

Lab – News Categorization

File

최성철 교수
Director of TEAMLAB

01100
00110

**비슷한 뉴스를
어떻게 선정할까?**

가상화폐 광풍

거래소 폐쇄 방침 '혼선'에 비트코인 가격 '롤러코스터'

연합뉴스 · 203



비트코인 '거래소 폐지 방침'에 우르르..한... 연합뉴스

정부 가상화폐 때리기에 靑 몰려간 투자자... 연합뉴스

비트코인 광맥 끊기나..중국 채굴 전면 금지

머니투데이 · 327



오라클 웹로직 서버, 암호화폐 채굴에 쓰였다 지디넷코리아
"경찰에 국세청까지" 규제공포에 암호화폐 ... 뉴스1

"비트코인은 인생의 동아줄" 2030은 왜?

조선일보 · 441



4차 산업혁명..국민 관심사는 암호화폐? 지디넷코리아

페이스북 부사장 "한국투자 확대..중소기업과 협력 강화"

연합뉴스 · 7



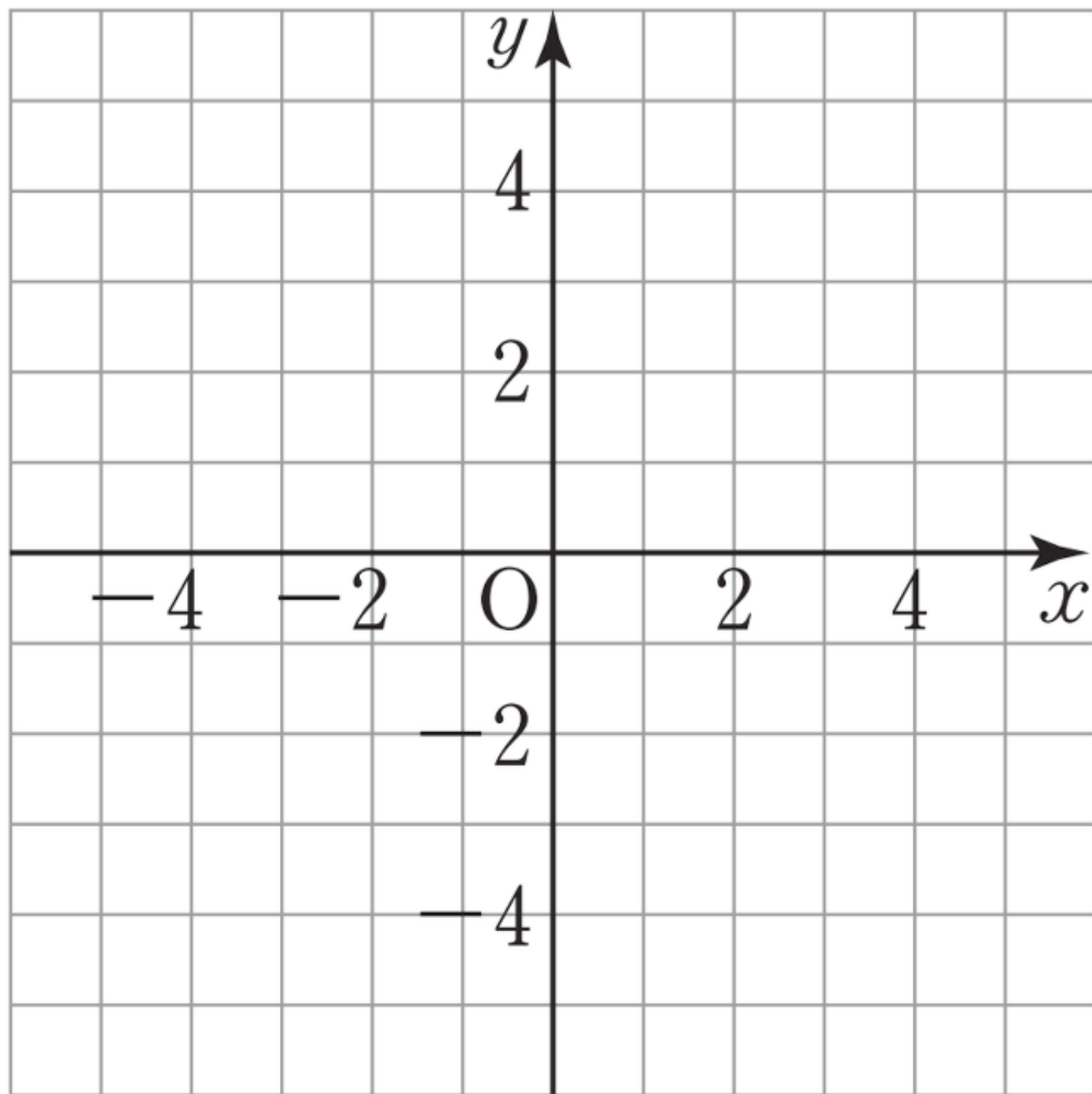
외국계IT기업 '국내 법적 대리인' 의무화 매일경제
'망 무임승차' 논란 페이스북 '사용료 문제' ... 연합뉴스

