자연어 처리를 이용한 감정 분석

word2vec
RandomForestClassifier

학과: 컴퓨터공학과
학번: 2017108243
학년: 3학년
이름: 강성범
날짜: 21.11.24

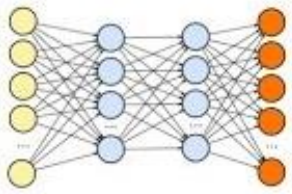
CONTENTS

01

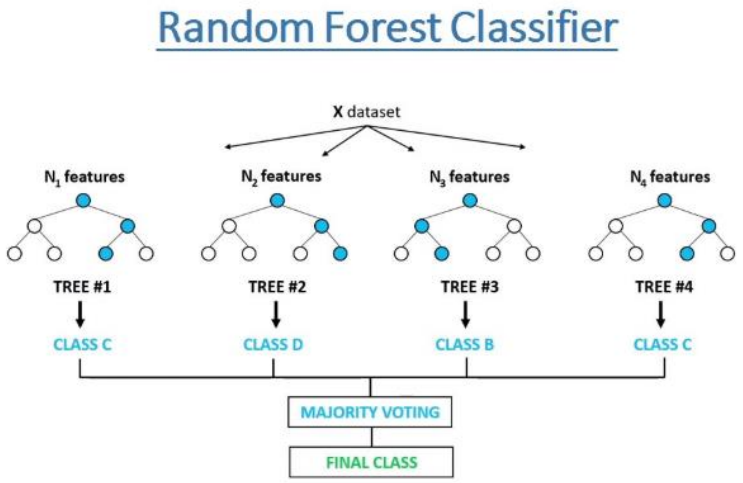
개요

1.1 요약

단어 벡터 변환 (word2vec)



딥러닝 자연어 처리



딥러닝(word2vec)과 지도학습(RandomForestClassifier)을 사용하여 영화 리뷰 긍정, 부정 확인

1.2 word2vec-1

Word2vec이란?

- 단어들을 벡터화 하여 단어들 간의 유사도를 확인하는 딥러닝

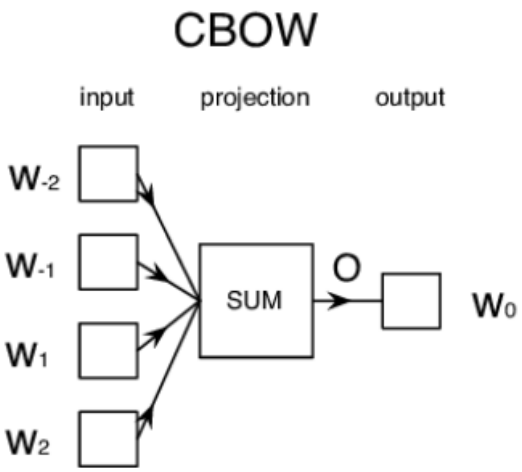
Wor2vec의 두 가지 방식

1. CBOW(Continuous Bag of Words)
   - 주변의 단어들을 이용하여 단어 예측
2. Skip-Gram
   - 중심 단어로 단어 예측

{"The", "fat", "cat", "on", "the", "mat"}으로부터 sat을 예측

중심 단어 / 주변 단어

중심 단어	주변 단어
The fat cat sat on the mat	
[1, 0, 0, 0, 0, 0]	[0, 1, 0, 0, 0, 0], [0, 0, 1, 0, 0, 0]
The fat cat sat on the mat	
[0, 1, 0, 0, 0, 0]	[1, 0, 0, 0, 0, 0], [0, 0, 1, 0, 0, 0], [0, 0, 0, 1, 0, 0]
The fat cat sat on the mat	
[0, 0, 1, 0, 0, 0]	[1, 0, 0, 0, 0, 0], [0, 1, 0, 0, 0, 0], [0, 0, 0, 1, 0, 0], [0, 0, 0, 0, 1, 0]
The fat cat sat on the mat	
[0, 0, 0, 1, 0, 0]	[0, 1, 0, 0, 0, 0], [0, 0, 1, 0, 0, 0], [0, 0, 0, 0, 1, 0], [0, 0, 0, 0, 0, 1]
The fat cat sat on the mat	
[0, 0, 0, 0, 1, 0]	[0, 0, 1, 0, 0, 0], [0, 0, 0, 1, 0, 0], [0, 0, 0, 0, 0, 1]
The fat cat sat on the mat	
[0, 0, 0, 0, 0, 1]	[0, 0, 0, 1, 0, 0], [0, 0, 0, 0, 1, 0]

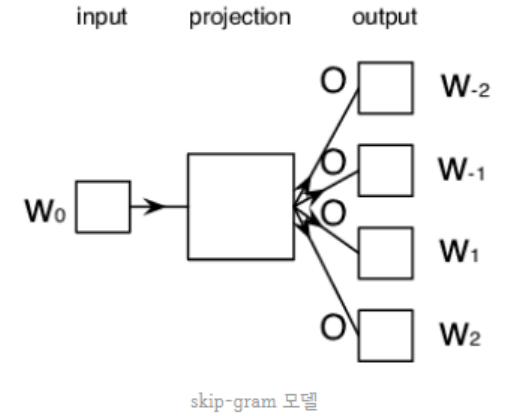


중심 단어 / 주변 단어

The fat cat sat on the mat

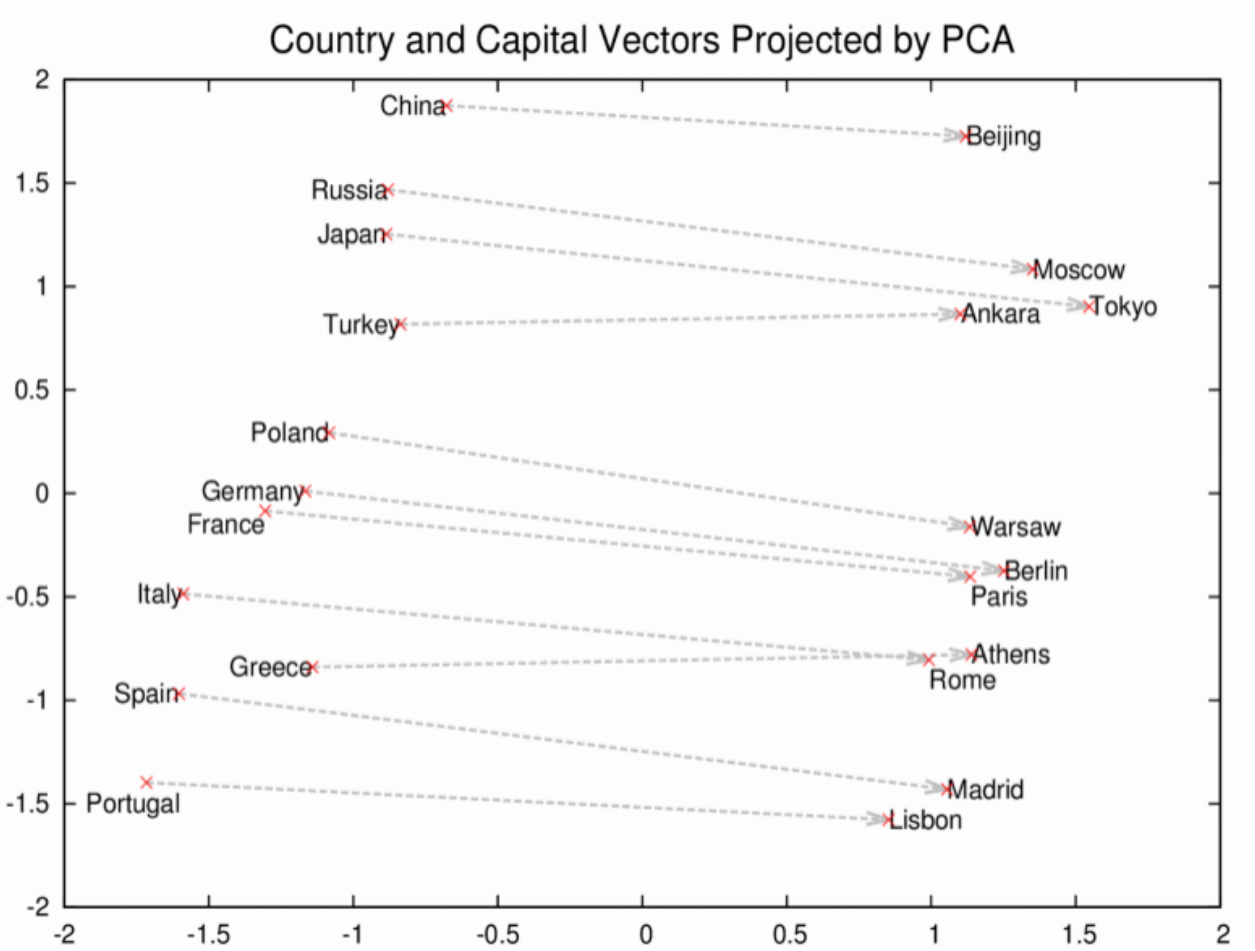
The fat cat sat on the mat

중심 단어	주변 단어
cat	The
cat	Fat
cat	sat
cat	on
sat	fat
sat	cat
sat	on
sat	the



skip-gram 모델

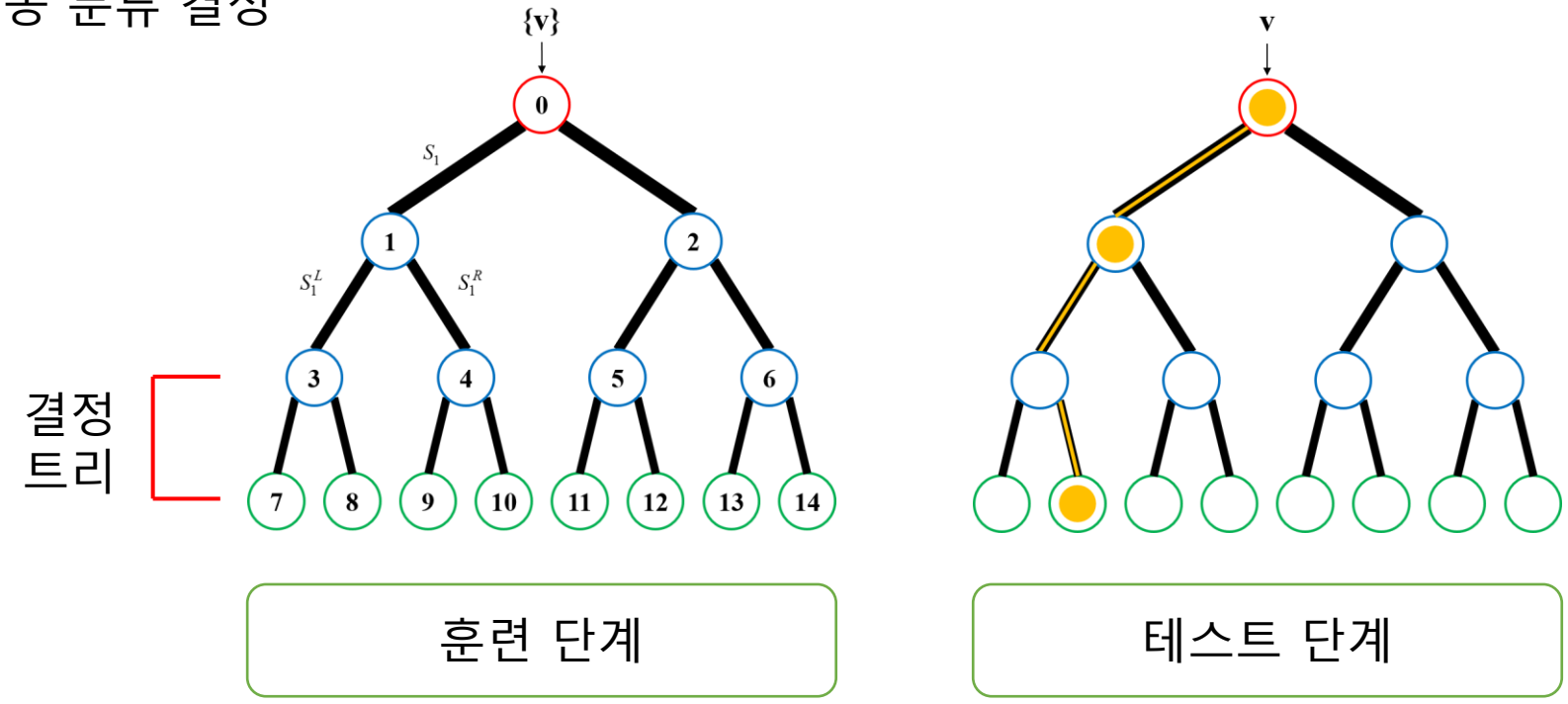
1.2 word2vec-2



1.3 RandomForestClassifier

RandomForestClassifier 이란?

- 다수의 결정 트리를 사용하여 분류를 하는 알고리즘
- 여러 개의 결정 트리로 구성하고, 새로운 데이터 포인트를 각 트리에 통과 및 투표를 하여 최종 분류 결정



02

데이터 세트

2.1 데이터

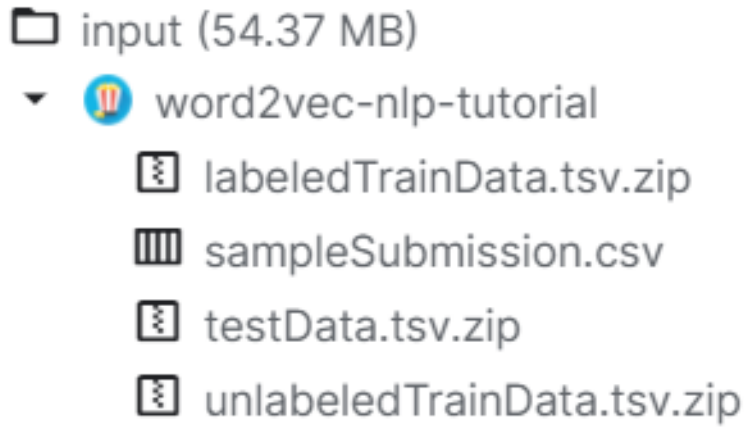
```
import pandas as pd

df_train = pd.read_csv("../input/word2vec-nlp-tutorial/labeledTrainData.tsv.zip", header = 0,
                       delimiter = "\t", quoting = 3)
df_test = pd.read_csv("../input/word2vec-nlp-tutorial/testData.tsv.zip", header = 0,
                       delimiter = "\t", quoting = 3)
df_unlabeled = pd.read_csv("../input/word2vec-nlp-tutorial/unlabeledTrainData.tsv.zip", header = 0,
                       delimiter = "\t", quoting = 3)

print(df_train.shape)
print(df_test.shape)
print(df_unlabeled.shape)

print(df_train["review"].size)
print(df_test["review"].size)
print(df_unlabeled["review"].size)
```

```
(25000, 3)
(25000, 2)
(50000, 2)
25000
25000
50000
```




```
df_train.head()
```

	id	sentiment	review
0	"5814_8"	1	"With all this stuff going down at the moment ...
1	"2381_9"	1	"\"The Classic War of the Worlds\" by Timothy ...
2	"7759_3"	0	"The film starts with a manager (Nicholas Bell...
3	"3630_4"	0	"It must be assumed that those who praised thi...
4	"9495_8"	1	"Superbly trashy and wondrously unpretentious ...

```
df_test.head()
```

	id	review
0	"12311_10"	"Naturally in a film who's main themes are of ...
1	"8348_2"	"This movie is a disaster within a disaster fi...
2	"5828_4"	"All in all, this is a movie for kids. We saw ...
3	"7186_2"	"Afraid of the Dark left me with the impressio...
4	"12128_7"	"A very accurate depiction of small time mob l...

```
df_unlabeled.head()
```

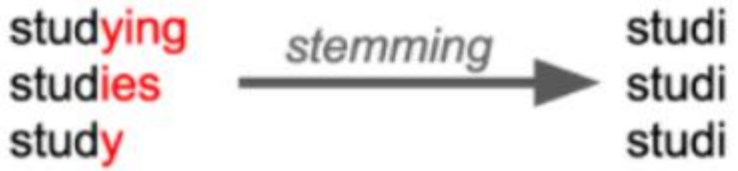
	id	review
0	"9999_0"	"Watching Time Chasers, it obvious that it was...
1	"45057_0"	"I saw this film about 20 years ago and rememb...
2	"15561_0"	"Minor Spoilers In New York, Joan B...
3	"7161_0"	"I went to see this film with a great deal of ...
4	"43971_0"	"Yes, I agree with everyone on this site this ...

2.2 데이터 전처리-1

```
def review_to_wordlist(review, remove_stopwords=False):
    # 1. HTML 제거
    review_text = BeautifulSoup(review, "html.parser").get_text()
    # 2. 특수문자를 공백으로 바꿔줌
    review_text = re.sub('[^a-zA-Z]', ' ', review_text)
    # 3. 소문자로 변환 후 나눈다.
    words = review_text.lower().split()
    # 4. 불용어 제거
    if remove_stopwords:
        stops = set(stopwords.words('english'))
        words = [w for w in words if not w in stops]
    # 5. 어간추출
    stemmer = SnowballStemmer('english')
    words = [stemmer.stem(w) for w in words]
    # 6. 리스트 형태로 반환
    return(words)
```

#4 불용어 : 어휘에서 잘 사용되지 않는 용어

#5 어간 추출



2.2 데이터 전처리-2

```
def review_to_sentences( review, remove_stopwords=False ):
    # punkt tokenizer를 로드한다.
    """
    이 때, pickle을 사용하는데
    pickle을 통해 값을 저장하면 원래 변수에 연결 된 참조값 역시 저장된다.
    저장된 pickle을 다시 읽으면 변수에 연결되었던
    모든 레퍼런스가 계속 참조 상태를 유지한다.
    """
    tokenizer = nltk.data.load('tokenizers/punkt/english.pickle')
    # 1. nltk tokenizer를 사용해서 단어로 토큰화 하고 공백 등을 제거한다.
    raw_sentences = tokenizer.tokenize(review.strip())
    # 2. 각 문장을 순회한다.
    sentences = []
    for raw_sentence in raw_sentences:
        # 비어있다면 skip
        if len(raw_sentence) > 0:
            # 태그제거, 알파벳문자가 아닌 것은 공백으로 치환, 불용어제거
            sentences.append(\
                KaggleWord2VecUtility.review_to_wordlist(\
                raw_sentence, remove_stopwords))
    return sentences
```

```
sentences = []
for review in df_train["review"]:
    sentences += KaggleWord2VecUtility.review_to_sentences(
    review, remove_stopwords = False)

# KaggleWord2VecUtility을 사용하여 train 데이터를 정제해준다.
```

```
for review in df_unlabeled["review"]:
    sentences += KaggleWord2VecUtility.review_to_sentences(
    review, remove_stopwords = False)

# KaggleWord2VecUtility을 사용하여 unlabeled train 데이터를 정제해준다.
```

전처리 전

"With all this stuff going down at the moment ...

전처리 후

```
sentences[0][:10]
```

```
['with', 'all', 'this', 'stuff', 'go', 'down', 'at', 'the', 'moment', 'with']
```

03

word2vec

3.1 모델 학습

```
# 파라미터 값을 지정해준다.

num_features = 300 # 문자 벡터 차원 수 (size)
min_word_count = 40 # 최소 문자 수 (min_count)
num_workers = 4 # 병렬 처리 스레드 수 (workers)
context = 10 # 문자열 창 크기 (window)
downsampling = 1e-3 # 문자 빈도 수 Downsample (sample)

# 초기화 및 모델 학습
from gensim.models import word2vec

model = word2vec.Word2Vec(sentences,
                          workers = num_workers,
                          size = num_features,
                          min_count = min_word_count,
                          window = context,
                          sample = downsampling)

model
```

```
<gensim.models.word2vec.Word2Vec at 0x7f31d36c66a0>
```

3.2 모델 저장

```
# 학습이 완료되면 필요없는 메모리를 unload 시킨다.
model.init_sims(replace = True)

model_name = "300features_40minwindows_10text"
model.save(model_name)
```

3.3 모델 결과

```
# 유사도가 없는 단어 추출
model.wv.doesnt_match("man woman child kitchen".split())
```

```
'kitchen'
```

```
model.wv.doesnt_match("france england germany berlin".split())
```

```
'berlin'
```

```
# 가장 유사한 단어를 추출
model.wv.most_similar("man")
```

```
[('woman', 0.629759669303894),
 ('lad', 0.5377025604248047),
 ('ladi', 0.5119224786758423),
 ('businessman', 0.5069717764854431),
 ('millionair', 0.4861908257007599),
 ('farmer', 0.4848712682723999),
 ('men', 0.48129451274871826),
 ('policeman', 0.46441251039505005),
 ('lawyer', 0.4539344310760498),
 ('widow', 0.4527662992477417)]
```

```
model.wv.most_similar("queen")
```

```
[('princess', 0.5945835113525391),
 ('stepmoth', 0.5683339834213257),
 ('latifah', 0.5520309805870056),
 ('madam', 0.5387923717498779),
 ('maid', 0.5379070043563843),
 ('eva', 0.5349502563476562),
 ('victoria', 0.532073974609375),
 ('countess', 0.5209550261497498),
 ('goddess', 0.5203566551208496),
 ('mistress', 0.5202128887176514)]
```

3.4 모델 시각화-1

```
from sklearn.manifold import TSNE
import matplotlib as mpl
import matplotlib.pyplot as plt
import gensim
import gensim.models as g

# 그래프에서 마이너스 폰트 깨지는 문제에 대한 대처
mpl.rcParams["axes.unicode_minus"] = False

model_name = "300features_40minwindows_10text"
model = g.Doc2Vec.load(model_name)

vocab = list(model.wv.vocab)
X = model[vocab]

print(len(X))
print(X[0][:10])
tsne = TSNE(n_components = 2)

# 100개의 단어에 대해서만 시각화
X_tsne = tsne.fit_transform(X[:100,:])
```

```
11986
[ 0.00440767  0.04906967 -0.03090064 -0.03337254  0.08941759  0.05634528
 -0.06284355  0.0608907   0.02803548 -0.12058227]
```

t-SNE
- 비선형으로 고차원의 데이터를 2, 3차원 으로 축소할 수 있다.

