SI630 Homework 2: Word2vec Vector Analysis

Important Note: Start this notebook only after you've gotten your word2vec model up and running!

Many NLP packages support working with word embeddings. In this notebook you can work through the various problems assigned in Task 3. We've provided the basic functionality for loading word vectors using [Gensim (https://radimrehurek.com/gensim/models/keyedvectors.html)](https://radimrehurek.com/gensim/models/keyedvectors.html), a good library for learning and using word vectors, and for working with the vectors.

One of the fun parts of word vectors is getting a sense of what they learned. Feel free to explore the vectors here!

In [1]:
```
from gensim.models import KeyedVectors
from gensim.test.utils import datapath
```

In [119]:
```
word_vectors = KeyedVectors.load_word2vec_format('model/word2vec_batch
```

In [120]:
```
word_vectors['the']
```

Out[120]:
```
array([-1.1471436 , -0.28772095, -2.2766628 ,  0.05391294, -1.4119608
,
       -0.29097795, -1.6112773 , -1.3050717 ,  1.815922  , -1.1087487
,
        0.9353696 , -0.8766251 , -0.82061183,  1.6959955 , -0.1036926
6,
       -0.20159033, -1.0378739 ,  0.7054374 , -1.3304659 , -0.3710644
8,
       -2.2896414 , -0.04715284, -0.5156441 ,  0.95758235, -0.0140381
5,
        0.97477394, -1.4969469 , -3.3509867 , -0.42322063, -0.8213046
,
       -0.9351953 ,  1.2764132 , -0.34828973, -0.0136232 ,  0.4795613
6,
        0.9739854 ,  1.2430013 , -0.52861917,  1.42831   , -0.8615355
,
        2.0485601 , -0.00539556, -0.7826375 ,  0.20810084,  0.0127139
9,
        1.938198  ,  1.3718727 ,  0.14091182, -2.4371796 ,  0.8924950
4],
      dtype=float32)
```

In [121]:
```
word_vectors.similar_by_word("books")
```

Out[121]:

```
[('articles', 0.9980189204216003),
 ('paintings', 0.9980115294456482),
 ('words', 0.9962870478630066),
 ('novels', 0.9962369799613953),
 ('portraits', 0.9962106943130493),
 ('material', 0.9956844449043274),
```

In [129]:
```
words=['crib','gin','stupid','motocross','england','victory','wonderfu

for word in words:
    print(word,word_vectors.similar_by_word(word)[0])
```

```
crib ('oxford_brookes_university', 0.9986768960952759)
gin ('bulger', 0.9968262314796448)
stupid ('remark', 0.9966140985488892)
motocross ('super_bowl', 0.9963929057121277)
england ('australia', 0.9930800795555115)
victory ('loss', 0.9942400455474854)
wonderful ('enormous', 0.9994308948516846)
teacher ('lab', 0.9973334074020386)
april ('march', 0.9998651146888733)
physics ('economics', 0.9949340224266052)
```

I picked 10 words of different frequencies. From the result, it could be seen that the result of prediction is not very well. For words like country name, months or subjects, the result is similar in category to the original word/ However, for words with less frequency, the result of prediction is worse

In [7]:
```
def get_analogy(a, b, c):
    return word_vectors.most_similar(positive=[b, c], negative=[a])[0]
```

In [174]:
```
print(get_analogy('sushi','japanese','pizza'))
print(get_analogy('math','physics','research'))
print(get_analogy('teacher','student','superior'))
print(get_analogy('literature', 'art', 'physics'))
print(get_analogy('man', 'woman', 'physician'))
```

```
italian
arts
staff
arts
biographer
```

The equations I got are:
japanese-sushi+piazza=italian
physics-math+research=arts
student-teacher+superior=staff
woman-man+physician=biographer
art-literature+physics=arts

I found that word analogies on words with the same part of speech are more likely to work, and analogies across different part of speech words are less effective. Since words of

subjects, food, country and jobs have the best prediction results, the analogies I made are
from these categories

In [9]:
```
import pandas as pd

df=pd.read_csv('word_pair_similarity_predictions.csv')
```

In [10]:
```
for idx,row in df.iterrows():
    word1=row[0]
    word2=row[1]
#     print(word1,word2)
    similarity=word_vectors.similarity(word1,word2)
    df['sim'][idx]=similarity
```

```
/Users/liuzihui/miniconda3/envs/python37/lib/python3.7/site-packages/
ipykernel_launcher.py:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/panda
s-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
(https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.htm
l#returning-a-view-versus-a-copy)
```

In [27]:
```
df.head()
```

Out[27]:

	word1	word2	sim
0	old	new	0.475980
1	smart	intelligent	0.970990
2	hard	difficult	0.850950
3	happy	cheerful	0.941494
4	hard	easy	0.884791

In [11]:
```
df.to_csv('word_pair_similarity_predictions.csv',index=False)
```

In [ ]: