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

In [ ]:
```
word_vectors = KeyedVectors.load_word2vec_format('target_embedding_layer.txt', binary=False)
```

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

Out[ ]:
```
array([-0.52905864,  0.2682494 ,  0.08240538,  0.6040402 , -0.40000755,
        0.40325102,  0.47333264,  0.12564887,  0.20180437, -0.14357038,
       -0.5900948 ,  0.31010377,  0.47124326, -0.28503367,  0.23567382,
        0.38805783, -0.00675848,  0.6721102 ,  0.24879739,  0.1021815 ,
        0.4804473 ,  0.5193118 , -0.13982862,  0.26506346,  0.20593485,
        0.03859131,  0.13175163, -0.43383372,  0.4473232 , -0.0037769 ,
       -0.25819823, -0.13163319,  0.5636755 , -0.3039646 ,  0.2620538 ,
        0.17377025,  0.2168366 ,  0.5943611 , -0.12959692,  0.42138287,
        0.08954721, -0.37473172, -0.26947808,  0.04299825,  0.08381032,
       -0.11034341,  0.13256617, -0.09809101,  0.4045905 ,  0.19616552],
      dtype=float32)
```

Problem 15

In [ ]:
```
word_vectors.similar_by_word("book")
```

Out[ ]:
```
[('series', 0.8369984030723572),
 ('one', 0.8337945342063904),
 ('novel', 0.7874147891998291),
 ('author', 0.7279027700424194),
 ('review', 0.7255702018737793),
 ('item', 0.725570023059845),
 ('story', 0.721172034740448),
 ('disappointed', 0.7175264358520508),
 ('price', 0.7053964138031006),
 ('ending', 0.692419171333313)]
```

In [ ]:
```
word_vectors.similar_by_word("son")
```

Out[ ]:
```
[('daughter', 0.7873197793960571),
 ('course', 0.7200912237167358),
 ('husband', 0.6906254291534424),
 ('money', 0.6792412400245667),
 ('interest', 0.6782873272895813),
 ('10', 0.6774787306785583),
 ('liking', 0.6729978322982788),
 ('structure', 0.6718193888664246),
 ('friends', 0.6632583737373352),
 ('kids', 0.6596392393112183)]
```

In [ ]:
```
word_vectors.similar_by_word("history")
```

Out[ ]:
```
[('events', 0.7585431933403015),
 ('part', 0.7572298049926758),
 ('last', 0.7551019191741943),
 ('period', 0.7526201009750366),
 ('beginning', 0.7491883635520935),
 ('perspective', 0.7470282912254333),
 ('premise', 0.7447119355201721),
 ('level', 0.7434907555580139),
 ('mystery', 0.7425572872161865),
 ('point', 0.7388588786125183)]
```

In [ ]:
```
word_vectors.similar_by_word("hand")
```

Out[ ]:
```
[('recipes', 0.7460935711860657),
 ('men', 0.7343083620071411),
 ('rest', 0.7195408940315247),
 ('works', 0.7159797549247742),
 ('students', 0.7137537598609924),
 ('spent', 0.7058455348014832),
 ('editing', 0.7005631327629089),
 ('ways', 0.6965265870094299),
 ('inside', 0.6964934468269348),
 ('attempts', 0.6962164044380188)]
```

In [ ]:
```
word_vectors.similar_by_word("time")
```

Out[ ]:
```
[('money', 0.7640765309333801),
 ('era', 0.7042155861854553),
 ('fault', 0.7006778717041016),
 ('game', 0.7000216245651245),
 ('while', 0.6908008456230164),
 ('copy', 0.6753082871437073),
 ('home', 0.6675803661346436),
 ('collection', 0.6673161387443542),
 ('heart', 0.6661427617073059),
 ('inside', 0.6577143669128418)]
```

In [ ]:
```
word_vectors.similar_by_word("target")
```

Out[ ]:
```
[('pretentious', 0.5812702178955078),
 ('drawn', 0.5461557507514954),
 ('subtle', 0.5405817031860352),
 ('dictated', 0.532189130783081),
 ('shameless', 0.5273035168647766),
 ('denial', 0.524050235748291),
 ('engage', 0.5140932202339172),
 ('outdated', 0.5114582777023315),
 ('language', 0.49841275811195374),
 ('Underwood', 0.4966738522052765)]
```

In [ ]:
```
word_vectors.similar_by_word("good")
```

Out[ ]:
```
[('great', 0.9033777713775635),
 ('nice', 0.8178068995475769),
 ('decent', 0.8010609745979309),
 ('funny', 0.797417163848877),
 ('short', 0.7970183491706848),
 ('fun', 0.788172721862793),
 ('useful', 0.7870458960533142),
 ('interesting', 0.785613477230072),
 ('informative', 0.7841373682022095),
 ('enjoyable', 0.7671247124671936)]
```

In [ ]:
```
word_vectors.similar_by_word("bad")
```

Out[ ]:
```
[('exciting', 0.7636231780052185),
 ('funny', 0.7555890083312988),
 ('good', 0.7522293329238892),
 ('simple', 0.7427698969841003),
 ('print', 0.7367860674858093),
 ('short', 0.7334396839141846),
 ('far', 0.7321767807006836),
 ('overall', 0.7297816872596741),
 ('movie', 0.7254855036735535),
 ('rushed', 0.723323404788971)]
```

In [ ]:
```
word_vectors.similar_by_word("humanistic")
```

Out[ ]:
```
[('Mattie', 0.6132054328918457),
 ('sucker', 0.5414050817489624),
 ('Inuit', 0.5228683352470398),
 ('by', 0.5099499821662903),
 ('cheesey', 0.5028645396232605),
 ('themes', 0.48187267780303955),
 ('homemade', 0.4800800681114197),
 ('overlay', 0.478593647480011),
 ('codependent', 0.477465957403183),
 ('Reeman', 0.4762853682041168)]
```

In [ ]:
```
word_vectors.similar_by_word("encyclopedia")
```

Out[ ]:
```
[('endure', 0.589384138584137),
 ('Force', 0.534084677696228),
 ('Serial', 0.5203403830528259),
 ('Nature', 0.5045533776283264),
 ('uk', 0.5043767094612122),
 ('Plenty', 0.4930706322193146),
 ('relato', 0.49204447865486145),
 ('Chief', 0.4882318079471588),
 ('rhythms', 0.48484084010124207),
 ('ultrasound', 0.48478803038597107)]
```

The training result seems to be different according to different target words chosen. I suspect that this is due to the size of the training data for each target word.

Problem 16

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

In [ ]:
```
get_analogy('man', 'woman', 'man')
```

Out[ ]:
```
'person'
```

In [ ]:
```
get_analogy('hostility', 'goodwill', 'enemy')
```

Out[ ]:
```
'narcissistic'
```

In [ ]:
```
get_analogy('sweet', 'truth', 'dream')
```

Out[ ]:
```
'portray'
```

In [ ]:
```
get_analogy('fake', 'true', 'history')
```

Out[ ]:
```
'real'
```

In [ ]:
```
get_analogy('moon', 'warmth', 'sun')
```

Out[ ]:
```
'century'
```

In [ ]:
```

```