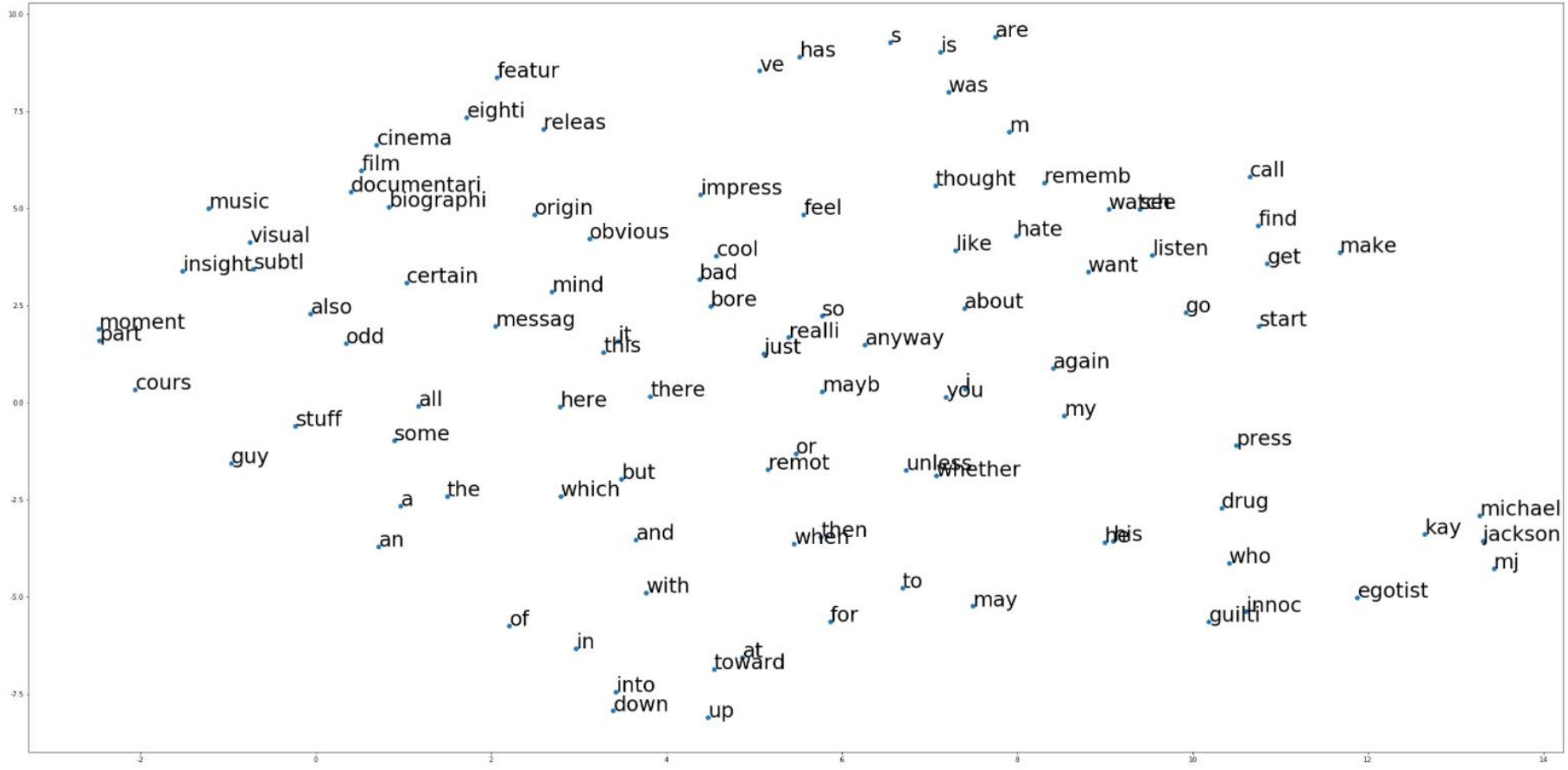
```
df = pd.DataFrame(X_tsne, index = vocab[:100], columns = ["x", "y"])
df.shape
```

```
(100, 2)
```

```
df.head()
```

	x	y
with	3.769738	-4.899901
all	1.169591	-0.083203
this	3.278209	1.289519
stuff	-0.231542	-0.610469
go	9.921402	2.320514

3.4 모델 시각화-2




```
fig = plt.figure()
fig.set_size_inches(40, 20)
ax = fig.add_subplot(1, 1, 1)

ax.scatter(df["x"], df["y"])

for word, pos in df.iterrows():
    ax.annotate(word, pos, fontsize = 30)

plt.show()
```

3.4 계산 및 배열 변환-1

```python
def _apply_df(args):
    df, func, kwargs = args
    return df.apply(func, **kwargs)
```

멀티 스레드 작업을 하여 작업 속도를 올리는 함수

```python
def apply_by_multiprocessing(df, func, **kwargs):
    # 키워드 항목 중 workers 파라메터를 꺼냄
    workers = kwargs.pop('workers')
    # 위에서 가져온 workers 수로 프로세스 풀을 정의
    pool = Pool(processes=workers)
    # 실행할 함수와 데이터프레임을 워커의 수 만큼 나눠 작업
    result = pool.map(KaggleWord2VecUtility._apply_df, [(d, func, kwargs)
            for d in np.array_split(df, workers)])
    pool.close()
    # 작업 결과를 합쳐서 반환
    return pd.concat(result)
```

3.4 계산 및 배열 변환-2

```python
import numpy as np


# def를 이용해서 주어진 문장에서 단어 벡터의 평균을 구하는 함수를 만든다.
def makeFeatureVec(words, model, num_features):
    featureVec = np.zeros((num_features,),dtype = "float32")

    # 속도를 위해 0으로 채운 배열로 초기화 한다.
    nwords = 0.
    # Index2word는 모델의 사전에 있는 단어명을 담은 리스트이다.
    # 속도를 위해 set 형태로 초기화 한다.
    index2word_set = set(model.wv.index2word)
    # 루프를 돌며 모델 사전에 포함이 되는 단어라면 피처에 추가한다.
    for word in words:
        if word in index2word_set:
            nwords = nwords + 1.
            featureVec = np.add(featureVec, model[word])
    # 결과를 단어수로 나누어 평균을 구한다.
    featureVec = np.divide(featureVec, nwords)
    return featureVec
```

단어 벡터 평균 구하는 함수

3.4 계산 및 배열 변환-3

```
def getAvgFeatureVecs(reviews, model, num_features):
    # 리뷰 단어 목록의 각각에 대한 평균 feature 벡터를 계산하고
    # 2d Numpy Array로 반환한다.

    # 카운터를 초기화 한다.
    counter = 0.
    # 속도를 위해 2D 넘파이 배열을 미리 할당한다.
    reviewFeatureVecs = np.zeros(
        (len(reviews), num_features), dtype = "float32")

    for review in reviews:
        # 매 1000개 리뷰마다 상태를 출력
        if counter%1000. == 0.:
            print("Review %d of %d"%(counter, len(reviews)))
        # 평균 피처 벡터를 만들기 위해 위에서 정의한 함수를 호출한다.
        reviewFeatureVecs[int(counter)] = makeFeatureVec(review,
                                                         model,
                                                        num_features)
        # 카운터를 증가시킨다.
        counter = counter + 1.
    return reviewFeatureVecs
```

각 리뷰마다 단어벡터 평균을 구해 반환하는 함수

3.4 계산 및 배열 변환-4

```
# 멀티스레드로 4개의 워커를 사용해 처리한다.

def getCleanReviews(reviews):
    clean_reviews = []
    clean_reviews = KaggleWord2VecUtility.apply_by_multiprocessing(\
        reviews["review"], KaggleWord2VecUtility.review_to_wordlist,\
        workers = 4)
    return clean_reviews
```

```
%time trainDataVecs = getAvgFeatureVecs(\
    getCleanReviews(df_train), model, num_features)
```

```
%time testDataVecs = getAvgFeatureVecs(\
    getCleanReviews(df_test), model, num_features)
```

```
Review 0 of 25000
Review 1000 of 25000
Review 2000 of 25000
Review 3000 of 25000
Review 4000 of 25000
Review 5000 of 25000
Review 6000 of 25000
Review 7000 of 25000
Review 8000 of 25000
Review 9000 of 25000
Review 10000 of 25000
Review 11000 of 25000
Review 12000 of 25000
Review 13000 of 25000
Review 14000 of 25000
Review 15000 of 25000
Review 16000 of 25000
Review 17000 of 25000
Review 18000 of 25000
Review 19000 of 25000
Review 20000 of 25000
Review 21000 of 25000
Review 22000 of 25000
Review 23000 of 25000
Review 24000 of 25000
CPU times: user 1min 14s, sys: 536 ms, total: 1min 15s
Wall time: 2min 2s
```

04

RandomForestClassifier

4.1 모델학습

```
from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier(
    n_estimators = 100, n_jobs = -1, random_state = 42)
```

n_estimators : 트리의 개수
n_jobs : 코어 병렬처리 개수
[-1이면 컴퓨터의 모든 코어 사용]

```
%time rf.fit(trainDataVecs, df_train["sentiment"])
```

정답(0 or 1)

평균 벡터

```
CPU times: user 1min 6s, sys: 54 ms, total: 1min 6s
Wall time: 17 s
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=100,
                       n_jobs=-1, oob_score=False, random_state=42, verbose=0,
                       warm_start=False)
```

