homework 1 report

Implementer : ati

inspired from Tonson, Todsavad , Teacher and Prof amanda

In [15]:
```
# Local Import
from modulenl import GloVe
from modulenl import SkipgramNegSampling
from modulenl import Skipgram
from modulenl import CBOW
from eval_wordy import *
from simi_corea import *
# Lib/ Framwork import

from nltk.corpus import brown
import numpy as np
import torch
import torch.nn as nn
import torch.optim as optim
import matplotlib.pyplot as plt
from collections import Counter
import math
from itertools import combinations_with_replacement
import pandas as pd
import re
import pickle
from gensim.test.utils import datapath
from gensim.models import KeyedVectors
from gensim.scripts.glove2word2vec import glove2word2vec
from sklearn.metrics import mean_squared_error as mNE
```

py script describe

- all model template can be found in `modulenl.py`.
- to preserve all index of vocab `data_opt_ati_brown.py` was once use and all train are using this same corpus and index.
- `eval data_pick.py` is used for getting the testset which we store in pickle file for ease of use.
- `eval wordy.py` store all function for calculate the accuracy.
- `simi_corea.py` store function to load similarity dataset and also a function for find a correation value

Corpus loading

In [2]:
```
with open('wordtotrain_use.atikeep','rb') as pic:
    corpus,vocab,word2index,index2word = pickle.load(pic)
flatten = lambda l: [item for sublist in l for item in sublist]
voc_size = len(vocab)
```

```
with open('data_to_test.atikeep','rb') as dic:
    test_set = pickle.load(dic)
```

Corpus component

- brown.sents(categories=['hobbies'])[:1500] From NLTK
  - To see more please visit `Train_PreTrain_datasorpus_gen_load.ipynb` or `data_opt_brown.py`

test set component

- family: 20
- gram2 Opposite : 20
- gram3 Comparative :20
- gram4 Superlative : 20
- gram8 Plural : 70

Total 150 test sample

- For more, please visit `eval_datapick.py`

Trained model weight loading

In [3]:
```
with open('myhobglove.atikeep','rb') as dic:
    gov_model = pickle.load(dic)

with open('myhob_C_Bro.atikeep','rb') as dic:
    cBro_model = pickle.load(dic)

with open('myhobskp_normal.atikeep','rb') as dic:
    skp_model = pickle.load(dic)

with open('myhobskpneg.atikeep','rb') as dic:
    neg_skp_model = pickle.load(dic)
```

Method

- Accuracy for both semantic and syntatic.
  - the semantic acc are calculate from 130 test sample with opposite, comparative,superlative and plural grammar.
  - On the other hand, semantic acc are calculate from 20 sample with the word relate to family.
- Similarity score test
  - we can calculate the correlation of 2 metrix with `Spearman correlation coefficient`

- though it give us corr from score -1,1 but our range is 1,10 so we perform normalization of correlation as follow eq.

$$ zi = (xi – min(x)) / (max(x) – min(x)) $$

Result

Glove result

```
Epoch: 1000 | cost: 16.179327 | time: -0.018680095672607422s
Epoch: 2000 | cost: 16.347530 | time: -0.017574310302734375s
Epoch: 3000 | cost: 1.635131 | time: -0.01890087127685547s
Epoch: 4000 | cost: 1.975415 | time: -0.033197164453552246s
Epoch: 5000 | cost: 3.769938 | time: -0.021063804626464844s
end loss = 3.769937515258789 + with time = 104.4947772026062
```

semantic and syntatic Accuracy

In [16]:
```
acc_gove = analogy_accuracy(test_set, vocab, get_embed_test, word2index, gov

print(f'Accuracy of Glove method with Brown corpus is {acc_gove*100}')
```

```
Accuracy of Glove method with Brown corpus is 0.0
```

Mean square error of similarity point

In [48]:
```
simi_score_gov = [[(call_score(get_embed_test(w1,word2index,gov_model),get_e
govmappe = mNE(simi_score_gov[-1],simi_score_gov[0])
print(govmappe)#
```

```
25.347199999999994
```

negative sampling skipgram

```
Epoch: 1000 | cost: 9.851048 | time: 0m 0s
Epoch: 2000 | cost: 8.355739 | time: 0m 0s
Epoch: 3000 | cost: 7.831168 | time: 0m 0s
Epoch: 4000 | cost: 10.446352 | time: 0m 0s
Epoch: 5000 | cost: 8.289867 | time: 0m 0s
```

```
ESt time = 8 min 3.5 sec
```

semantic and syntatic acc

In [6]:
```
acc_neg = analogy_accuracy(test_set, vocab, get_embed_test, word2index, neg_

print(f'Accuracy of SkipGram with negative sampling method with Brown corpus
```

```
Accuracy of SkipGram with negative sampling method with Brown corpus is 0.0
```

Mean square error of similarity point

In [49]:
```
simi_score_neggram = [[(call_score(get_embed_test(w1,word2index,neg_skp_mode
negmappe = mNE(simi_score_neggram[-1] ,simi_score_neggram[0])
```

```
print(negmappe)
```

```
75.34719999999999
```

SkipGram

```
Epoch: 1000 | cost: 8.582491 | time: 0m 0s
Epoch: 2000 | cost: 9.056213 | time: 0m 0s
Epoch: 3000 | cost: 9.070553 | time: 0m 0s
Epoch: 4000 | cost: 10.536556 | time: 0m 0s
Epoch: 5000 | cost: 8.487428 | time: 0m 0s
```

Est time = 7 min 39 sec

semantic and syntatic acc

In [7]:
```
acc_skp = analogy_accuracy(test_set, vocab, get_embed_test, word2index, skp_

print(f'Accuracy of skipGram method with Brown corpus is {acc_skp*100}')
```

```
Accuracy of skipGram method with Brown corpus is 0.0
```

Mean square error of similarity point

In [51]:
```
simi_score_skp = [[(call_score(get_embed_test(w1,word2index,skp_model),get_e
skpmappe = mNE(simi_score_skp[-1],simi_score_skp[0])
print(skpmappe)#
```

```
25.347199999999994
```

CBow

```
Epoch: 100 | cost: 11.299652 | time: 17.72466206550598
Epoch: 200 | cost: 10.979570 | time: 35.26954507827759
Epoch: 300 | cost: 11.599943 | time: 55.95752930641174
Epoch: 400 | cost: 11.446724 | time: 75.82228302955627
Epoch: 500 | cost: 11.278444 | time: 94.97708511352539
EstTime = 1 min 36 sec**
```

semantic and syntatic acc

In [8]:
```
acc_cBro = analogy_accuracy(test_set, vocab, get_embed_test_c_Bro, word2inde

print(f'Accuracy of CBOW method with Brown corpus is {acc_cBro*100}')
```

```
Accuracy of CBOW method with Brown corpus is 0.0
```

Mean square error of similarity point

In [54]:
```
simi_score_cbro = [[(call_score(get_embed_test_c_Bro(w1,word2index,cBro_mode
cbromappe = mNE(simi_score_cbro[-1],simi_score_cbro[0])
print(cbromappe)#
```

```
75.34719999999999
```

gensim trained version

In [4]:
```
#you have to put this file in some python/gensim directory; just run it and
glove_file = datapath('glove.6B.50d.txt')
model = KeyedVectors.load_word2vec_format(glove_file, binary=False, no_heade
```

In [5]:
```
predict = model.most_similar(positive=['bird', 'bananas'], negative=['banana
# banana bananas bird birds
print(predict)
```

```
birds
```

In [7]:
```
genss,cateCorr = analogy_accuracy_(test_set,model)

print(f'accuracy by categories = {np.array(cateCorr)/ np.array([20,20,20,20,
print(f'Accuracy of Gensim pretrian model is {genss*100}')
```

```
Count by categories [13, 1, 6, 5, 47]
accuracy by categories = [65.          5.         30.         25.         6
7.14285714]
Accuracy of Gensim pretrian model is 48.0
```

All categories for test

- family
- gram2 Opposite
- gram3 Comparative
- gram4 Superlative
- gram8 Plural

Result

| model method | time | all accuracy | Syntactic acc | Semantic accuracy | Similarity Score Error | is_Learn | |----------------|--------------|-------------|---------------|--------------------|------------------------|----------| | GloVe | 1 min 45 sec | 0 | 0 | 0 | 25.347199999999994 | Yes | | NegsamskipGram | 8 min 3.5 sec | 0 | 0 | 0 | 75.34719999999999 | No | | SkipGram | 7 min 39 sec | 0 | 0 | 0 | 25.347199999999994 | No | | CBOW | 16 min ** | 0 | 0 | 0 | 75.34719999999999 | No | | Gensim_Pretrain | - | 48 | 45.38 | 65 | - | - |

- ** CBOW training time are calculate by forward estimation 10 fold since I did not wait to get the result
- *2 Accuracy was calculate by `most_similar` method to match the test set that TA.Amanda provided
  - Method of accuracy is using the top 1 similar check by match.
- To shorten to training time, `windowsize =1` is used for all build from stratch
- embsize = 2 for our 4 model
- Negative sampling No. = 3
- **Important note**

    - the semantic acc are calculate from 130 test sample with opposite, comparative,superlative and plural grammar.
    - On the other hand, semantic acc are calculate from 20 sample with the word relate to family.
- Since the train dataset is quite small so we do cheate a bit on pick some easy test category.

Inference

- accuracy and similarity score
    - we observe 0 accuracy from training with a small size of corpus (~4000 sentence) and really bad error in similarity test
    - since we use small corpus, small window size , and personally, small embed size (= 2) result are not good.
- Training time are fastest on glove and CBOW is the slowest

Extra Ref

R. Řehůřek and P. Sojka, ‘Software Framework for Topic Modelling with Large Corpora’, in Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks, 2010, pp. 45–50.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation