실습

word2vec

Develop Word2Vec

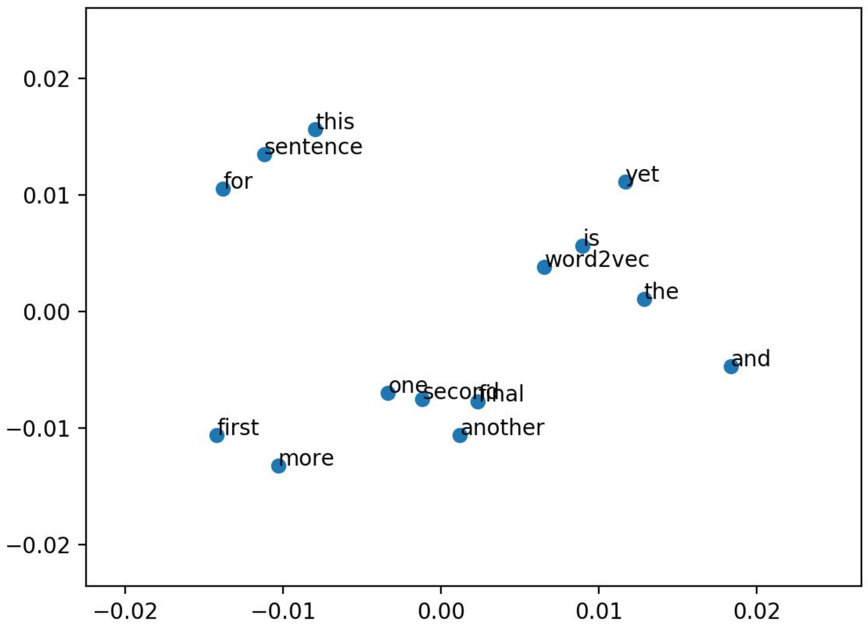
Parameters:

size: (default 100) The number of dimensions of the embedding, e.g. the length of the dense vector to represent each token (word).
window: (default 5) The maximum distance between a target word and words around the target word.
min_count: (default 5) The minimum count of words to consider when training the model; words with an occurrence less than this count will be ignored.
sg: (default 0 or CBOW) The training algorithm, either CBOW (0) or skip gram (1).

```
from gensim.models import Word2Vec
# define training data
sentences = [['this', 'is', 'the', 'first', 'sentence', 'for', 'word2vec'],
             ['this', 'is', 'the', 'second', 'sentence'],
             ['yet', 'another', 'sentence'],
             ['one', 'more', 'sentence'],
             ['and', 'the', 'final', 'sentence']]
# train model
model = Word2Vec(sentences, min_count=1)
# summarize vocabulary
words = list(model.wv.vocab)
print(words)
# access vector for one word
print(model['sentence'])
# save model
model.save('model.bin')
# load model
new_model = Word2Vec.load('model.bin')
# summarize the loaded model
print(new_model)
```

