정보검색 과제 리포트

과제 #1

학번: B511074
이름: 박준형

1. 과제 개요

본 과제의 목적은 정보 검색의 자연어 처리의 시작으로, word2vec을 이해하고 익숙해지기 위함이다.

2. 결과

1)

```
In [1]: """
        1. Gensim 의 Text8Corpus를 이용하여 text8을 로드하고, 다음과 같이 처음 10 개의 단어를 출력하시오. 다음은 하나의 예시이다.
        """
        import gensim
        sentences = gensim.models.word2vec.Text8Corpus('../text8')

        for sentence in sentences:
            print(sentence[0:10])
            break

        ['anarchism', 'originated', 'as', 'a', 'term', 'of', 'abuse', 'first', 'used', 'against']
```

2)

```
In [2]: """
        2. text8을 입력으로 하는 gensim word2vec model을 생성하시오.
        """
        model = gensim.models.word2vec.Word2Vec(sentences)
```

3)

```
In [3]: """
        3. 매번 word vector를 계산하는 것은 낭비이므로, gensim에서는 학습된 model 을 save, load할 수 있도록 지원한다. 학습된 모델을 save 하고 다시 load하는 code를 작성하시오.
        """
        model.save("word2vec.model")
        model = gensim.models.word2vec.Word2Vec.load("word2vec.model")
```

4)

```
In [4]: """
        4. model.wv.vocab의 type()을 확인해보자. 결과는?
        """
        print(type(model.wv.vocab))

        <class 'dict'>
```

5)

In [5]:
```
"""
5. model.wv.vocab을 print해본다. 다음과 같이 나오면 된다.
"""
model.wv.vocab
```

```
{'anarchism': <gensim.models.keyedvectors.Vocab at 0x1f923555608>,
 'originated': <gensim.models.keyedvectors.Vocab at 0x1f91f636148>,
 'as': <gensim.models.keyedvectors.Vocab at 0x1f91f633f88>,
 'a': <gensim.models.keyedvectors.Vocab at 0x1f922710f08>,
 'term': <gensim.models.keyedvectors.Vocab at 0x1f922710f48>,
 'of': <gensim.models.keyedvectors.Vocab at 0x1f922710ec8>,
 'abuse': <gensim.models.keyedvectors.Vocab at 0x1f91f65e888>,
 'first': <gensim.models.keyedvectors.Vocab at 0x1f91f65e8c8>,
 'used': <gensim.models.keyedvectors.Vocab at 0x1f91f65e848>,
 'against': <gensim.models.keyedvectors.Vocab at 0x1f91f65e948>,
 'early': <gensim.models.keyedvectors.Vocab at 0x1f91f65e908>,
 'working': <gensim.models.keyedvectors.Vocab at 0x1f91f65e7c8>,
 'class': <gensim.models.keyedvectors.Vocab at 0x1f91f65e808>,
 'radicals': <gensim.models.keyedvectors.Vocab at 0x1f923688d88>,
 'including': <gensim.models.keyedvectors.Vocab at 0x1f923688688>,
 'the': <gensim.models.keyedvectors.Vocab at 0x1f9236883c8>,
 'diggers': <gensim.models.keyedvectors.Vocab at 0x1f923688fc8>,
 'english': <gensim.models.keyedvectors.Vocab at 0x1f923688b08>,
 'revolution': <gensim.models.keyedvectors.Vocab at 0x1f9236884c8>,
 'and': <gensim.models.keyedvectors.Vocab at 0x1f923688f48>,
 'sans': <gensim.models.keyedvectors.Vocab at 0x1f923688808>,
 'culottes': <gensim.models.keyedvectors.Vocab at 0x1f923688508>,
 'french': <gensim.models.keyedvectors.Vocab at 0x1f923688248>,
 'whilst': <gensim.models.keyedvectors.Vocab at 0x1f923688ec8>
```

6)

In [6]:
```
"""
6. word2vec 단어의 일부를 출력해보시오. 예를 들면 다음과 같은 결과가 나오 는지 확인해보시오.
"""
print(list(model.wv.vocab.keys())[0:10])
```

```
['anarchism', 'originated', 'as', 'a', 'term', 'of', 'abuse', 'first', 'used', 'against']
```

