```
NAME = "Yinglu Deng"
```

Lab 6: Skip Gram

Please read the following instructions very carefully

Working on the assignment / FAQs

- **Always use the seed/random_state as *42* wherever applicable** (This is to ensure repeatability in answers, across students and coding environments)
- The type of question and the points they carry are indicated in each question cell
- To avoid any ambiguity, each question also specifies what *value* must be set. Note that these are dummy values and not the answers
- If an autograded question has multiple answers (due to differences in handling NaNs, zeros etc.), all answers will be considered.
- You can delete the `raise NotImplementedError()`
- **Submitting the assignment** : Download the '.ipynb' file from Colab and upload it to bcourses. Do not delete any outputs from cells before submitting.
- That's about it. Happy coding!

Available software:

- Python's Gensim module: https://radimrehurek.com/gensim/ (install using pip)
- Sklearn's TSNE module in case you use TSNE to reduce dimension (optional)
- Python's Matplotlib (optional)

Note: The most important hyper parameters of skip-gram/CBOW are vector size and windows size

```
!pip install gensim
!wget -nc https://s3.amazonaws.com/dl4j-distribution/GoogleNews-vectors-negative300.bi
```

```
import pandas as pd
import numpy as np
import gensim
```

```
Requirement already satisfied: gensim in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: smart-open>=1.2.1 in /usr/local/lib/python3.7/dis
Requirement already satisfied: six>=1.5.0 in /usr/local/lib/python3.7/dist-packa
```

```
Requirement already satisfied: numpy>=1.11.3 in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: scipy>=0.18.1 in /usr/local/lib/python3.7/dist-pac
--2021-10-26 22:48:26--  https://s3.amazonaws.com/dl4j-distribution/GoogleNews-ve
Resolving s3.amazonaws.com (s3.amazonaws.com)... 52.217.66.70
Connecting to s3.amazonaws.com (s3.amazonaws.com)|52.217.66.70|:443... connected
HTTP request sent, awaiting response... 200 OK
Length: 1647046227 (1.5G) [application/x-gzip]
Saving to: 'GoogleNews-vectors-negative300.bin.gz'

GoogleNews-vectors- 100%[===================>]   1.53G  90.8MB/s    in 17s

2021-10-26 22:48:42 (95.0 MB/s) - 'GoogleNews-vectors-negative300.bin.gz' saved
```

```
from gensim.models import KeyedVectors

model = KeyedVectors.load_word2vec_format('GoogleNews-vectors-negative300.bin.gz', bin


word_vectors = model.wv
word_vectors.vocab
```

```
 'homes': <gensim.models.keyedvectors.Vocab at 0x7f22890b45d0>,
 'stage': <gensim.models.keyedvectors.Vocab at 0x7f22890b4650>,

 'department': <gensim.models.keyedvectors.Vocab at 0x7f22890b4710>,
 'named': <gensim.models.keyedvectors.Vocab at 0x7f22890b4750>,
 'earnings': <gensim.models.keyedvectors.Vocab at 0x7f22890b47d0>,
 'offers': <gensim.models.keyedvectors.Vocab at 0x7f22890b4850>,
 'star': <gensim.models.keyedvectors.Vocab at 0x7f22890b4910>,
 'certain': <gensim.models.keyedvectors.Vocab at 0x7f22890b4950>,
 'double': <gensim.models.keyedvectors.Vocab at 0x7f22890b49d0>,
 'longer': <gensim.models.keyedvectors.Vocab at 0x7f22890b4a50>,
 'followed': <gensim.models.keyedvectors.Vocab at 0x7f22890b4ad0>,
 'cause': <gensim.models.keyedvectors.Vocab at 0x7f22890b4b50>,
 'Association': <gensim.models.keyedvectors.Vocab at 0x7f22890b4c10>,
 'signed': <gensim.models.keyedvectors.Vocab at 0x7f22890b4c50>,
 'committee': <gensim.models.keyedvectors.Vocab at 0x7f22890b4d10>,
 'hour': <gensim.models.keyedvectors.Vocab at 0x7f22890b4d90>,
 'college': <gensim.models.keyedvectors.Vocab at 0x7f22890b4dd0>,
 'Pakistan': <gensim.models.keyedvectors.Vocab at 0x7f22890b4e50>,
 'users': <gensim.models.keyedvectors.Vocab at 0x7f22890b4ed0>,
 'Iran': <gensim.models.keyedvectors.Vocab at 0x7f22890b4f90>,
 'sign': <gensim.models.keyedvectors.Vocab at 0x7f22890b7050>,
 'living': <gensim.models.keyedvectors.Vocab at 0x7f22890b7090>,
 'failed': <gensim.models.keyedvectors.Vocab at 0x7f22890b7110>,
 'reach': <gensim.models.keyedvectors.Vocab at 0x7f22890b7190>,
 'quickly': <gensim.models.keyedvectors.Vocab at 0x7f22890b7210>,
 'receive': <gensim.models.keyedvectors.Vocab at 0x7f22890b7290>,
 'debt': <gensim.models.keyedvectors.Vocab at 0x7f22890b7350>,
 'sale': <gensim.models.keyedvectors.Vocab at 0x7f22890b73d0>,
 'Board': <gensim.models.keyedvectors.Vocab at 0x7f22890b7410>,
 'Americans': <gensim.models.keyedvectors.Vocab at 0x7f22890b74d0>,
 'Road': <gensim.models.keyedvectors.Vocab at 0x7f22890b7550>,
 'Brown': <gensim.models.keyedvectors.Vocab at 0x7f22890b7590>,
 'insurance': <gensim.models.keyedvectors.Vocab at 0x7f22890b7650>,
 '##:##': <gensim.models.keyedvectors.Vocab at 0x7f22890b7690>,
 'anyone': <gensim.models.keyedvectors.Vocab at 0x7f22890b7710>
```

```
 'anyone': <gensim.models.keyedvectors.Vocab at 0x7f22890b7710>,
 'tournament': <gensim.models.keyedvectors.Vocab at 0x7f22890b77d0>,
 'More': <gensim.models.keyedvectors.Vocab at 0x7f22890b7850>,
 'gas': <gensim.models.keyedvectors.Vocab at 0x7f22890b78d0>,
 'talks': <gensim.models.keyedvectors.Vocab at 0x7f22890b7910>,
 'serious': <gensim.models.keyedvectors.Vocab at 0x7f22890b7990>,
 'required': <gensim.models.keyedvectors.Vocab at 0x7f22890b7a10>,
 'sell': <gensim.models.keyedvectors.Vocab at 0x7f22890b7ad0>,
 'construction': <gensim.models.keyedvectors.Vocab at 0x7f22890b7b50>,
 'evidence': <gensim.models.keyedvectors.Vocab at 0x7f22890b7b90>,
 'remains': <gensim.models.keyedvectors.Vocab at 0x7f22890b7c10>,
 'black': <gensim.models.keyedvectors.Vocab at 0x7f22890b7c90>,
 'below': <gensim.models.keyedvectors.Vocab at 0x7f22890b7d10>,
 'improve': <gensim.models.keyedvectors.Vocab at 0x7f22890b7d90>,
 'crisis': <gensim.models.keyedvectors.Vocab at 0x7f22890b7e10>,
 'address': <gensim.models.keyedvectors.Vocab at 0x7f22890b7e90>,
 'questions': <gensim.models.keyedvectors.Vocab at 0x7f22890b7f50>,
 'easy': <gensim.models.keyedvectors.Vocab at 0x7f22890b7fd0>,
 'begin': <gensim.models.keyedvectors.Vocab at 0x7f22890b9050>,
 'view': <gensim.models.keyedvectors.Vocab at 0x7f22890b9110>,
 'School': <gensim.models.keyedvectors.Vocab at 0x7f22890b9150>,
 'heard': <gensim.models.keyedvectors.Vocab at 0x7f22890b91d0>,
 'executive': <gensim.models.keyedvectors.Vocab at 0x7f22890b9290>,
 'raised': <gensim.models.keyedvectors.Vocab at 0x7f22890b92d0>,
 ...}
```

Q1 (1 point)

Find the cosine similarity between the following word pairs

- (France, England)
- (smaller, bigger)
- (England, London)
- (France, Rocket)
- (big, bigger)

```
model.most_similar(positive=['woman', 'king'], negative=['man'])
```

```
[('queen', 0.7118192911148071),
 ('monarch', 0.6189674139022827),
 ('princess', 0.5902431011199951),
 ('crown_prince', 0.5499460697174072),
 ('prince', 0.5377321243286133),
 ('kings', 0.5236844420433044),
 ('Queen_Consort', 0.5235945582389832),
 ('queens', 0.518113374710083),
 ('sultan', 0.5098593235015869),
 ('monarchy', 0.5087411999702454)]
```

```
word_pairs = [
              ('France', 'England') ,
              ('smaller', 'bigger'),
```

```
              ('England', 'London'),
              ('France', 'Rocket'),
              ('big', 'bigger')
            ]
for pair in word_pairs:
    print(model.similarity(pair[0], pair[1]))
```

```
0.39804944
0.7302272
0.43992856
0.07114174
0.68423855
```

```
#Replace 0 with the code / value; Do not delete this cell
similarity_pair1 = 0.39804944
similarity_pair2 = 0.7302272
similarity_pair3 = 0.43992856
similarity_pair4 = 0.07114174
similarity_pair5 = 0.68423855
```

```
#This is an autograded cell, do not edit/delete
print(similarity_pair1, similarity_pair2, similarity_pair3, similarity_pair4, similari
```

```
0.39804944 0.7302272 0.43992856 0.07114174 0.68423855
```

Q2 (1 point)

Write an expression to extract the vector representations of the words:

- France
- England
- smaller
- bigger
- rocket
- big

Get only the first 5 elements for each vector representation.

```
words = ['France', 'England', 'smaller', 'bigger', 'rocket', 'big']
for word in words:
  print(len(model[word]), model[word])
```

```
    -1.62353516e-02 -2.24609375e-01  1.99218750e-01 -4.80468750e-01
     1.36718750e-01  6.98852539e-03 -1.40380859e-02  2.91015625e-01
     3.18359375e-01 -1.13769531e-01 -1.54296875e-01 -8.64257812e-02
    -1.00097656e-02  4.02832031e-02  5.68847656e-02  8.30078125e-03
     6.15234375e-02  1.28906250e-01  1.47460938e-01  5.29785156e-02
```

```
  6.15234375e-02 -1.28906250e-01 -1.47460938e-01 -5.29785156e-02
 -1.84570312e-01  6.68945312e-02  5.78613281e-02  2.24609375e-01
  9.42382812e-02  5.15625000e-01  1.70898438e-02 -8.69140625e-02

  1.80664062e-01 -2.92968750e-01  6.88476562e-02  4.04296875e-01
  2.47070312e-01  1.25976562e-01  1.01562500e-01  2.47070312e-01
 -3.08593750e-01 -7.56835938e-02  5.24902344e-02 -5.51757812e-02
 -6.83593750e-03  1.56250000e-01 -3.90625000e-01 -1.59179688e-01]
300 [ 0.11132812  0.10595703 -0.07373047  0.18847656  0.07666016 -0.3828125
 -0.0625     -0.07470703  0.05957031  0.22167969  0.20507812 -0.09228516
  0.05395508  0.01379395 -0.16992188  0.05493164  0.09619141  0.06103516
 -0.14160156  0.03173828 -0.08642578  0.12011719  0.06445312  0.22070312
  0.06835938  0.04956055 -0.22460938 -0.06298828  0.09179688 -0.00531006
 -0.11425781  0.20605469  0.31054688 -0.0625     -0.02026367 -0.13476562
 -0.02697754  0.2734375   0.27929688  0.21386719  0.25195312 -0.13964844
  0.19824219 -0.07421875  0.09228516  0.125       0.0612793  -0.02990723
  0.0072937  -0.05615234 -0.08447266  0.1796875  -0.17578125 -0.11328125
 -0.17578125 -0.1171875   0.09082031 -0.07177734  0.30273438 -0.2734375
 -0.07128906  0.33007812 -0.13574219 -0.0390625   0.01397705 -0.02526855
  0.05981445  0.14550781 -0.11035156  0.12988281  0.12695312 -0.04980469
  0.16992188  0.18261719 -0.23144531  0.07910156 -0.06738281  0.34960938
 -0.07324219 -0.03442383  0.01507568 -0.05957031 -0.07373047 -0.03857422
 -0.06542969 -0.2578125  -0.05151367  0.08154297  0.08740234  0.01318359
  0.06640625  0.06152344 -0.20800781 -0.10253906 -0.12158203 -0.06152344
  0.1484375   0.06005859 -0.19140625 -0.05761719 -0.03149414  0.12255859
  0.19628906 -0.24902344  0.03735352  0.08056641  0.13183594 -0.01977539
  0.19238281 -0.109375   -0.02172852 -0.09912109 -0.00793457 -0.01251221
  0.23144531 -0.01080322  0.00234985  0.11279297 -0.00485229  0.01373291
 -0.02441406  0.01123047 -0.22460938  0.14160156 -0.00744629 -0.01385498
  0.20996094  0.05712891  0.00366211 -0.07617188 -0.05859375 -0.05957031
  0.00613403  0.13183594  0.04003906 -0.11376953  0.046875    0.04199219
 -0.0088501  -0.01672363  0.03173828 -0.08398438  0.16308594 -0.17285156
 -0.00866699 -0.04443359  0.10595703  0.0145874   0.04492188 -0.07568359
 -0.06689453 -0.0050354  -0.10986328  0.11572266 -0.10107422  0.16210938
 -0.0144043  -0.140625   -0.09326172 -0.02502441 -0.13867188  0.04760742
 -0.01452637 -0.10986328  0.31640625 -0.11914062  0.08203125 -0.22949219
  0.09472656 -0.0324707  -0.18847656 -0.10986328  0.22949219 -0.22363281
 -0.07080078  0.05053711  0.20507812  0.03979492 -0.03881836  0.11962891
  0.01818848 -0.2421875   0.10791016 -0.02307129 -0.06982422  0.05737305
 -0.15527344  0.01989746 -0.01153564 -0.04833984  0.16210938  0.01916504
  0.06079102  0.01177979 -0.07714844 -0.02380371 -0.10693359 -0.06542969
 -0.20214844 -0.07373047  0.25390625  0.12353516  0.09130859 -0.1484375
  0.03662109  0.02294922  0.09375    -0.13183594  0.03857422  0.13378906
 -0.13183594  0.30859375 -0.0255127   0.07763672 -0.09033203  0.00350952
  0.32617188 -0.08203125 -0.01831055  0.01879883 -0.0390625   0.17578125
 -0.05273438 -0.22070312 -0.10302734  0.0390625  -0.00315857 -0.06176758
 -0.01397705 -0.04003906 -0.17773438  0.20507812  0.05004883  0.10595703
 -0.00170898 -0.04223633 -0.02478027 -0.28320312 -0.08789062 -0.08349609
  0.12792969  0.04931641 -0.16503906 -0.07275391  0.00405884 -0.23828125
  0.16992188 -0.06689453  0.07958984 -0.14160156  0.01287842  0.1640625
  0.24511719  0.09570312  0.18261719 -0.00179291 -0.00714111  0.20507812
 -0.40429688 -0.01940918 -0.07373047 -0.16113281 -0.20800781  0.00860596
 -0.08642578  0.0189209  -0.08642578  0.03930664  0.03564453  0.14550781
 -0.01806641  0.14746094  0.10888672 -0.06982422 -0.02209473 -0.01361084
 -0.03039551 -0.13574219 -0.0625      0.04003906 -0.13964844  0.02026367
 -0.12597656  0.0703125  -0.03015137  0.11376953  0.00337219 -0.06347656
```

```
#Replace 0 with the code / value to get the first 5 elements of each vector; Do not de
```

```
vector_1 = [4.85839844e-02, 7.86132812e-02, 3.24218750e-01, 3.49121094e-02, 7.7148437
vector_2 = [-1.98242188e-01, 1.15234375e-01, 6.25000000e-02, -5.83496094e-02, 2.26562
vector_3 = [-0.05004883, 0.03417969, -0.0703125, 0.17578125, 0.00689697]
vector_4 = [-6.54296875e-02, -9.52148438e-02, -6.22558594e-02, 1.62109375e-01, 1.98974
vector_5 = [-3.19824219e-02, 2.71484375e-01, -2.89062500e-01, -1.54296875e-01, 1.68945
vector_6 = [0.11132812, 0.10595703, -0.07373047, 0.18847656, 0.07666016]


#This is an autograded cell, do not edit/delete
print(vector_1)
print(vector_2)
print(vector_3)
print(vector_4)
print(vector_5)
print(vector_6)
```

```
[0.0485839844, 0.0786132812, 0.32421875, 0.0349121094, 0.0771484375]
[-0.198242188, 0.115234375, 0.0625, -0.0583496094, 0.2265625]
[-0.05004883, 0.03417969, -0.0703125, 0.17578125, 0.00689697]
[-0.0654296875, -0.0952148438, -0.0622558594, 0.162109375, 0.0198974609]
[-0.0319824219, 0.271484375, -0.2890625, -0.154296875, 0.168945312]
[0.11132812, 0.10595703, -0.07373047, 0.18847656, 0.07666016]
```

Q3 (1 point)

Find the euclidean distances between the word pairs :

- (France, England)
- (smaller, bigger)
- (England, London)
- (France, Rocket)
- (big, bigger)

```
for pair in word_pairs:
    print(np.linalg.norm(model[pair[0]] - model[pair[1]]))
```

```
3.0151067
1.8618743
2.8752837
3.892071
1.9586496
```

```
#Replace 0 with the code / value; Do not delete this cell
eu_dist1 = 3.0151067
eu_dist2 = 1.8618743
eu_dist3 = 2.8752837
eu_dist4 = 3.892071
eu_dist5 = 1.9586496
```

```
#This is an autograded cell, do not edit / delete
print(eu_dist1)
print(eu_dist2)
print(eu_dist3)
print(eu_dist4)
print(eu_dist5)
```

```
3.0151067
1.8618743
2.8752837
3.892071
1.9586496
```

Q4 (1 point)

Time to dabble with the power of Word2Vec. Find the 2 closest words for the following conditions:

- (King - Man + Queen)
- (bigger - big + small)
- (waiting - wait + run)
- (Texas + Milwaukee − Wisconsin)

```
#Replace 0 with the code / value; Do not delete this cell
closest1 = model.most_similar(positive=['King', 'Queen'], negative=['Man'], topn=2)
closest2 = model.most_similar(positive=['bigger', 'small'], negative=['big'], topn=2)
closest3 = model.most_similar(positive=['waiting', 'run'], negative=['wait'], topn=2)
closest4 = model.most_similar(positive=['Texas', 'Milwaukee'], negative=['Wisconsin'],
closest5 = 0
```

```
#This is an autograded cell, do not edit/delete
print(closest1)
print(closest2)
print(closest3)
print(closest4)
print(closest5)
```

```
[('Queen_Elizabeth', 0.5257916450500488), ('monarch', 0.5004087090492249)]
[('larger', 0.7402471899986267), ('smaller', 0.732999324798584)]
[('running', 0.5654535889625549), ('runs', 0.49640005826950073)]
[('Houston', 0.7767744064331055), ('Fort_Worth', 0.7270511388778687)]
0
```

Q5 (3 points)

Using the vectors for the words in the Google News dataset, explore the semantic representation of these words through K-means clustering and explain your findings.

Note : Since there are ~3Mil words in the vocabulary, you can downsample it to ~20-30k randomly selected words

Do not delete the below cell

```
import random
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
from sklearn.decomposition import PCA
word = list(model.wv.vocab)

random.seed(42)
num = random.randint(20000, 30000)
word_selected = word[:num]  #selected 20k words
# word_selected
df = pd.DataFrame([model[w] for w in word_selected])
df['word'] = word_selected
# df.head()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:5: DeprecationWarni
  """
```

```
X = df.set_index('word')
kmeans = KMeans(n_clusters = 120, random_state = 42).fit(X)
centroids, labels = kmeans.cluster_centers_, kmeans.labels_
# print(centroids)
print(labels)
```

```
[94 94 94 ... 94  4 68]
```

```
index = 0
index_list = []
for i in labels:
  if i == 12:
    index_list.append(index)
  index += 1


len(index_list)
```

```
294
```

```
X_copy = X.reset_index()
for i in index_list:
  print(X_copy.word[i])
```

```
adamant
outspoken
proclaimed
advocated
underlined
reaffirmed
emphasizing
outlining
accusation
depicted
clarified
intervened
reacting
insistence
contemplated
mentions
lamented
spelled
doubted
boasted
purported
conveyed
cautions
mentioning
testifying
dismissing
construed
mindful
recounted
imply
summed
contending
endorsing
Referring
condemning
likened
championed
touting
risked
downplayed
underestimated
regretted
referenced
reassured
asserts
concedes

assertions
reckons
affirming
cooperated
resorted
disagrees
invoked
implication
```

```
implication
asserting
echoing
balked
apologizing
implying
```

I did the k-mean clustering to the dataset and created 120 clustering. We picked cluster 12 as an example, there are 294 total words labeled 12, the cluster 12 included "said", "says", "say", "told", "added", "called", "according", "announced", "reported" and so on. They are all verbs with any tense (past tense, present tense etc.) which shows a person's actions. These words share a same or similar feature and meaning for this cluster.

Q6 (1 point)

What loss function does the skipgram model use and briefly describe what this function is minimizing.

Do not delete the below cell

1. Cross-Entropy Loss is used in skipgram model, because we treat this as a classification problem. It is applied to compute the loss and back-propagation updates the weights (the word embeddings).
2. Processing:

   -Build the corpus vocabulary -Build a skip-gram [(target, context), relevancy] generator -Build the skip-gram model architecture -Train the Model -Get Word Embeddings

3. $J(\theta)=-(1/T) * \sum t=1T * \sum \log p(wt+j|wt)$

   and $p(wt+j|wt)=exp(vt \cdot uTt+j)\sum v \in Vexp(vtuTv)$ where vt represents the word embedding for middle word wt and $ut+j$ represents word embedding for context word $wt+j$.

4. We use cross entropy to minimize the $J(\theta)$ function.

Bonus Question (1 point)

Find at least 2 interesting word vec combinations like the ones given in Q4

Do not delete the below cell

```
# YOUR CODE HERE
closest_bouns = model.most_similar(positive=['Photo', 'Camera'], negative=['Swimming'])
closest_bouns
```

```
[('photo', 0.5570032596588135), ('Photographer', 0.5424692630767822)]
```

✓ 0s completed at 4:31 PM