```
NAME = "Jeonghyun Lee"
```

Lab 7: Dimensionality Reduction

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 and the PDF file from Colab and upload it to Gradescope. 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

import pandas as pd
import numpy as np
import gensim
import requests
import string

from IPython.display import Image
from sklearn.manifold import TSNE

# To make the visualizations
!git clone https://github.com/CAHLR/d3-scatterplot.git
from google.colab.output import eval_js
from IPython.display import Javascript
from gensim.models import KeyedVectors

# To download trained model (Google news)
import gensim.downloader as api
google_model = api.load('word2vec-google-news-300')

import nltk
from nltk import word_tokenize, tokenize
nltk.download('punkt')
```

```
Requirement already satisfied: gensim in /usr/local/lib/python3.10/dist-packages (4.3.
Requirement already satisfied: numpy>=1.18.5 in /usr/local/lib/python3.10/dist-package
Requirement already satisfied: scipy>=1.7.0 in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: smart-open>=1.8.1 in /usr/local/lib/python3.10/dist-pack
```

Q1 (0.25 points)

Download your text corpus. (A good place to start is the nltk corpus or the gutenberg project)

```
18 # To download trained model (Google news)
```

```
#your code here
nltk.download('gutenberg')
```

```
#Save the raw text that you just downloaded in this variable
url = "https://www.gutenberg.org/cache/epub/72064/pg72064.txt"
raw = requests.get(url).content.decode('utf8')
```

```
--> 496         uncompress = self._decompressor.decompress(buf, size)
```

```
#This is an autograded cell, do not edit/delete
print(raw[:1000])
```

```
KeyboardInterrupt:
```

Q2 (0.25 points)

Tokenize your corpus. Make sure that that the result is a list of lists i.e. The top-level list (outer list) is a list of sentences, and the inner list is a list of words in a given sentence.

Consider the following text:

```
text = "I spent $15.35 on my lunch today. Food in Berkeley is very expensive!"
```

It could be tokenized as follows:

```
tok_corp = [['I', 'spent', '$', '15.35', 'on', 'my', 'lunch', 'today'],
 ['Food', 'in', 'Berkeley', 'is', 'very', 'expensive']]
```

Note: There are many different (and correct) ways of tokenizing. Your answer doesn't need to match exactly with this illustrative example.

```
#code here
pattern = r'''(?x)  # set flag to allow verbose regexps
(?:[A-Z]₩.)+       # abbreviations, e.g. U.S.A.
|₩w+(?:[-']₩w+)*    # words with optional internal hyphens
|₩$?₩d+(?:₩.₩d+)?   # currency, e.g. $12.80
|₩.₩.₩.            # elipses
|[.,;"'?()-_`]      # these are separate tokens
'''
tokenized_raw = " ".join(nltk.regexp_tokenize(raw, pattern))
tokenized_raw = tokenize.sent_tokenize(tokenized_raw)

nopunct = []
for sent in tokenized_raw:
    a = [w for w in sent.split() if w not in string.punctuation]
    nopunct.append(" ".join(a))
```

```
#Save the tokenized sentences as a list of list in this variable
tok_corp = [nltk.word_tokenize(sent) for sent in nopunct]
```

```
#This is an autograded cell, do not edit/delete
for sent in tok_corp[:3]:
  print(sent)
  print("₩n")
```

Q3 (0.25 points)

Train gensim using your own dataset. Name the trained model variable as `model`.

```
#code here
model = gensim.models.Word2Vec(tok_corp, min_count=1, vector_size=16, window=5)
```

```
#This is an autograded cell, do not edit/delete
print(f'Corpus Size: {model.corpus_total_words}')
print(f'Corpus Count: {model.corpus_count}')
print(f'Training time: {model.total_train_time}')
print(f'Sample words: {list(model.wv.index_to_key[:10])}')
```

Q4 (0.25 points)

Q4a

Create a list of the unique set of words from your corpus. Name the list variable as `unique_words`.

더블클릭 또는 Enter 키를 눌러 수정

```
#code here
unique_words = list(set([item for sublist in tok_corp for item in sublist]))
```

```
#This is an autograded cell, do not edit/delete
print(unique_words[:10])
len(unique_words)
```

Q4b

Extract respective vectors corresponding to the words in your corpus and store the vectors in a variable called `vector_list`.

```
#code here
vector_list = model.wv[unique_words]
```

```
#This is an autograded cell, do not edit/delete
print(f'Array Shape: {np.array(vector_list).shape}')
for i in range(5):
   print(unique_words[i], vector_list[i])
```

Q5 (3 points)

Based on your knowledge and understanding of the text corpus you have chosen, **form 3 hypotheses** of analogies or relationships (between words) that you expect will hold and **give a reason why. Experimentally validate these hypotheses** using similarity of the word vectors.

Example: If using Moby Dick as the corpus, one hypothesis might be that the whale, "Moby Dick" is (cosine) more similar to "fate" than "evil" because Moby Dick is symbolic of the nature and the universe and isn't necessarily 'bad'. Or "Moby Dick" is more similar to "opposition" than to "surrender" because Moby Dick fights for its survival.

Note: Please do NOT use the same example as in the prompt.

Note 2: It's okay if the model disproves your hypotheses.

---Your hypotheses here---

```
#your code here for validating hypotheses 1
#The word "marriage" is more similar to "happiness" than to "wealth" because, in the narrative of "Pride and Prejudice," the concept of marr

print(model.wv.similarity('marriage', 'happiness'))
print(model.wv.similarity('marriage', 'wealth'))
```

```
#your code here for validating hypotheses 2
#"Castle" is more similar to "history" than to "modern" because castles in Scotland are historical structures that are emblematic of Scotlanc

print(model.wv.similarity('Castle', 'history'))
print(model.wv.similarity('Castle', 'modern'))
```

```
#your code here for validating hypotheses 3
#"Sword" is more similar to "battle" than to "peace" since swords are likely to be associated with historical battles in Scottish stories, which


print(model.wv.similarity('sword', 'battle'))
print(model.wv.similarity('sword', 'peace'))
```

Q6 Visualizing the trained vectors (1.5 points)

Q6a

Run K-means clustering on your word vectors (as you did in Q-6 of Lab-5). Use the word vectors from the model you trained in this lab.

```
#your code here
from sklearn.cluster import KMeans
km = KMeans(n_clusters=25, random_state=42)
X = np.array(vector_list)
km.fit(X)
kmeans = pd.DataFrame()
kmeans['word'] = unique_words
kmeans['cluter'] = km.labels_
```

Q6b

Reduce the dimensionality of your word vectors using TSNE

```
#your code here
from sklearn.manifold import TSNE
import numpy as np


tsne_model = TSNE(n_components=2, perplexity=50, verbose=2, method='barnes_hut').fit_transform(vector_list)
```

Q6c

Create a dataframe with the following columns:

Column	Description
x	the first dimension of result from TSNE
y	the second dimension of result from TSNE
Feature 1	the word corresponding to the vector
Feature 2	the kmeans cluster label

Below is a sample of what the dataframe could look like:

x	y	Feature 1	Feature 2
7.154159	9.251100	lips	8
-53.254147	-13.514915	them	9
34.049191	-13.402843	topic	0
-32.515320	28.699677	sofa	24
13.006302	-4.270626	half-past	21

```
#your code here
df = pd.DataFrame(tsne_model[:,:2],columns=["x","y"])
df['Feature 1'] = unique_words
df['Feature 2'] = km.labels_
df.head()
```

Q6d: Visualization

In this question, you are required to visualize and explore the reduced dataset you created in Q6c using the d3-scatterplot library.

Note: The first code-chunk at the top in this notebook clones the library from github. Make sure that it has been executed properly before you proceed.

Save your dataset as a tsv file 'd3-scatterplot/mytext_new.tsv'

Example:

```
df.to_csv('d3-scatterplot/mytext_new.tsv', sep='₩t', index=False)
```

Note 1: The TSV file needs to be stored in the `d3-scatterplot` folder so that the d3-scatterplot library can access it.

Note 2: Make sure to save the TSV file WITHOUT the row index i.e. use `index=False`.

```
#your code here
df.to_csv('d3-scatterplot/mytext_new.tsv', sep='₩t', index=False)
```

Visualize the reduced vectors by running the following code

```
def show_port(port, data_file, width=600, height=800):
  display(Javascript("""
  (async ()=>{
    fm = document.createElement('iframe')
    fm.src = await google.colab.kernel.proxyPort(%d) + '/index.html?dataset=%s'
    fm.width = '90%%'
    fm.height = '%d'
    fm.frameBorder = 0
    document.body.append(fm)
  })();
  """ % (port, data_file, height)))

port = 8001
data_file = 'mytext_new.tsv'
height = 1400

get_ipython().system_raw('cd d3-scatterplot && python3 -m http.server %d &' % port)
show_port(port, data_file, height)
```

Color the points by their kmeans cluster. You can do this by choosing "Feature 2" as the variable in the "Color" drop-down in the visualization.

Please include a snapshot of your visualization in the notebook. Refer to the tutorial video, and/or follow the steps below:

1) Take a snapshot of the visualization and save it on your computer with the filename `trained_scatter.png`

2) Upload the `trained_scatter.png` by clicking on the 'Files' icon on the left sidebar and clicking on the upload icon (the one with an upward arrow on top left)

3) Run the code below to display the image

```
#This is an autograded cell, do not edit/delete
Image('trained_scatter.png')
```

Q7 Visualizing the PRE-TRAINED vectors (1.5 points)

In this question, you'll execute the same analysis as in Q6, but on PRE-TRAINED vectors.

Q7a

Load the google vector model

(It must be downloaded as `GoogleNews-vectors-negative300` for you if you ran the first code-chunk at the top of this notebook)

```
# google_model
```

Downsample the pre-trained google model to anywhere between 10,000 to 25,000 words.