Word2Vec model

Scatter Plot of PCA Projection of Word2Vec Model




```
['second', 'sentence', 'and', 'this', 'final', 'word2vec', 'for', 'another',
'one', 'first', 'more', 'the', 'yet', 'is']
[ -4.61881841e-03 -4.88735968e-03 -3.19508743e-03  4.08568839e-03
 -3.38211656e-03  1.93076557e-03  3.90265253e-03 -1.04349572e-03
  4.14286414e-03  1.55219622e-03  3.85653134e-03  2.22428422e-03
 -3.52565176e-03  2.82056746e-03 -2.11121864e-03 -1.38054823e-03
 -1.12888147e-03 -2.87318649e-03 -7.99703528e-04  3.67874932e-03
  2.68940022e-03  6.31021452e-04 -4.36326629e-03  2.38655557e-04
 -1.94210222e-03  4.87691024e-03 -4.04118607e-03 -3.17813386e-03
  4.94802603e-03  3.43150692e-03 -1.44031656e-03  4.25637932e-03
 -1.15106850e-04 -3.73274647e-03  2.50349124e-03  4.28692997e-03
 -3.57313151e-03 -7.24728088e-05 -3.46099050e-03 -3.39612062e-03
  3.54845310e-03  1.56780297e-03  4.58260969e-04  2.52689526e-04
  3.06256465e-03  2.37558200e-03  4.06933809e-03  2.94650183e-03
 -2.96231941e-03 -4.47433954e-03  2.89590308e-03 -2.16034567e-03
 -2.58548348e-03 -2.06163677e-04  1.72605237e-03 -2.27384618e-04
 -3.70194600e-03  2.11557443e-03  2.03793868e-03  3.09839356e-03
 -4.71800892e-03  2.32995977e-03 -6.70911541e-05  1.39375112e-03
 -3.84263694e-03 -1.03898917e-03  4.13251948e-03  1.06330717e-03
  1.38514000e-03 -1.18144893e-03 -2.60811858e-03  1.54952740e-03
  2.49916781e-03 -1.95435272e-03  8.86975031e-05  1.89820060e-03
 -3.41996481e-03 -4.08187555e-03  5.88635216e-04  4.13103355e-03
 -3.25899688e-03  1.02130906e-03 -3.61028523e-03  4.17646067e-03
  4.65870230e-03  3.64110398e-04  4.95479070e-03 -1.29743712e-03
 -5.03367570e-04 -2.52546836e-03  3.31060472e-03 -3.12870182e-03
 -1.14580349e-03 -4.34387522e-03 -4.62882593e-03  3.19007039e-03
  2.88707414e-03  1.62976081e-04 -6.05802808e-04 -1.06368808e-03]
Word2Vec(vocab=14, size=100, alpha=0.025)
```

목표 (TED talk or wikipedia data에 대하여 word2vec 모델 학습)

- Gensim 라이브러리 사용하여 훈련 후 단어 임베딩을 분석하고 가시화

- 50~100개의 가장 자주 쓰이는 단어와 발생 빈도 리스트 (sklearn.feature_extraction.text 의 CountVectorizer 클래스나 collections 모듈의 Counter 클래스 사용 가능)를 만들고 발생 빈도에 대하여 histogram 그리기 (플로팅 코드 제공)

- 제공: 데이터셋, 부분적으로 완성된 코드

- t-SNE plot에서 재미있는 클러스터를 찾아보세요(가능시)

코드 제공

기본 전처리, 차트 코드 제공.

Part 0: Download the TED dataset

```
import urllib.request
import zipfile
import lxml.etree
```

```
# Download the dataset if it's not already there: this may take a minute as it is 75MB
if not os.path.isfile('ted_en-20160408.zip'):
    urllib.request.urlretrieve("https://wit3.fbk.eu/get.php?path=XML_releases/xml/ted_en-20160408.zip&filename=ted_en-20160408.zip", filename
```

```
# For now, we're only interested in the subtitle text, so let's extract that from the XML:
with zipfile.ZipFile('ted_en-20160408.zip', 'r') as z:
    doc = lxml.etree.parse(z.open('ted_en-20160408.xml', 'r'))
input_text = '\n'.join(doc.xpath('//content/text()'))
del doc
```

데이터 구조

- 중첩 리스트 구조
- flatten 처리 필요 or 다른 방안 모색.
- 시간을 줄이기위해 10000개정도의 문장을 추출하여 사용

```
len(sentences_ted)
```