7)

In [7]:
```
"""
7. ' computer' 의 word vector를 구하시오.
"""
vector = model.wv['computer']
print(vector)
```

```
[-0.39353436 -2.062091    0.92765814 -1.1383324   0.24400587  2.6703663
  0.634406   -1.2897234  -0.82417274 -2.5269618   3.01074    -0.01057431
 -2.3544583   0.31918254 -1.4685227   1.0991782   0.24261948 -0.00859001
 -1.043389   -2.1066964   0.87861013 -0.4616209   0.34873617  1.7563388
 -0.25328872 -1.1134956   2.4735432   1.1629175  -1.5042294   1.0117351
  0.3421789   1.4060724  -0.8068057  -1.0882336  -2.9158986   0.26499528
  1.4291294   2.6122718  -1.1028486   0.7673002   0.4534042  -0.28792903
  0.04794642 -0.07771966  3.3747451   0.31758747  0.79575044  0.26001492
 -1.9949085  -0.8303062  -0.370904    0.1417987   2.796618    1.0395424
 -3.049073   -3.9500535  -1.4875815   0.8271246   1.4978257   0.7893205
 -0.7190449   2.5611157  -3.3905506  -0.06930465  0.4501821   1.2112226
  2.544447    2.886131    0.2692773   2.368938   -0.03418776  1.0590326
 -0.84781235  1.0286508   1.7985586   1.3528459  -1.3785769   2.0291693
  1.2116585   0.92252547 -1.6206882  -0.50595844 -1.6120863   0.1377912
  2.0434272   2.85897     0.62104607 -0.59769815 -2.7374735  -0.37242663
 -1.3696979  -0.53475267 -0.64247346  1.531241    2.358759    1.5945278
 -2.9548118  -1.9577565  -0.8501445  -0.18574308]
```

8)

In [8]:
```
"""
8. (Sanity check 또는 identity 의 확인) ' computer' 의 word vector로부터 가장 가까운 단어를 구하시오 (hint: similar by vector()를 이용하자.)
"""
model.wv.most_similar(['computer'], topn = 1)
```

```
[('computers', 0.7314187288284302)]
```

9)

In [9]:
```
"""
9. 이제, word vector들의 연산이 가능한지 알아보기 위해 다음 연산을 수행하고 결과를 말하시오.
most similar = woman + king - man most similar는 무엇인가?
"""
model.wv.most_similar(positive=['woman','king'], negative=['man'], topn = 1)
```

```
[('queen', 0.6842294931411743)]
```

10)

In [10]:
```
"""
10. (Extra) Reverse dictionary는 주어진 설명으로부터 단어를 찾아내는 기법이 다 [1]. 예를 들어, 앞에서 우리는 ' computer' 의 정의가 다음과 같음을 알 수 있었다.
 a machine for performing calculations automatically
 그렇다면, 반대로 이 설명으로부터 ' computer' 를 찾아낼 수 있을까? 이를 위해 몇가지 기법을 사용할 수 있다. 가장 단순한 모델은 설명을 이루는 각 단어들의 mean을 이용하는 것이다. 다음과 같이 질의를 만들어 보자.
"""
from nltk.corpus import wordnet as wn

def definition(word):
    return wn.synsets(word)[0].definition()

definition('computer')
```

```
'a machine for performing calculations automatically'
```

In [11]:
```
q = definition('computer')
model.wv.most_similar(positive=q.split(), topn=50)
```

```
[('instructions', 0.7046757340431213),
 ('device', 0.7040002346038818),
 ('checking', 0.7032796740531921),
 ('instruction', 0.6959731578826904),
 ('interrupts', 0.6587979793548584),
 ('application', 0.6581565737724304),
 ('circuitry', 0.6574883460998535),
 ('tasks', 0.6550710797309875),
 ('user', 0.6539882421493530),
 ('setup', 0.6518963840942383),
 ('switching', 0.6481111645698547),
 ('input', 0.6479632854461670),
```