Step 0 - install and import dependencies

Suthasinee Pojam 6220422065

```
!pip install pythainlp
!pip install tensorflow_text
!pip install umap-learn
```

```
Requirement already satisfied: h5py>=2.9.0 in /usr/local/lib/python3.7/dist-packages (from tensorflow<2
Requirement already satisfied: protobuf>=3.9.2 in /usr/local/lib/python3.7/dist-packages (from tensorflow
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.21.0 in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: numpy>=1.14.5 in /usr/local/lib/python3.7/dist-packages (from tensorflow
Requirement already satisfied: tensorflow-estimator<2.8,~=2.7.0rc0 in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.7/dist-packages (from tensorflow<2.
Requirement already satisfied: wheel<1.0,>=0.32.0 in /usr/local/lib/python3.7/dist-packages (from tenso
Requirement already satisfied: wrapt>=1.11.0 in /usr/local/lib/python3.7/dist-packages (from tensorflow<
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /usr/local/lib/python3.7/dist-packages (from tenso
Requirement already satisfied: flatbuffers<3.0,>=1.12 in /usr/local/lib/python3.7/dist-packages (from ten
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.7/dist-packages (from tensorflo
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.7/dist-packages (from tensorf
Requirement already satisfied: gast<0.5.0,>=0.2.1 in /usr/local/lib/python3.7/dist-packages (from tensor
Requirement already satisfied: google-pasta>=0.1.1 in /usr/local/lib/python3.7/dist-packages (from tenso
Requirement already satisfied: cached-property in /usr/local/lib/python3.7/dist-packages (from h5py>=2.
Requirement already satisfied: tensorboard-data-server<0.7.0,>=0.6.0 in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: google-auth-oauthlib<0.5,>=0.4.1 in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.7/dist-packages (from tensorb
Requirement already satisfied: google-auth<3,>=1.6.3 in /usr/local/lib/python3.7/dist-packages (from ten
Requirement already satisfied: setuptools>=41.0.0 in /usr/local/lib/python3.7/dist-packages (from tensorl
Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in /usr/local/lib/python3.7/dist-packages (f
Requirement already satisfied: werkzeug>=0.11.15 in /usr/local/lib/python3.7/dist-packages (from tensor
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.7/dist-packages (from tenso
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.7/dist-packages (from go
Requirement already satisfied: cachetools<5.0,>=2.0.0 in /usr/local/lib/python3.7/dist-packages (from go
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.7/dist-packages (from google-auth
Requirement already satisfied: requests-oauthlib>=0.7.0 in /usr/local/lib/python3.7/dist-packages (from g
Requirement already satisfied: importlib-metadata>=4.4 in /usr/local/lib/python3.7/dist-packages (from m
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-packages (from importlib-metad
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in /usr/local/lib/python3.7/dist-packages (from pya
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (from requests<3,
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packages (from request
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.7/dist-p
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packages (from request
Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.7/dist-packages (from requests-o
Installing collected packages: tensorflow-text
Successfully installed tensorflow-text-2.7.3
Collecting umap-learn
  Downloading umap-learn-0.5.2.tar.gz (86 kB)
     |████████████████████████████████| 86 kB 2.0 MB/s
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.7/dist-packages (from umap-learn)
Requirement already satisfied: scikit-learn>=0.22 in /usr/local/lib/python3.7/dist-packages (from umap-le
Requirement already satisfied: scipy>=1.0 in /usr/local/lib/python3.7/dist-packages (from umap-learn) (1
Requirement already satisfied: numba>=0.49 in /usr/local/lib/python3.7/dist-packages (from umap-learn)
Collecting pynndescent>=0.5
```

```
  Downloading pynndescent-0.5.5.tar.gz (1.1 MB)
     |████████████████████████████████| 1.1 MB 13.0 MB/s
Requirement already satisfied: tqdm in /usr/local/lib/python3.7/dist-packages (from umap-learn) (4.62.3)
Requirement already satisfied: setuptools in /usr/local/lib/python3.7/dist-packages (from numba>=0.49->
Requirement already satisfied: llvmlite<0.35,>=0.34.0.dev0 in /usr/local/lib/python3.7/dist-packages (fro
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/dist-packages (from pynndescent>
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.7/dist-packages (from scikit
Building wheels for collected packages: umap-learn, pynndescent
  Building wheel for umap-learn (setup.py) ... done
  Created wheel for umap-learn: filename=umap_learn-0.5.2-py3-none-any.whl size=82709 sha256=eb41
  Stored in directory: /root/.cache/pip/wheels/84/1b/c6/aaf68a748122632967cef4dffef68224eb16798b679
  Building wheel for pynndescent (setup.py) ... done
  Created wheel for pynndescent: filename=pynndescent-0.5.5-py3-none-any.whl size=52603 sha256=89
```

```
import numpy as np
import pandas as pd
import re

import tensorflow as tf
import tensorflow_hub as hub
import tensorflow_text
import umap

from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

from sklearn.cluster import AgglomerativeClustering
from sklearn.neighbors import kneighbors_graph

import pythainlp
from pythainlp.corpus.common import thai_words
from pythainlp.util import Trie
import collections
```

```
module_url = 'https://tfhub.dev/google/universal-sentence-encoder-multilingual/3' #'https://tfhub.dev/google/univ

model = hub.load(module_url)
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
df = pd.read_csv("/content/Wongnai Reviews - Small.csv")
```

```
df.head()
```

	Review ID	Review
0	1	เป็นคนที่ชอบทาน Macchiato เป็นประจำ มีวันนึงเด...
1	2	Art of Coffee Kasetsart เป็นร้านกาแฟรสชาติเยี่...
2	3	กวงทะเลเผา อาหารทะเลเค้าสดจริงๆเนื้อปูหวานไม่ค...
3	4	วันนี้มีโอกาสตื่นเช้าครับเลยถึงโอกาสออกมาหาอะไ...
4	5	ชอบมาทานร้านนี้ถ้าอยากกินอาหารเวียดนามใกล้บ้าน...

Step 1 - document embedding and dimension reduction

```
#embed sentences using Universal Sentence Encoder (USE)

embed_comments_array = model(df['Review'].values).numpy()
embed_comments_array
```

```
array([[ 0.08993827,  0.01941084,  0.03787038, ..., -0.03488849,
         0.06299512,  0.04635989],
       [ 0.00634244,  0.00814594,  0.03071941, ..., -0.01478723,
        -0.03080936, -0.03316405],
       [ 0.0633687 , -0.02027139, -0.05077003, ..., -0.06530775,
        -0.00952999, -0.03439987],
       ...,
       [ 0.08775924,  0.03609736,  0.01263062, ..., -0.03102781,
        -0.03361677,  0.01928871],
       [ 0.05691195,  0.05381691, -0.0399575 , ..., -0.06598807,
        -0.05390478, -0.01037725],
       [ 0.0777048 ,  0.05080631,  0.02680681, ..., -0.0061413 ,
        -0.01313567,  0.02236264]], dtype=float32)
```

```
#reduce array dimensions using umap (you can chagne n_components)

reducer = umap.UMAP(random_state=42,n_components=50)
umap_embed_comments_array = reducer.fit_transform(embed_comments_array)
```

```
/usr/local/lib/python3.7/dist-packages/numba/np/ufunc/parallel.py:363: NumbaWarning: The TBB threading
  warnings.warn(problem)
```

Step 2 - document clustering using KMeans

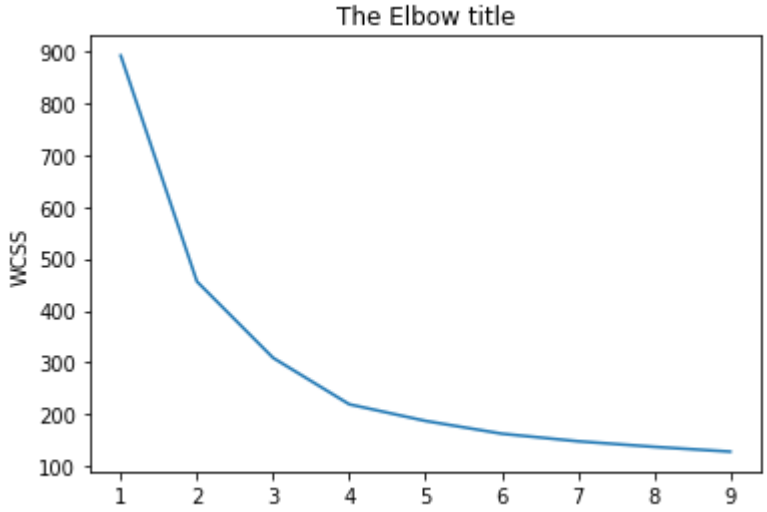
```
#run kmeans with various number of k. evaluate no. of k based on the elbow plot

wcss=[]
max_k = 10
for i in range(1, max_k):
  kmeans = KMeans(i)
  kmeans.fit(umap_embed_comments_array)
```

```
  wcss_iter = kmeans.inertia_
  wcss.append(wcss_iter)

number_clusters = range(1, max_k)
plt.plot(number_clusters,wcss)
plt.title('The Elbow title')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
```

```
Text(0, 0.5, 'WCSS')
```




```
#run kmeans with no. of clusters you see fit the most

k = 4

kmeans = KMeans(n_clusters = k)
kmeans.fit(umap_embed_comments_array)

df['KMeans ID'] = kmeans.labels_


#merge all reviews of each cluster into one big sentence

df_kmeans = pd.DataFrame(columns=["KMeans ID", "texts"])


for i in range(0, k):
  row = []
  row.append(i)
  row.append(df['Review'][df['KMeans ID'] == i].to_string())
  df_kmeans.loc[len(df_kmeans)] = row


df_kmeans
```

	KMeans ID	texts
0	0	0 เป็นคนที่ชอบทาน Macchiato เป็นประจำ มีว...
1	1	2 กวงทะเลเผา อาหารทะเลเค้าสดจริงๆเนื้อปูห...
2	2	3 วันนี้มีโอกาสตื่นเช้าครับเลยถึงโอกาสออก...
3	3	[illegible]

```
#create regex compiler for removal of a character you don't want

special_characters = "/[!@#$%^&*']/g"

specialchar_pattern = re.compile(special_characters)


#create regex compiler for removal of any emoji

emoji_pattern = re.compile("["
        u"\U0001F600-\U0001F64F"  # emoticons
        u"\U0001F300-\U0001F5FF"  # symbols & pictographs
        u"\U0001F680-\U0001F6FF"  # transport & map symbols
        u"\U0001F1E0-\U0001F1FF"  # flags (iOS)
                           "]+", flags=re.UNICODE)


#create regex compiler for removal of digit

number_pattern = re.compile("[0-9]")


#create regex compiler for removal of white space

space_pattern = re.compile("\s+")


#create regex compiler for removal of .

dot_pattern = re.compile(r"\.+")


#create regex compiler for removal of \

backslash_pattern = re.compile(r"\\+")


#create regex compiler for removal of [

left_pattern = re.compile(r"\[+")


#create regex compiler for removal of [

right_pattern = re.compile(r"\]+")
```

```
#create regex compiler for removal of "

quote_pattern = re.compile(r"\"+")


#define a function to tokenize a sentence into words - you can define words you want to remove as well as new wo

stopwords = list(pythainlp.corpus.thai_stopwords())
removed_words = ['u', 'b', 'n', 'nn', 'nn-', '\n', 'ร้าน']
screening_words = stopwords + removed_words

new_words = {"สตารบัก"}

words = new_words.union(thai_words())

custom_dictionary_trie = Trie(words)

def tokenize_to_list(sentence):
  merged = []
  words = pythainlp.word_tokenize(str(sentence), engine='newmm', custom_dict=custom_dictionary_trie)
  for word in words:
    if word not in screening_words:
      merged.append(word)
  return merged


#clean and tokenize sentences. count the occurences of each word

df_kmeans['texts'] = df_kmeans['texts'].apply(lambda x: emoji_pattern.sub(r'', x))
df_kmeans['texts'] = df_kmeans['texts'].apply(lambda x: specialchar_pattern.sub(r'', x))
df_kmeans['texts'] = df_kmeans['texts'].apply(lambda x: number_pattern.sub(r'', x))
df_kmeans['texts'] = df_kmeans['texts'].apply(lambda x: space_pattern.sub(r'', x))
df_kmeans['texts'] = df_kmeans['texts'].apply(lambda x: dot_pattern.sub(r'', x))
df_kmeans['texts'] = df_kmeans['texts'].apply(lambda x: backslash_pattern.sub(r'', x))

df_kmeans['texts'] = df_kmeans['texts'].apply(lambda x: left_pattern.sub(r'', x))
df_kmeans['texts'] = df_kmeans['texts'].apply(lambda x: right_pattern.sub(r'', x))
df_kmeans['texts'] = df_kmeans['texts'].apply(lambda x: quote_pattern.sub(r'', x))

df_kmeans['texts_tokenized'] = df_kmeans['texts'].apply(lambda x: tokenize_to_list(x))
df_kmeans['texts_count'] = df_kmeans['texts_tokenized'].apply(lambda x: collections.Counter(x).most_common())


#results of tokenization

df_kmeans
```

	KMeans ID	texts	texts_tokenized	texts_count
0	0	เป็นคนที่ชอบทานMacchiatoเป็นประจำ	[คน, ชอบ, ทาน, Macchiato, เป็นประจำ,	[(ร้านกาแฟ, 22), (กาแฟ, 19), (ทาน, 11),

```
#show top keywords of each cluster

top_N_words = 10

for i in range(0, len(df_kmeans)):
  print(f"Cluster ID : {i}\n")
  print(f"Most common words include : {list(df_kmeans['texts_count'][i])[:top_N_words]}\n")

#tune a model by remove unwanted characters and words and add more words to a custom dictionary
```

```
Cluster ID : 0

Most common words include : [('ร้านกาแฟ', 22), ('กาแฟ', 19), ('ทาน', 11), ('ชอบ', 8), ('คาเฟ่', 6), ('กิน', 6), ('แ

Cluster ID : 1

Most common words include : [('ร้านอาหาร', 11), ('กก', 7), ('กิน', 7), ('อร่อย', 6), ('ซอย', 6), ('อาหาร', 6), ('ทาน

Cluster ID : 2

Most common words include : [('กิน', 10), ('อร่อย', 9), ('ทาน', 7), ('ผม', 7), ('รีวิว', 7), ('บ้าน', 5), ('ร้านกาแฟ', 5

Cluster ID : 3

Most common words include : [('ชา', 18), ('นม', 14), ('ไข่มุก', 14), ('ทาน', 6), ('เครื่องดื่ม', 4), ('ร้า', 3), ('ตั้งอยู่',
```

Step 3 - document clustering using Agglomorative Clustering with cosine similarity

```
#clustering using agglomorative clustering

knn_graph = kneighbors_graph(embed_comments_array, 5, include_self=False)
model = AgglomerativeClustering(linkage="average", connectivity=knn_graph, n_clusters=10, affinity="cosine")
model.fit(embed_comments_array)
df['Agglomerative ID'] = model.labels_


#merge all reviews of each cluster into one big sentence

df_Agglomerative = pd.DataFrame(columns=["Agglomerative ID", "texts"])


for i in range(0, k):
  row = []
  row.append(i)
```

```
  row.append(str(df['Review'][df['Agglomerative ID'] == i].tolist()))
  df_Agglomerative.loc[len(df_Agglomerative)] = row



#clean and tokenize sentences. count the occurences of each word

df_Agglomerative['texts'] = df_Agglomerative['texts'].apply(lambda x: emoji_pattern.sub(r'', x))
df_Agglomerative['texts'] = df_Agglomerative['texts'].apply(lambda x: specialchar_pattern.sub(r'', x))
df_Agglomerative['texts'] = df_Agglomerative['texts'].apply(lambda x: number_pattern.sub(r'', x))
df_Agglomerative['texts'] = df_Agglomerative['texts'].apply(lambda x: space_pattern.sub(r'', x))
df_Agglomerative['texts'] = df_Agglomerative['texts'].apply(lambda x: dot_pattern.sub(r'', x))
df_Agglomerative['texts'] = df_Agglomerative['texts'].apply(lambda x: backslash_pattern.sub(r'', x))

df_Agglomerative['texts'] = df_Agglomerative['texts'].apply(lambda x: left_pattern.sub(r'', x))
df_Agglomerative['texts'] = df_Agglomerative['texts'].apply(lambda x: right_pattern.sub(r'', x))
df_Agglomerative['texts'] = df_Agglomerative['texts'].apply(lambda x: quote_pattern.sub(r'', x))

df_Agglomerative['texts_tokenized'] = df_Agglomerative['texts'].apply(lambda x: tokenize_to_list(x))
df_Agglomerative['texts_count'] = df_Agglomerative['texts_tokenized'].apply(lambda x: collections.Counter(x).most



#show top keywords of each cluster

top_N_words = 10

for i in range(0, len(df_Agglomerative)):
  print(f"Cluster ID : {i}\n")
  print(f"Most common words include : {list(df_Agglomerative['texts_count'][i])[:top_N_words]}\n")
```

```
Cluster ID : 0

Most common words include : [('อร่อย', 508), ('ทาน', 416), ('รสชาติ', 407), ('ดี', 347), ('กิน', 339), ('กาแฟ', 31

Cluster ID : 1

Most common words include : [('แตงโม', 22), ('น้ำ', 8), ('ปั่น', 6), ('เนื้อ', 6), ('เลือก', 4), ('ซื้อ', 4), ('ดื่ม', 4), ('พั

Cluster ID : 2

Most common words include : [('ดิชั้น', 4), ('แย่มาก', 3), ('"', 2), ('โต๊ะ', 2), ('รอง', 2), ('แก้ว', 2), ('ดิ', 1), ('ชั้น',

Cluster ID : 3

Most common words include : [('นม', 3), ('"', 2), ('แน่น', 2), ('tamp', 2), ('เท', 2), ('แก้', 2), ('เรื่อง', 1), ('ขนม',
```

Step 4 - result discussion

จากการตัดคำที่ซ้ำกันในทุก cluster ออกทำให้ได้ผลลัพท์ใหม่

Cluster 0 : review ประเภทเครื่องดื่ม + อาหาร มีเมนูกาแฟ และรีวิวดี

Cluster 1 : review ประเภทเครื่องดื่ม + อาหาร โดยเมนูที่คนกินเยอะจะเป็นเมนูน้ำผลไม้ปั่น

Cluster 2 : review ร้านอาหารไทย ไม่ค่อยอร่อย

Cluster 3 : เป็น review ทั่วไปที่ไม่ได้เจาะจงมีทั้งแย่และไม่แย่ คนเยอะ

✓ 0 วินาที เสร็จสมบูรณ์เมื่อ 23:10