Report on
Vector Semantics

Problem 1 :
Run the analogy test. Pick any **two** sets of pretrained embeddings, implement the analogy prediction method described in Equation 2, and compare their accuracies on the eight analogy tasks listed above. Make sure to mention the details of your selection in writing.

Problem Statement

Analogy prediction using word embeddings.

Data Used

Pretrained embeddings:

GoogleNews-vectors:
GoogleNews-vectors-negative300.bin.gz from
https://drive.google.com/file/d/0B7XkCwpI5KDYNlNUTTlSS21pQmM/edit
This repository hosts the word2vec pre-trained **Google News** corpus (3 billion running words) word vector model (3 million 300-dimension English word vectors).

GloVe:
I have downloaded pre-trained word vectors from the below link.
https://nlp.stanford.edu/projects/glove/

Test data:
http://www.fit.vutbr.cz/~imikolov/rnnlm/word-test.v1.txt
Test data contains 19,550 analogies. Aim is to predict the fourth word of the analogy using cosine similarity.

Methodology:

Word2Vec:

The word's representations in vector space. It requires lots and lots of data to train the model. Hence the model works best on google news corpus or GloVe or Wikipedia corpus as these are very large corpus. Here I have used google news and glove data.
I have used Gensim to find the vectors of the words. Before we can proceed to transform the words into a vector, we need to know the vocabulary of the English language. This means that we need to know each word in the English language. The dimension of the vector representing a document will be equal to the number of words in the vocabulary.
Eg:

Suppose if we are to predict the analogy for Athens Greece Bangkok

We will convert each word of the analogy into vector using google news data.

```
print(model['Athens'])
print(model['Greece'])
print(model['Bangkok'])
Athenes: [  2.65625000e-01  -6.25000000e-02   4.48608398e-03   1.03515625e-01
   2.89062500e-01  -9.32617188e-02  -1.01074219e-01….. …. .        .. ]

Greece: [ 0.47265625  0.00701904  0.05908203  0.09033203  0.03955078 -0.05859375
  0.06005859 -0.2578125  -0.24121094 -0.04589844 -0.19238281 -0.16015625 .  . ….]

Bangkok: [  1.60156250e-01  -1.25000000e-01  -1.13769531e-01   3.86718750e-01
  -1.46484375e-01  -2.18505859e-02  -2.32421875e-01  -1.74804688e-01..   …  ..]
```

Cosine similarity:

With cosine similarity we can measure the similarity between two document vectors. the cos(angle) will be 0 when the angle between vectors is such that the vectors are orthogonal and 1 if the vectors are similar.

Cos(Theta) = A . B/ ||A|| ||B||

Va (Athenes) = vector [2.65625000e-01 -6.25000000e-02 4.48608398e-03 1.03515625e-01 2.89062500e-01 -9.32617188e-02 -1.01074219e-01….. …. . ..]

Similarly ,

Vb = Vector of Greece

Bc = Vector of Bangkok

Now we will find cosine value of 3 vectors to predict the fourth word of the analogy.

Using .

Cos(vd , vb - va + vc) which will return the list of words nearest to these 3 words along with cosine value.

Find Nearest word using cosine similarity:

To predict the analogy we will find the nearest neighbours to the words and find the word with maximum cosine similarity by using Mikolov model as mention in below formula.

$$d = \arg\max_{d \in \mathcal{V} \setminus \{a,b,c\}} \cos\left(\mathbf{v}_d, \mathbf{v}_b - \mathbf{v}_a + \mathbf{v}_c\right).$$

-eq 1

Suppose we predicting analogy for

Athens Greece Bangkok Thailand

So our aim is to predict Thailand and complete the analogy.