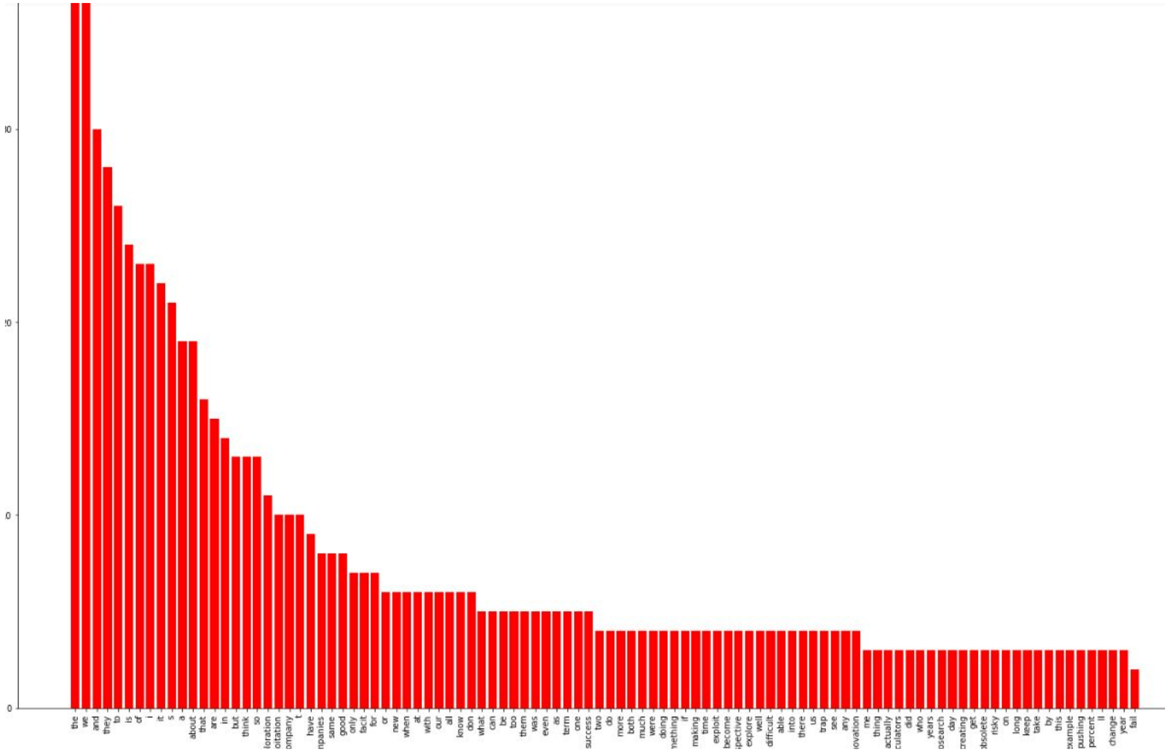
```
266694
```

```
print(sentences_ted[0])
print(sentences_ted[1])
```

```
['here', 'are', 'two', 'reasons', 'companies', 'fail', 'they', 'only', 'do', 'more', 'of', 'the', 'same', 'or', 'they', 'only', 'do', 'wha
t', 's', 'new']
['to', 'me', 'the', 'real', 'real', 'solution', 'to', 'quality', 'growth', 'is', 'figuring', 'out', 'the', 'balance', 'between', 'two', 'act
ivities', 'exploration', 'and', 'exploitation']
```

Most Common word plot

주어진 pyplot 차트 코드를 활용하여 collections의 most_common함수를 사용하여 빈도가 높은 단어 histogram 가시화



- 데이터의 인덱스 [:10000] 까지 사용

- Word2Vec의 CBOW를 이용하여 100차원의 임베딩 학습 (min_count=10)

- 훈련한 인스턴스를 model_ted라 할 때, most_similar()를 이용하여 'man', 'computer'와 유사한 단어 출력.

- 재밌거나 놀라운 결과가 나온 가까운 이웃 단어들을 찾아 보세요

```python
model_ted.most_similar("man")
```

```
[('woman', 0.7737863063812256),
 ('guy', 0.7371513843536377),
 ('boy', 0.6724778413772583),
 ('girl', 0.6597930192947388),
 ('person', 0.6322830319404602),
 ('kid', 0.6013292670249939),
 ('soldier', 0.6011217832565308),
 ('lady', 0.5984901189804077),
 ('gentleman', 0.5743257999420166),
 ('husband', 0.5690660476684570)]
```

```python
model_ted.most_similar("computer")
```

```
[('software', 0.6310103535652161),
 ('3d', 0.5868310332298279),
 ('computers', 0.5724669694900513),
 ('simulation', 0.5515922904014587),
 ('machine', 0.5469782948493958),
 ('programmer', 0.5440416932106018),
 ('robot', 0.5341072678565979),
 ('microprocessor', 0.5305216908454895),
 ('screen', 0.5291713476181030),
 ('computing', 0.5286943912506104)]
```

T-SNE를 사용하여

Common words를 벡터차원축소

로 나타낸 그래프.

most common 단어 학습하여

출력하기