**컴퓨터는 문자를
그대로 이해하지 못함**

문자 → 숫자

**숫자로 유사하다는
어떻게 표현할까?**

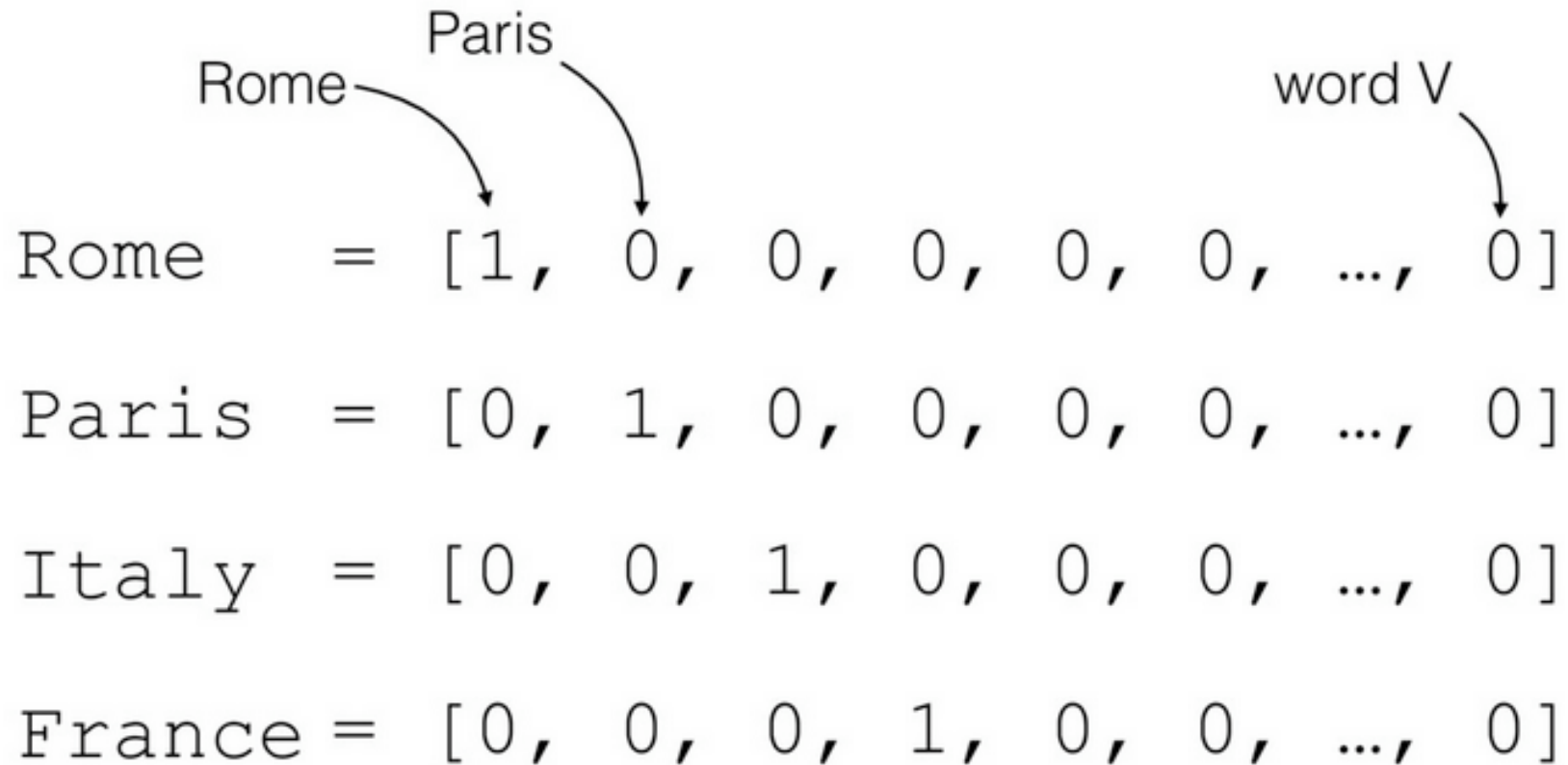
유사하다=가깝다



문자 → 숫자 → Vector

문자를 Vector로 – One-hot Encoding

- 하나의 단어를 Vector의 Index로 인식, 단어 존재시 1 없으면 0



Bag of words

- 단어별로 인덱스를 부여해서, 한 문장(또는 문서)의 단어의 개수를 Vector로 표현

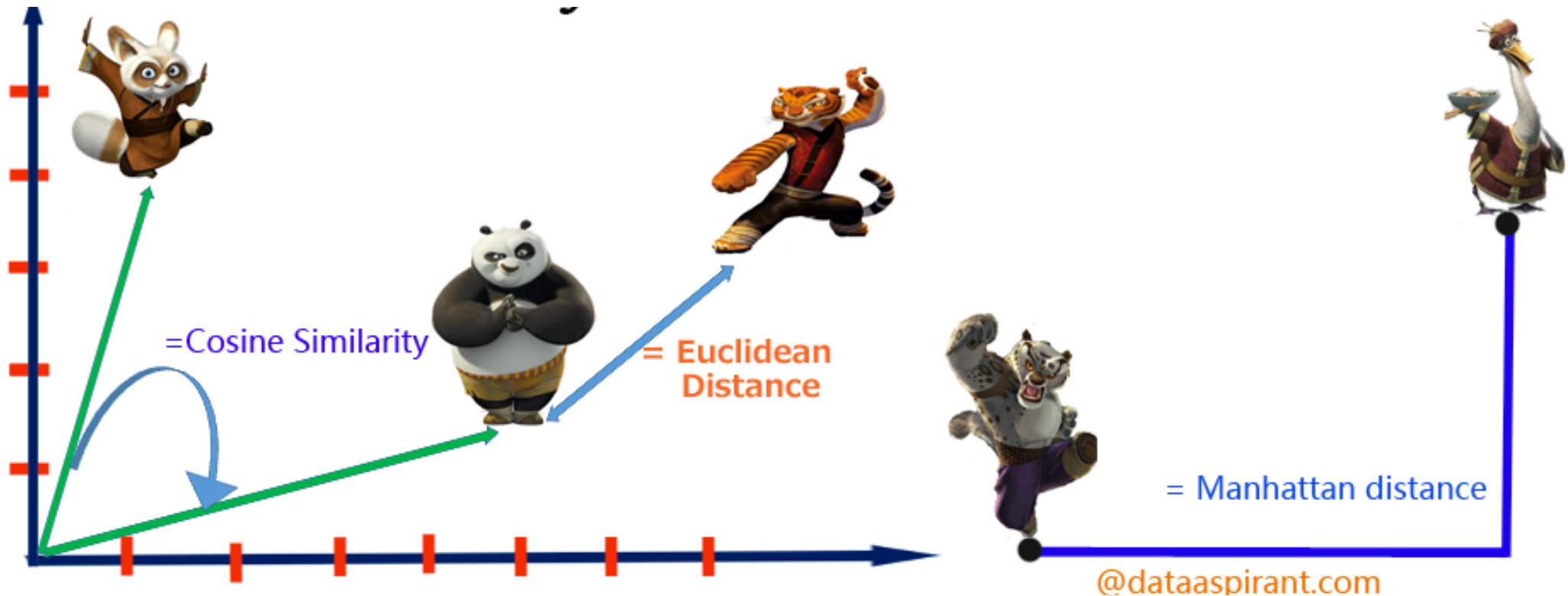
the dog is on the table

0	0	1	1	0	1	1	1
are	cat	dog	is	now	on	table	the

그렇다면 유사성은?