[('Thailand', 0.9979326128959656), ('Thai', 0.902099072933197), ('Thailands', 0.8963393568992615), ('Chiang_Mai', 0.8866919279098511), ('Thais', 0.8770806789398193), ('Chavalit', 0.8728467226028442), ('Gen_Sonthi', 0.8636178374290466), ('Vejjajiva', 0.8617464303970337), ('Cambodia', 0.8585249781608582), ('Thaksin', 0.858080267906189)

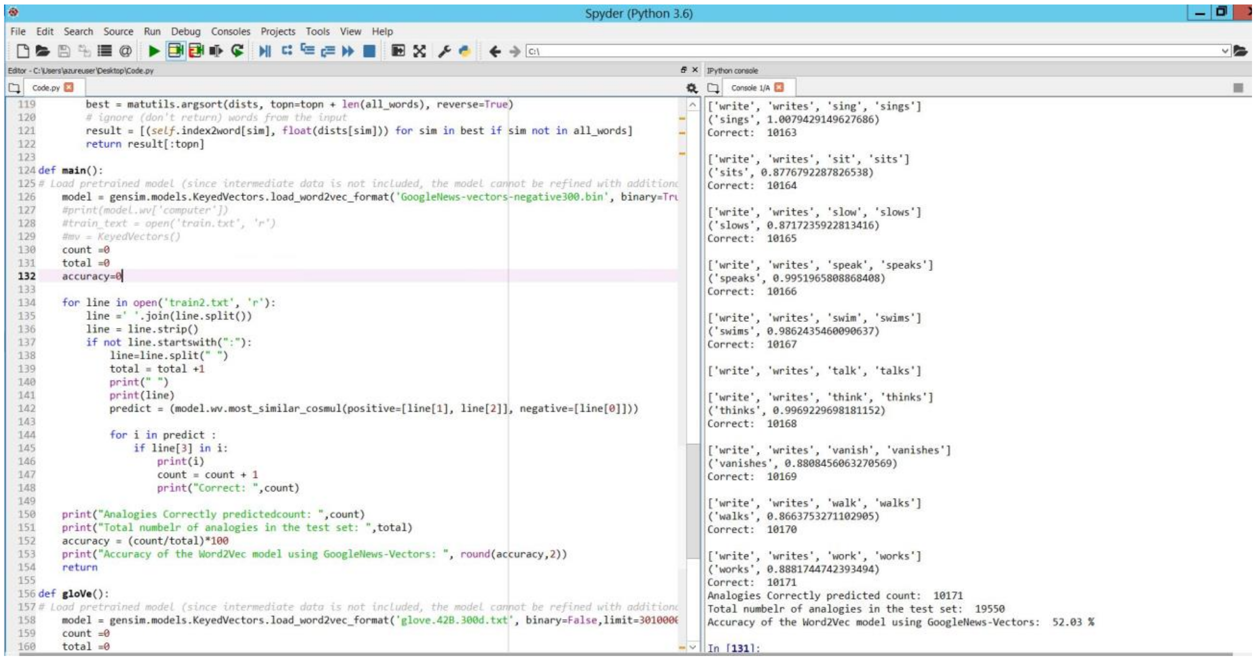
For the above list let's apply eq-1 and find the word with highest cosine similarity.
Hence model will predict "Thailand as fourth word to complete the analogy.

('Thailand', 0.9979326128959656)

<u>Challenges Faced</u>
Having huge corpus of Google news and GloVe calculating cosine similarity using vectors was computing intensive.