4.2 교차검증

```
from sklearn.model_selection import cross_val_score
%time score= np.mean(cross_val_score(\
    rf, trainDataVecs, df_train["sentiment"], cv = 10, scoring = "roc_auc"))
```

```
CPU times: user 5.98 s, sys: 843 ms, total: 6.82 s
Wall time: 2min 42s
```

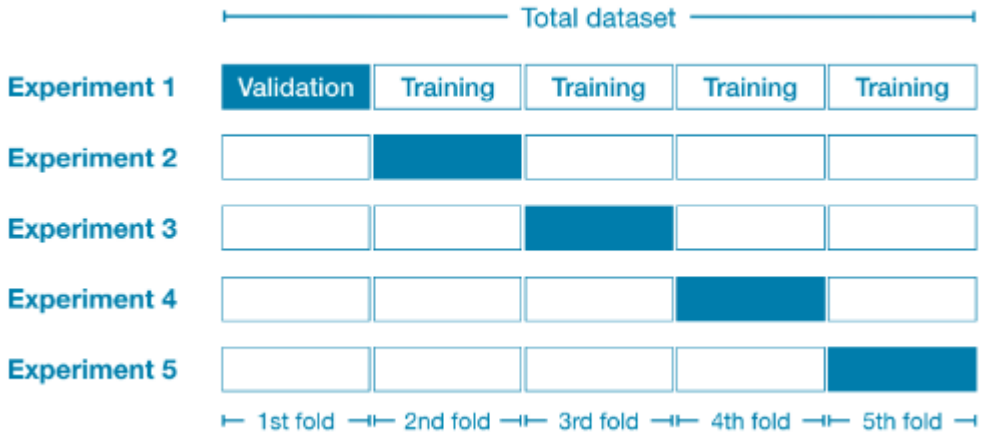
```
score
```

```
0.901574624
```

모델 성능 : 0.901574624

np.mean : 산술평균 계산

cross_val_score : 교차검증
cv : 중첩, 폴드



4.3 결과-1

```
result = rf.predict(testDataVecs)
```

```
[1 0 1 ... 0 1 0]
```

```
output = pd.DataFrame(data = {"id": df_test["id"], "sentiment": result})
output.to_csv("./Word2Vec_Tutorial_{:.5f}.csv".format(score),
              index = False, quoting = 3)
```

```
output_sentiment = output["sentiment"].value_counts()
print(output_sentiment[0] - output_sentiment[1])
output_sentiment
```

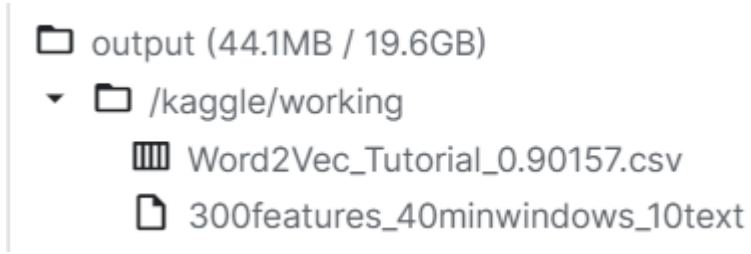
```
162
0    12581
1    12419
Name: sentiment, dtype: int64
```

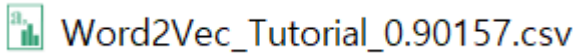
부정(0) : 12581
긍정(1) : 12419

4.3 결과-2

	A	B	C	D	E	F	G	H
1	id	sentiment						
2	12311_10	1						
3	8348_2	0						
4	5828_4	1						
5	7186_2	0						
6	12128_7	1						
7	2913_8	1						
8	4396_1	0						
9	395_2	0						
10	10616_1	0						
11	9074_9	1						
12	9252_3	1						
13	9896_9	1						
14	574_4	0						
15	11182_8	1						
16	11656_4	0						
17	2322_4	1						
18	8703_1	1						
19	7483_1	0						
20	6007_10	1						
21	12424_4	0						
22	4672_1	0						

Word2Vec_Tutorial_0.90157





참고

IMDb review NLP Tutorial Part 2
https://www.kaggle.com/kongnyooong/imdb-review-nlp-tutorial-part-2

Meets Bags of Popcorn - A Beginner's Notebook
https://www.kaggle.com/kobeerose/meets-bags-of-popcorn-a-beginner-s-notebook

sentiment analysis 101 using Word2Vec
https://www.kaggle.com/rawaaelghali/sentiment-analysis-101-using-word2vec

word2vec
https://wikidocs.net/22660

Q & A