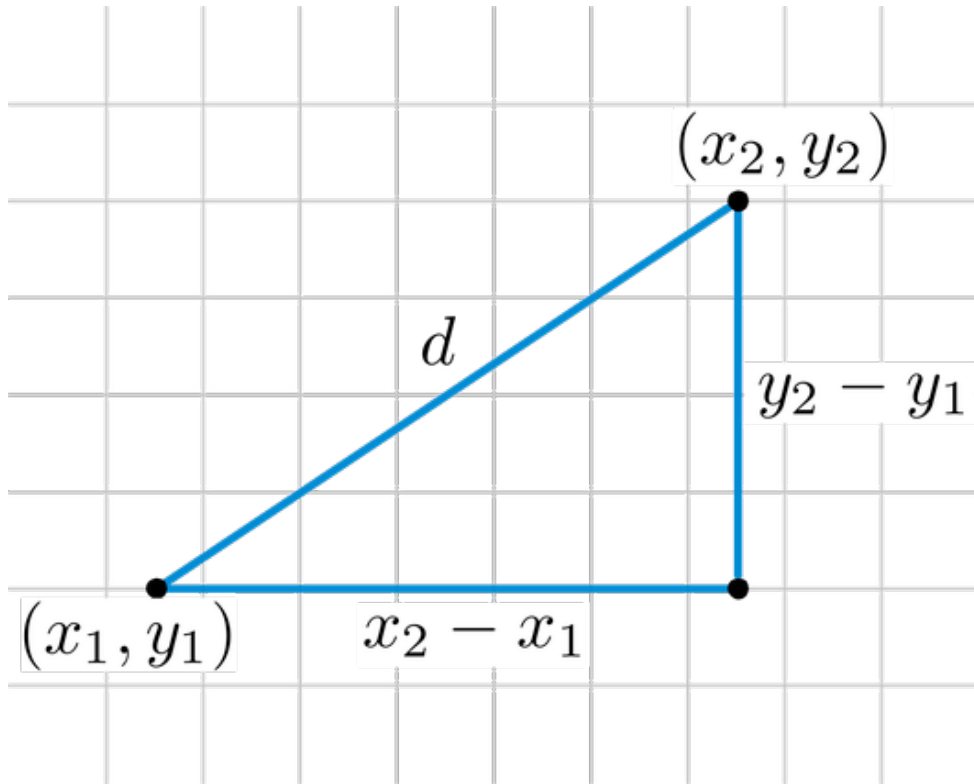
Distance measure

- 고등학교때 배운 2차원 평면상 거리측정 방법들



Euclidian distance

- 피타고라스 정리, 두 점 사이의 직선의 거리



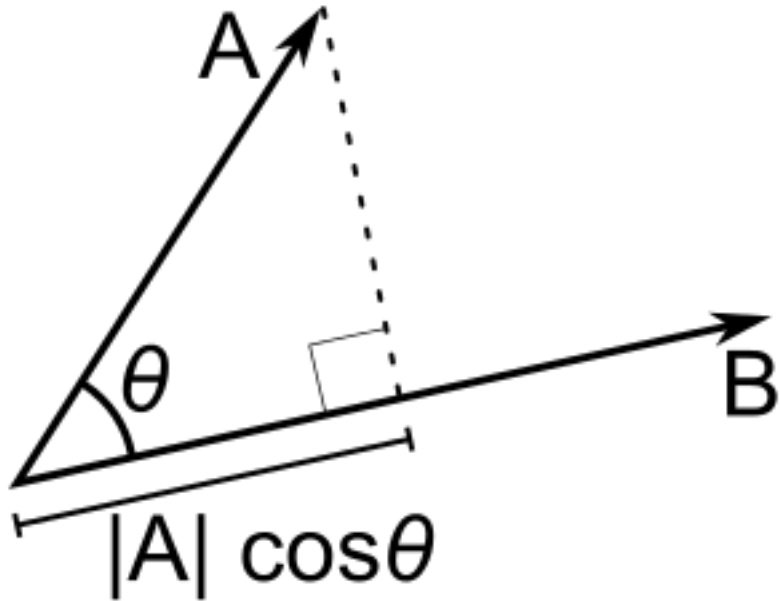
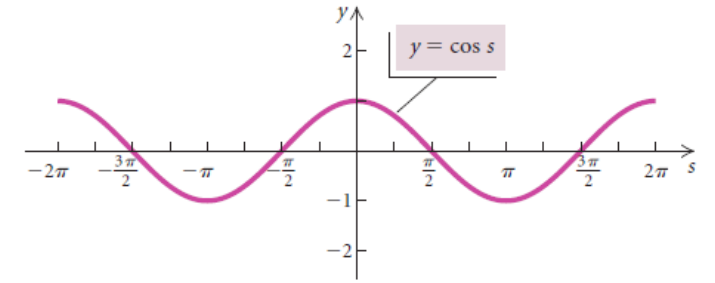
<https://goo.gl/cVmeKr>

$$\begin{bmatrix} a_1 & a_2 & a_3 & \dots & a_n \end{bmatrix}$$
$$\begin{bmatrix} b_1 & b_2 & b_3 & \dots & b_n \end{bmatrix}$$

$$d(a, b) = \sqrt{\sum_i^n (a_i - b_i)^2}$$

Cosine distance

- 두 점 사이의 각도



$$\cos(\theta) = \frac{A \cdot B}{\|A\| \|B\|} = \frac{\sum_{i=0}^n A_i \times B_i}{\sqrt{\sum_{i=0}^n (A_i)^2} \times \sqrt{\sum_{i=0}^n (B_i)^2}}$$

$$\therefore A \cdot B = AB \cos(\theta)$$

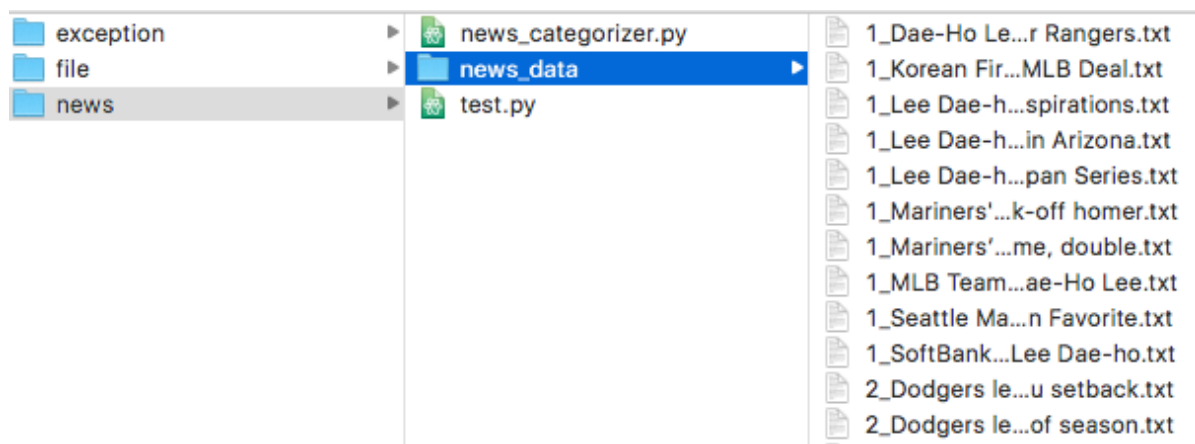
Cosine distance

- Why cosine similarity? Count < Direction
- Love,hate (5,0), (5,4), (4,0) , 어느점이 가장 가까운가