<u>Results:</u>

<u>Output using Google-news as input word embeddings:</u>


Spyder (Python 3.6)
File Edit Search Source Run Debug Consoles Projects Tools View Help
Editor - C:\Users\azureuser\Desktop\Code.py
Code.py
IPython console
Console 1/A

```
        best = matutils.argsort(dists, topn=topn + len(all_words), reverse=True)
        # ignore (don't return) words from the input
        result = [(self.index2word[sim], float(dists[sim])) for sim in best if sim not in all_words]
        return result[:topn]

def main():
# Load pretrained model (since intermediate data is not included, the model cannot be refined with additionc
    model = gensim.models.KeyedVectors.load_word2vec_format('GoogleNews-vectors-negative300.bin', binary=Tru
    #print(model.wv['computer'])
    #train_text = open('train.txt', 'r')
    #mv = KeyedVectors()
    count =0
    total =0
    accuracy=0

    for line in open('train2.txt', 'r'):
        line =' '.join(line.split())
        line = line.strip()
        if not line.startswith(":"):
            line=line.split(" ")
            total = total +1
            print(" ")
            print(line)
            predict = (model.wv.most_similar_cosmul(positive=[line[1], line[2]], negative=[line[0]]))

            for i in predict :
                if line[3] in i:
                    print(i)
                    count = count + 1
                    print("Correct: ",count)

    print("Analogies Correctly predictedcount: ",count)
    print("Total numbelr of analogies in the test set: ",total)
    accuracy = (count/total)*100
    print("Accuracy of the Word2Vec model using GoogleNews-Vectors: ", round(accuracy,2))
    return

def gloVe():
# Load pretrained model (since intermediate data is not included, the model cannot be refined with additionc
    model = gensim.models.KeyedVectors.load_word2vec_format('glove.42B.300d.txt', binary=False,limit=301000€
    count =0
    total =0
```

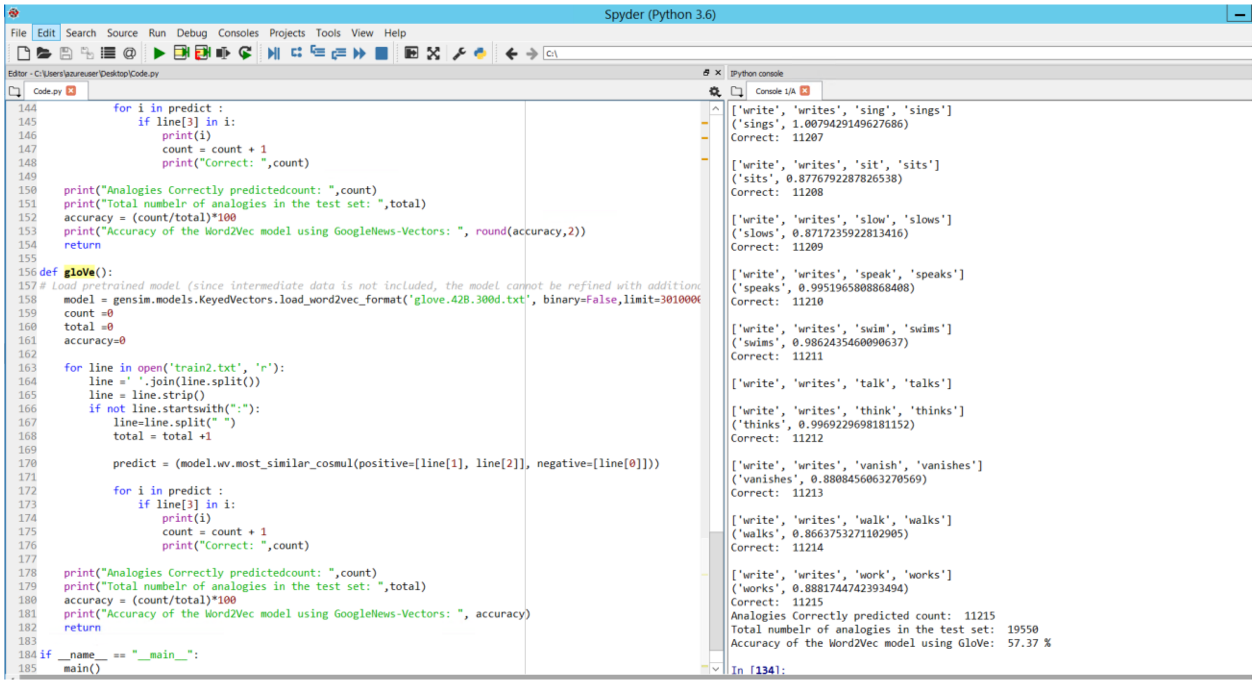
```
['write', 'writes', 'sing', 'sings']
('sings', 1.0079429149627686)
Correct:  10163

['write', 'writes', 'sit', 'sits']
('sits', 0.8776792287826538)
Correct:  10164

['write', 'writes', 'slow', 'slows']
('slows', 0.8717235922813416)
Correct:  10165

['write', 'writes', 'speak', 'speaks']
('speaks', 0.9951965808868408)
Correct:  10166

['write', 'writes', 'swim', 'swims']
('swims', 0.9862435460090637)
Correct:  10167

['write', 'writes', 'talk', 'talks']

['write', 'writes', 'think', 'thinks']
('thinks', 0.9969229698181152)
Correct:  10168

['write', 'writes', 'vanish', 'vanishes']
('vanishes', 0.8808456063270569)
Correct:  10169

['write', 'writes', 'walk', 'walks']
('walks', 0.8663753271102905)
Correct:  10170

['write', 'writes', 'work', 'works']
('works', 0.8881744742393494)
Correct:  10171
Analogies Correctly predicted count:  10171
Total numbelr of analogies in the test set:  19550
Accuracy of the Word2Vec model using GoogleNews-Vectors:  52.03 %

In [131]:
```

<u>Screenshot of Ouput</u>

Correct: 10171

Analogies Correctly predicted count: 10171
Total number of analogies in the test set: 19550
Accuracy of the Word2Vec model using GoogleNews-Vectors: 52.03 %

Output using glove as input word embeddings:

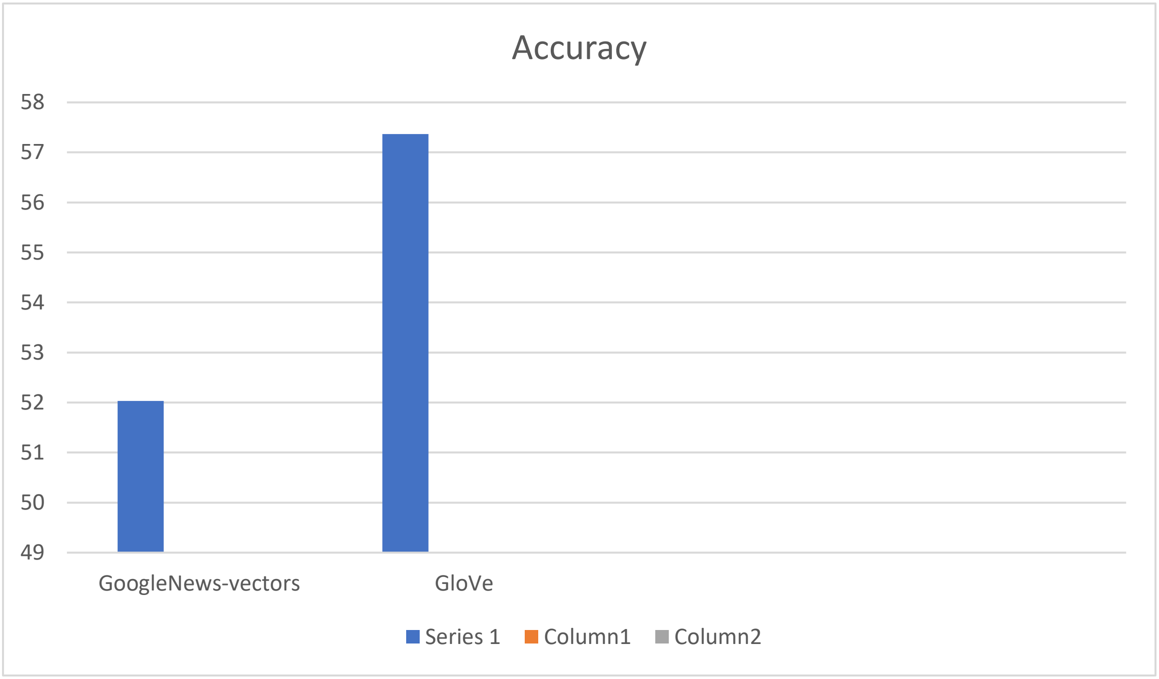

Spyder (Python 3.6)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\
Editor - C:\Users\azureuser\Desktop\Code.py
Code.py
IPython console
Console 1/A

```python
            for i in predict :
                if line[3] in i:
                    print(i)
                    count = count + 1
                    print("Correct: ",count)

    print("Analogies Correctly predictedcount: ",count)
    print("Total numbelr of analogies in the test set: ",total)
    accuracy = (count/total)*100
    print("Accuracy of the Word2Vec model using GoogleNews-Vectors: ", round(accuracy,2))
    return

def gloVe():
# Load pretrained model (since intermediate data is not included, the model cannot be refined with additiona
    model = gensim.models.KeyedVectors.load_word2vec_format('glove.42B.300d.txt', binary=False,limit=301000(
    count =0
    total =0
    accuracy=0

    for line in open('train2.txt', 'r'):
        line =' '.join(line.split())
        line = line.strip()
        if not line.startswith(":"):
            line=line.split(" ")
            total = total +1

            predict = (model.wv.most_similar_cosmul(positive=[line[1], line[2]], negative=[line[0]]))

            for i in predict :
                if line[3] in i:
                    print(i)
                    count = count + 1
                    print("Correct: ",count)

    print("Analogies Correctly predictedcount: ",count)
    print("Total numbelr of analogies in the test set: ",total)
    accuracy = (count/total)*100
    print("Accuracy of the Word2Vec model using GoogleNews-Vectors: ", accuracy)
    return

if __name__ == "__main__":
    main()
```

```
['write', 'writes', 'sing', 'sings']
('sings', 1.0079429149627686)
Correct:  11207

['write', 'writes', 'sit', 'sits']
('sits', 0.8776792287826538)
Correct:  11208

['write', 'writes', 'slow', 'slows']
('slows', 0.8717235922813416)
Correct:  11209

['write', 'writes', 'speak', 'speaks']
('speaks', 0.9951965808868408)
Correct:  11210

['write', 'writes', 'swim', 'swims']
('swims', 0.9862435460090637)
Correct:  11211

['write', 'writes', 'talk', 'talks']

['write', 'writes', 'think', 'thinks']
('thinks', 0.9969229698181152)
Correct:  11212

['write', 'writes', 'vanish', 'vanishes']
('vanishes', 0.8808456063270569)
Correct:  11213

['write', 'writes', 'walk', 'walks']
('walks', 0.8663753271102905)
Correct:  11214

['write', 'writes', 'work', 'works']
('works', 0.8881744742393494)
Correct:  11215
Analogies Correctly predicted count:  11215
Total numbelr of analogies in the test set:  19550
Accuracy of the Word2Vec model using GloVe:  57.37 %

In [134]:
```

<u>Screenshot of Ouput</u>

Analogies Correctly predicted count: 11215
Total number of analogies in the test set: 19550
Accuracy of the Word2Vec model using GloVe: 57.37 %



The results of the accuracy for different models are as follows. Accuracy is calculated by , the number of valid corrections, divided by the number of test analogies.

Conclusion:

As seen from the results above, Analogy prediction using Glove word embeddings proved to provide better accuracy that googleNews-vectors. However, there is some drawbacks of this model antonyms are placed near to each which actually has to be far apart.

Problem2:

Problem 2 : One known problem with word embeddings is that antonyms often have similar embeddings. You can verify this by searching for the top 10 most similar words to a few verbs like increase or enter that have clear antonyms (e.g., decrease and exit, respectively) using the cosine similarity. Discuss why embeddings might have this tendency.

Ans:

Because they can have similar contexts in many cases.
word embedding models based solely on the distributional hypothesis often place words improperly in vector spaces. For example, in a vector space, a word and its antonym should be sufficiently far apart, but they can be quite close because they can have similar contexts in many cases.

Adjusting word embeddings with semantic lexicons can help to solve this problem.
We adjust word vectors so that the cosine similarity between each word and its antonyms is zero or lower.
Also this could be solved by semantic intensity ordering, we should augment the synonym and antonym adjusted model with semantic intensity information to adjust word vectors.

Problem3: Design two new types of analogy tests that are not part of Mikolov's analogy dataset. You will create your own test questions (3 questions for each type, so in total 6 new questions). Report how well the two sets of embeddings perform on your test questions. You're encouraged to be adversarial so that the embeddings might get an accuracy of zero! Discuss any interesting observations you have made in the process.

- Below are the analogies that are not part of Mikolov's analogy dataset.

: opposite

1. [‘brain’,’ know’, ‘heart’,’ love’]
   Predicted word: tell

[('tell', 0.9291435480117798), ('think', 0.9014353156089783), ('knew', 0.8943924307823181), ('want', 0.8787474036216736), ('say', 0.8773857951164246), ('feel', 0.8746090531349182), ('understand', 0.8626413941383362), ('really', 0.8623691201210022), ('see', 0.8579029440879822), ('do', 0.854408860206604)]

2. [‘white’ ,’black’,’ in’, ‘out’]
   Predicted word: in the

[('inthe', 0.7972123026847839), ('where', 0.7907840013504028), ('the', 0.7845531702041626), ('during', 0.7625339031219482), ('In', 0.7495757341384888), ('since', 0.7490687370300293), ('at', 0.7484633922576904), ('from', 0.7442451119422913), ('outside', 0.742484450340271), ('for', 0.736226499080658)]

3. [‘branch’,’ tree’,’ tulip’,’ garden’]
   Predicted word: tulips

[('tulips', 1.1078016757965088), ('clematis_vine', 1.0723944902420044), ('trailing_arbutus', 1.0343390703201294), ('maple_sapling', 1.0328561067581177), ('hellebore', 1.0308610200881958), ('hackberry_trees', 1.0226091146469116), ('honeysuckles', 1.021685242652893), ('tulip_bulbs', 1.0195735692977905), ('Helleborus', 1.0194931030273438), ('dogwoods_azaleas', 1.0155656337738037)]

4. ['Day', 'squirrels', 'night', 'day']
   Predicted: bobcats

[('bobcats', 1.011855125427246), ('owls', 0.9969713687896729), ('owl', 0.9461898803710938), ('raccoons', 0.9454007148742676), ('cobras', 0.9383209347724915), ('pileated', 0.9367533326148987), ('rattlers', 0.9364383816719055), ('foxes', 0.9364168047904968), ('squirrel', 0.9317644238471985), ('sharptail', 0.929735004901886)]

: past-present-tense

5. ['ringing', 'rang', 'put', 'put']
   Predicted: came

[('came', 0.9302408695220947), ('went', 0.9198939800262451), ('took', 0.9021478891372681), ('puts', 0.890600323677063), ('pulled', 0.8886502981185913), ('Put', 0.8817678689956665), ('pushed', 0.8644694089889526), ('brought', 0.8586330413818359), ('putting', 0.8571400046348572), ('snuck', 0.851089358329773)]

6. ['play', 'played', 'choke', 'choked']
   Predicted word: cooked

[('cooked', 0.9450734853744507), ('cooks', 0.8634952306747437), ('accountant_paymaster_supply', 0.8521370887756348), ('cooking', 0.8266416192054749), ('baked_lasagna', 0.8213077187538147), ('stews', 0.8136942386627197), ('sautés', 0.8135122656822205), ('browned', 0.8130966424942017), ('cornmeal_porridge', 0.8106313943862915), ('pepper_steak', 0.8100829124450684)]
('cooked', 0.9450734853744507)

7. ['ringing', 'rang', 'raising', 'rose']
   Predicted word: rained

[('rained', 0.9438021183013916), ('snowing', 0.856504499912262), ('blowing_gale', 0.8113986849784851), ('went', 0.810787558555603), ('drizzling', 0.8049805760383606), ('sun_shone', 0.8041938543319702), ('sleeting', 0.7992990016937256), ('snowing_lightly',

```
0.798683226108551), ('came', 0.793958842754364), ('bucketing',
0.7903618216514587)]
('rained', 0.9438021183013916)
```

Predicted correctly: 0
Total: 7
Accuracy: 0

Some more:
Framework/Technologies:
Eclipse java pycharm python
Watsapp text gmail mail
Unix command windows gui
Firefox browser linux os

***END**