```
#your code here
import random
sample_model = np.random.choice(list(google_model.wv.vocab), 20000, replace = False)
```

Create a list of the unique set of words from this downsampled model

```
#your code here
unique_words = list(set(sample_model))
```

Extract respective vectors corresponding to the words in the down-sampled, pre-trained model

```
#your code here
vector_list_sampled = google_model[unique_words]
```

Q7b

Run Kmeans clustering on the pre-trained word vectors. Make sure to use the word vectors from the pre-trained model.

```
#your code here
from sklearn.cluster import KMeans
km_google = KMeans(n_clusters=25, random_state=42)
X_google = np.array(vector_list_sampled)
km_google.fit(X_google)
```

Q7c

Reduce the dimensionality of the word vectors from the pre-trained model using tSNE

```
#your code here
data_embed_google = TSNE(n_components=2, perplexity=50, verbose=2, method='barnes_hut').fit_transform(vector_list_sampled)
```

Q7d

Create a dataframe with the following columns using the pre-trained vectors and corpus:

Column	Description
x	the first dimension of result from TSNE
y	the second dimension of result from TSNE
Feature 1	the word corresponding to the vector
Feature 2	the kmeans cluster label

Below is a sample of what the dataframe could look like:

x	y	Feature 1	Feature 2
7.154159	9.251100	lips	8
-53.254147	-13.514915	them	9
34.049191	-13.402843	topic	0
-32.515320	28.699677	sofa	24
13.006302	-4.270626	half-past	21

```
#your code here
df_google = pd.DataFrame(data_embed_google[:,:2],columns=["x","y"])
df_google['Feature 1'] = unique_words
df_google['Feature 2'] = km_google.labels_
df_google.head()
```

Q7e: Visualization

In this question, you are required to visualize and explore the reduced dataset **from the pretrained model** you created in Q7d using the d3-scatterplot library.

Note: The first code-chunk at the top in this notebook clones the libary from github. Make sure that it has been executed before you proceed.

Save your dataset as a tsv file 'd3-scatterplot/google_mytext.tsv'

Example:

```
google_df.to_csv('d3-scatterplot/google_mytext.tsv', sep='₩t', index=False)
```

Note 1: The TSV file needs to be stored in the `d3-scatterplot` folder so that the d3-scatterplot library can access it.

Note 2: Make sure to save the TSV file WITHOUT the row index i.e. use `index=False`.

```
#your code here
df_google.to_csv('d3-scatterplot/google_mytext.tsv', sep='₩t', index=False)
```

Visualize the reduced vectors by running the following code

```
def show_port(port, data_file, width=600, height=800):
  display(Javascript("""
  (async ()=>{
    fm = document.createElement('iframe')
    fm.src = await google.colab.kernel.proxyPort(%d) + '/index.html?dataset=%s'
    fm.width = '90%%'
    fm.height = '%d'
    fm.frameBorder = 0
    document.body.append(fm)
  })();
  """ % (port, data_file, height)))

port = 8001
data_file = 'google_mytext.tsv'
height = 1400
```

```
get_ipython().system_raw('cd d3-scatterplot && python3 -m http.server %d &' % port)
show_port(port, data_file, height)
```

Color the points by their kmeans cluster. You can do this by choosing "Feature 2" as the variable in the "Color" drop-down in the visualization.

Please include a snapshot of your visualization in the notebook. Refer to the tutorial video, and/or follow the steps below:

1) Take a snapshot of the visualization and save it on your computer with the filename `google_trained_scatter.png`

2) Upload the `google_trained_scatter.png` by clicking on the 'Files' icon on the left sidebar and clicking on the upload icon (the one with an upward arrow on top left)

3) Run the code below to display the image

```
#This is an autograded cell, do not edit/delete
Image('google_trained_scatter.png')
```

Q8: Exploration (0.5 points)

This is an open-ended question.

On the visualizations in Q6 & Q7, lasso-select a group of points with the left mouse button and look at summaries of the group on the right-side of the plot. (Refer to the tutorial video for a demo on the lasso selection). Also look at the words / features of the selected points.

Comment on at least 3 patterns / similarities you see in the selected words in the visualization for the pre-trained vectors and the vectors trained on your corpus. Are you able to find any group of points that are close to each other in the 2D space that also have semantic similarity?

--your answer here--