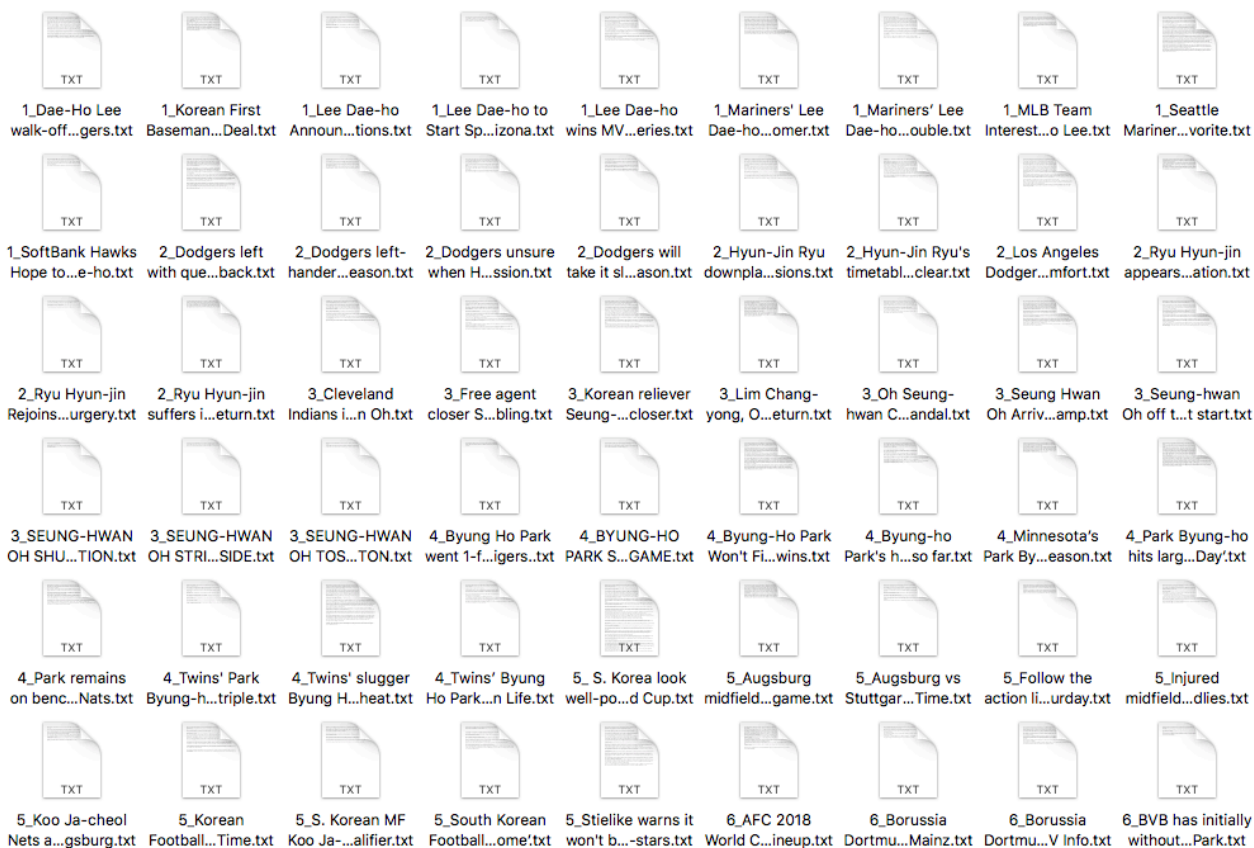
Codes

Data set

- 축구와 야구 선수들의 영문 기사를 분류해보자!



