480453  0.480453  0.000000
12           0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
12table      0.480453  0.480453  0.480453  0.480453  0.480453  0.000000
13table      0.480453  0.480453  0.480453  0.480453  0.480453  0.000000
14           0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
14table      0.000000  0.000000  0.000000  1.596030  0.000000  0.000000
15           0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
16           0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
17           0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
18           0.000000  0.000000  0.000000  0.000000  0.000000  0.761500
19           0.000000  0.000000  0.000000  0.000000  0.000000  0.834530
1a           0.960906  0.960906  0.960906  0.960906  0.960906  0.000000
20           0.000000  0.000000  0.000000  0.000000  0.000000  0.834530
```

4) 코사인 유사도 구하기

```
Entrée [89]: #Let's find cosine similarity
similarity = cosine_similarity(tf_idf.transpose())
similarity = pd.DataFrame(similarity, index=tf_idf.columns,columns=tf_idf.columns)
print(similarity)
```

```
Company                Amazon                                          \
Time Period              2014      2015      2016      2017      2018
Company Time Period
Amazon  2014         1.000000  0.949288  0.909757  0.676695  0.677021
        2015         0.949288  1.000000  0.926841  0.684863  0.682113
        2016         0.909757  0.926841  1.000000  0.714120  0.697585
        2017         0.676695  0.684863  0.714120  1.000000  0.623660
        2018         0.677021  0.682113  0.697585  0.623660  1.000000
Oracle  2014         0.012158  0.012575  0.013020  0.023673  0.016233
        2015         0.004812  0.005095  0.005220  0.017151  0.012894
        2016         0.006156  0.006443  0.006662  0.018711  0.014877
        2017         0.007156  0.008189  0.008368  0.028756  0.019834
        2018         0.024158  0.019267  0.018630  0.056751  0.046559

Company                Oracle
Time Period              2014      2015      2016      2017      2018
Company Time Period
Amazon  2014         0.012158  0.004812  0.006156  0.007156  0.024158
        2015         0.012575  0.005095  0.006443  0.008189  0.019267
        2016         0.013020  0.005220  0.006662  0.008368  0.018630
        2017         0.023673  0.017151  0.018711  0.028756  0.056751
        2018         0.016233  0.012894  0.014877  0.019834  0.046559
Oracle  2014         1.000000  0.783387  0.795717  0.720088  0.531729
        2015         0.783387  1.000000  0.899076  0.796492  0.584254
        2016         0.795717  0.899076  1.000000  0.866855  0.620872
        2017         0.720088  0.796492  0.866855  1.000000  0.656020
        2018         0.531729  0.584254  0.620872  0.656020  1.000000
```

회사 내부 유사도 > 회사 외부(amazon,oracle) 유사도 → 개별 위협, 리스크에 관한 언급 x 때문

⇒ 회사의 시계열 내에서 자체 섹션으로 구할 것.

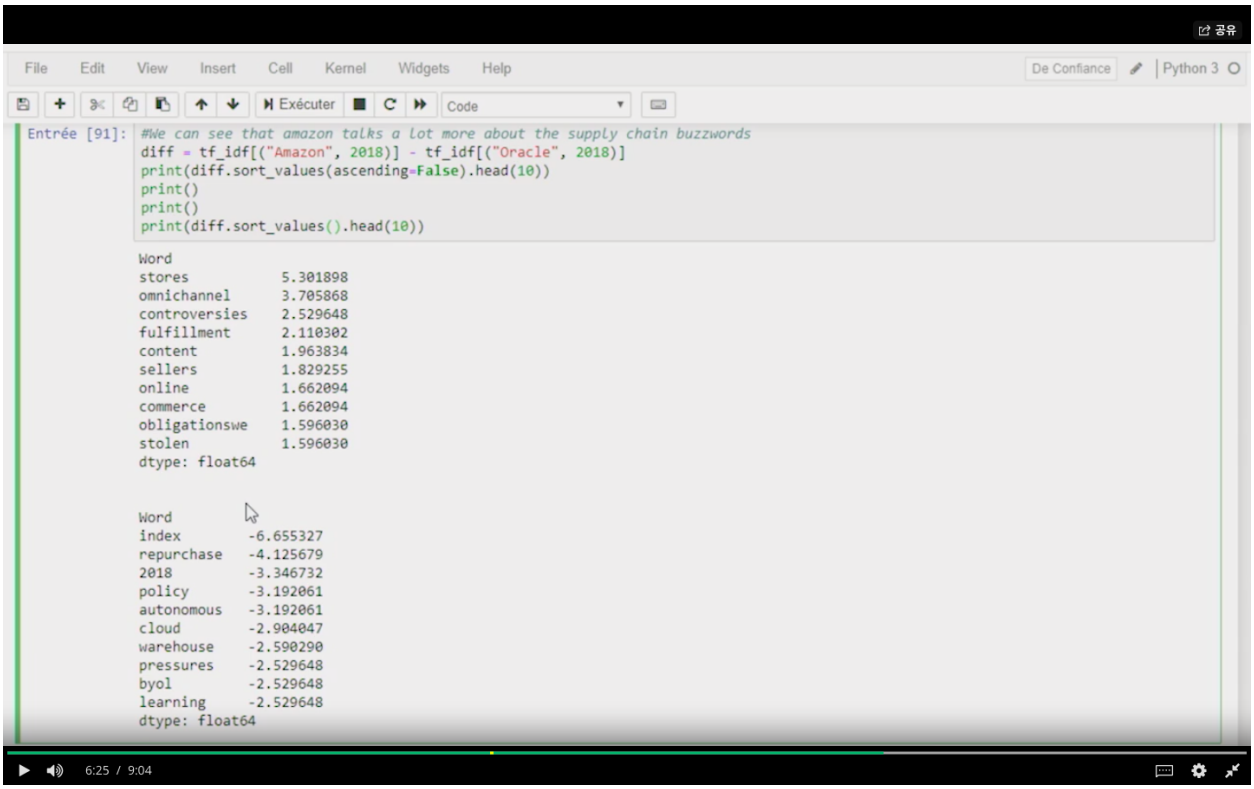
5) Amazon, Oracle 갖고 시간별 유사성 측정

```
Entrée [90]: #And for each time period find the similarity between the two
yearly_sim = pd.Series([similarity.loc[("Amazon",x),("Oracle", x)] for x in [2018,2017,2016,2015,2014]], index=[2018,2017,2016,2
print(yearly_sim)
```

```
2018    0.046559
2017    0.028756
2016    0.006662
2015    0.005095
2014    0.012158
dtype: float64
```

시간이 지날수록, 유사성은 커진다

6) tf-idf 에서 'diff'를 통해 다른 회사에 더 중요한 것(그외 단어) 추출



Amazon → supply chain, stores, omnichannel, fulfillment, sellers

Oracle → autonomous, cloud, warehouse

⇒ Amazon 과 Oracle의 경쟁 산업이 다른걸 알 수 있다! (Amazon : 온라인 상거래 중심)

7) 빈도수 높은 상위 5개 단어 추출

```
Entrée [92]:
#We see that two new words that amazon began using stores, and omnichannel became a big difference
#One limitation is that words are split
#So if amazon mentions whole foods it would be split losing the actual importance of it being a company
#rather than two distinct words
i = diff.sort_values(ascending=False).head(5).index
print(counts.loc[i])
```

Company	Amazon					Oracle				
Time Period	2014	2015	2016	2017	2018	2014	2015	2016	2017	2018
Word										
stores	0.0	0.0	0.0	0.0	9.0	0.0	0.0	0.0	0.0	0.0
omnichannel	0.0	0.0	0.0	0.0	4.0	0.0	0.0	0.0	0.0	0.0
controversies	0.0	0.0	0.0	0.0	2.0	0.0	0.0	0.0	0.0	0.0
fulfillment	20.0	20.0	20.0	20.0	20.0	0.0	0.0	0.0	0.0	0.0
content	16.0	16.0	16.0	16.0	16.0	0.0	0.0	0.0	0.0	0.0

```
Entrée [93]:
#This analysis can be useful to assess country level threats
#And correlations in the currency markets
print(counts.loc[['yemen', 'turkey', 'china']])
```

Company	Amazon					Oracle				
Time Period	2014	2015	2016	2017	2018	2014	2015	2016	2017	2018
Word										
yemen	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0
turkey	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0	0.0
china	2.0	3.0	3.0	3.0	3.0	2.0	2.0	2.0	2.0	2.0

Amazon → omnichannel, controversies, fulfillment, content.

Amazon에만 있고 Oracle에는 없음

⇒ tf-idf 높은 이유

국가별 FX시장 위협 구하는데 활용 가능

4주차

데이터의 종류

- Media
  - 신문, 보도, 전문지, 소셜
- 두가지 목표
  1. 우리과 필요한 주제에 대해서 어떤 의견을 가지고 있는지?
  2. 이렇. 수익률, 변동성, 리스크 이런걸 직관을 통해서 예측하는 것
- 미디어자료는 얻기는 쉬운데 이걸 가지고 응용을 하는 것은 어렵다. 왜냐면
  1. 일단 이게 어떻게 연관되는지를 잘 모르고
     - 파운드스털링 vs 라힘 스털링, 터키 vs 칠면조
  2. 뭐 관련이 있다 칠때 그 영향력/"톤"을 평가하기가 어렵다. e.g:
     - 미디어의 정치성향 (우파는 증시가 잘될때, 좌파는 잘 안 될때 언급을 많이 하더라)
     - 일반지가 전문지보다 전문분야에 대해서 부정적
     - 주기를 가지고 다루는 언론은 당연히 주기를 보겠죠?
     - 지역지일수록 지역뉴스에 호의적
     - 문서 길이가 길수록 중립적

감정 분석

- 하버드 감정사전: 77가지 감정카테고리를 나눔. 좋음 아주좋음 뭐 이런곳으로 분류가 될 수 있는...
- 각 감정카테고리에 포함되는 단어들의 목록들이 있음
- 문서에서 그런 단어가 발견될 때만 해당 감정의 점수를 높임
- 많으면 그 감정이 세다는 뜻
- 예시
  - 긍정카운트 - 부정카운트
  - 그런식으로 수익률을 예측해 봤는데, 부정적인 감정으로 예측할 땐 잘 맞는데, 긍정적인 감정으로 예측할 때는 잘 안 맞더라
  - 왜 안될까? - 단어가 금융시장이 아닌 심리학에서 작성되었기 때문에 아무래도 느낌이 좀 다르겠죠?
- 개선 1
  - 실제로 사전목록을 업데이트해서 다시 돌려보니까 결과가 좀 더 좋았다.
  - 부정쪽에서는 확 나아졌지만 긍정쪽은 그닥.
- 개선 2
  - 단어마다 가중치를 다르게 줌. 뭔가 영향력 클거같은 건 점수 크게 적은건 적게
  - 부정방향도 잘 맞췄지만, 이건 긍정방향도 잘 맞추더라

페이지랭크 알고리즘

- 래리페이지가 구글 만들때 썼던, 문서별 유사도 매기는 알고리즘
- 알고리즘
  - 각 문서(노드)에 같은 "랭크"를 분배
  - while not 수렴
    - 각 노드가 링크를 가진 다른 노드들에 자신의 랭크를 N빵해서 배분
    - 이 작업을 모든 노드에서 하고, N빵을 받은 결과를 합
- 트위터 매트릭스로 돌아가서
- 붙어있으면 걍 다 1이라고 치고
- 대각성분은 다 0으로 처리... 자기 스스로 돌아가는건 좋지 않아서욜
- 마스크 행렬을 만들고, 그 조건이 true면 0.