1,2,3,4 야구
5,6,7,8 축구



Process

- 파일을 불러오기
- 파일을 읽어서 단어사전 (corpus) 만들기
- 단어별로 Index 만들기
- 만들어진 인덱스로 문서별로 Bag of words vector 생성
- 비교하고자 하는 문서 비교하기
- 얼마나 맞는지 측정하기

파일 불러오기

```
def get_file_list(dir_name):  
    return os.listdir(dir_name)
```

```
if __name__ == "__main__":  
    dir_name = "news_data"  
    file_list = get_file_list(dir_name)  
    file_list = [os.path.join(dir_name, file_name) for file_name in file_list]
```

파일별로 내용읽기

```
def get_contents(file_list):
    y_class = []
    X_text = []
    class_dict = {
        1: "0", 2: "0", 3: "0", 4: "0", 5: "1", 6: "1", 7: "1", 8: "1"}

    for file_name in file_list:
        try:
            f = open(file_name, "r", encoding="cp949")
            category = int(file_name.split(os.sep)[1].split("_")[0])
            y_class.append(class_dict[category])
            X_text.append(f.read())
            f.close()
        except UnicodeDecodeError as e:
            print(e)
            print(file_name)
    return X_text, y_class
```

Corpus 만들기 + 단어별 index 생성하기

```
def get_cleaned_text(text):
```

의미없는 문장보호 등은 제거하기

```
    import re
```

```
    text = re.sub('\W+', '', text.lower() )
```

```
    return text
```

```
def get_corpus_dict(text):
```

```
    text = [sentence.split() for sentence in text]
```

```
    clenad_words = [get_cleaned_text(word) for words in text for word in words]
```

```
    from collections import OrderedDict
```

```
    corpus_dict = OrderedDict()
```

```
    for i, v in enumerate(set(clenad_words)):
```

```
        corpus_dict[v] = i
```

```
    return corpus_dict
```

문서별로 Bag of words vector 생성

```
def get_count_vector(text, corpus):  
    text = [sentence.split() for sentence in text]  
    word_number_list = [[corpus[get_cleaned_text(word)] for word in words]  
for words in text]  
    X_vector = [[0 for _ in range(len(corpus))] for x in range(len(text))]  
  
    for i, text in enumerate(word_number_list):  
        for word_number in text:  
            X_vector[i][word_number] += 1  
    return X_vector
```

비교하기

$$\cos(\theta) = \frac{A \cdot B}{\|A\| \|B\|} = \frac{\sum_{i=0}^n A_i \times B_i}{\sqrt{\sum_{i=0}^n (A_i)^2} \times \sqrt{\sum_{i=0}^n (B_i)^2}}$$

```
import math
def get_cosine_similarity(v1,v2):
    "compute cosine similarity of v1 to v2: (v1 dot
v2)/{||v1||*||v2||}"
    sumxx, sumxy, sumyy = 0, 0, 0
    for i in range(len(v1)):
        x = v1[i]; y = v2[i]
        sumxx += x*x
        sumyy += y*y
        sumxy += x*y
    return sumxy/math.sqrt(sumxx*sumyy)
```


비교결과 정리하기

```
def get_similarity_score(X_vector, source):  
    source_vector = X_vector[source]  
    similarity_list = []  
    for target_vector in X_vector:  
        similarity_list.append(  
            get_cosine_similarity(source_vector, target_vector))  
    return similarity_list
```

```
def get_top_n_similarity_news(similarity_score, n):  
    import operator  
    x = {i:v for i, v in enumerate(similarity_score)}  
    sorted_x = sorted(x.items(), key=operator.itemgetter(1))  
  
    return list(reversed(sorted_x))[1:n+1]
```

성능 측정하기

```
def get_accuracy(similarity_list, y_class, source_news):
    source_class = y_class[source_news]

    return sum([source_class == y_class[i[0]] for i in similarity_list]) /
len(similarity_list)

for i in range(80):
    source_number = i

    similarity_score = get_similarity_score(X_vector, source_number)
    similarity_news = get_top_n_similarity_news(similarity_score, 10)
    accuracy_score = get_accuracy(similarity_news, y_class, source_number)
    result.append(accuracy_score)
print(sum(result) / 80)
```



Human knowledge belongs to the world.