CS635 - Homework 3 Submission

Student Name - Vivek Lad

Email - vl9244@rit.edu

GitHub Submission link - Copy and paste the GitHub submission Link through the classroom

Disclaimer: Consider this a guide to follow. You are not required to use these functions, you are free to use your own. The overall structure should be retained and the header modified.

```
!pip install sentence_transformers
```

```
    Collecting sentence_transformers
      Downloading https://files.pythonhosted.org/packages/78/e0/65ad8fd86eba720412d9ff10
         |████████████████████████████████| 71kB 2.0MB/s
    Collecting transformers>=3.0.2
```

```
# Pre Processing
import datetime
import math
import pandas as pd
import warnings
from collections import defaultdict,OrderedDict
import os, json, re, string
import numpy as np
from sentence_transformers import SentenceTransformer

#Change this function for the file paths if its different.
def load_default_parameters():
    train_file = "data/facebook/processed/fb_train.json"
    dev_file = "data/facebook/processed/fb_dev.json"
    test_file = "data/facebook/processed/fb_test.json"
    output_file = "facebook_kmeans"
    folder_name = "data/facebook/processed/fb/kmeans_predict"
    return train_file,dev_file,test_file,output_file,folder_name

def get_feature_vectors_only(fdict, data):
    #output = {}
    output = defaultdict(list)
    for item in data:
        vect = vectorize(fdict, item["labels"])
        total_labels = float(sum(vect))
        vect[:] = [x /total_labels for x in vect]
        item["message_id"] = item["message_id"]
        output[item["message_id"]] = vect
    return output

def compile_tweet_dict(json_list):
    result = {int(x["message_id"]): x["message"] for x in json_list}
    return result

def create_folder(foldername):
    if not os.path.exists(foldername):
        os.makedirs(foldername)

def read_json(fname):
    datastore = defaultdict(list)
    if fname:
        with open(fname, 'r') as f:
            datastore = json.load(f)
    return datastore

def get_data_dict (l):
    enuml = enumerate(l)
    fdict = defaultdict(list)
    rdict = defaultdict(list)
    fdict = {k:v for v, k in enuml}
    rdict = {k:v for v, k in fdict.items()}
```

```
  for i in train_message_dict.keys():
    cluster_dict[i] = cluster_assignment[index]
    index = index + 1
  #print(cluster_assignment)
  #print(cluster_dict)

  #print("Cluster Dict = ",cluster_dict)
  list1 = []
  for i in range(num_of_clusters):
    l = [0]*5
    list1.append(l)

  #print("List1 intially = " , list1)
  cluster_size_counter = [0] * num_of_clusters
  for key in cluster_dict.keys():
    cluster_num = cluster_dict[key]
    z = train_answer_counters[str(key)]
    for i in range(5):
      list1[cluster_num][i] = list1[cluster_num][i] + z[i]
    cluster_size_counter[cluster_num] = cluster_size_counter[cluster_num] + 1
 # print("Number of items per cluster = ")
 # print(cluster_size_counter)

  #print(list1)
  for i in range(num_of_clusters):
    for j in range(5):
      list1[i][j] = list1[i][j] / cluster_size_counter[i]

  #print("PDF = ")
 # for i in range(num_of_clusters):
   # print(list1[i])

  q = [1/5] * 5
  avgKLloss = 0
  for i in range(num_of_clusters):
    avgKLloss = avgKLloss + KLDivergence(list1[i], q)
  avgKLloss = avgKLloss / num_of_clusters

  Klloss_array.append(avgKLloss)
  print("Num of cluster = ",c, " Average KL loss = " , avgKLloss)
  if avgKLloss < avgKLlossGlobal:
    avgKLlossGlobal = avgKLloss
    cluster_size = c
print()

sns.lineplot(np.arange(4,35), Klloss_array)
print()
print("The cluster size for which the loss is minimum is = ", cluster_size)
print("The minimum loss is = ", avgKLlossGlobal)
```

```
for c in range(4,35,1):
  num_of_clusters = c
  clustering_model = KMeans(n_clusters=num_of_clusters)
  clustering_model.fit(label_list)
  cluster_assignment = clustering_model.labels_

  #print(cluster_assignment)
 # print("Number of clusters = " , num_of_clusters)
  cluster_dict = {}
  index = 0
  for i in train_message_dict.keys():
    cluster_dict[i] = cluster_assignment[index]
    index = index + 1
  #print(cluster_assignment)
  #print(cluster_dict)

  #print("Cluster Dict = ",cluster_dict)
  list1 = []
  for i in range(num_of_clusters):
    l = [0]*5
    list1.append(l)

  #print("List1 intially = " , list1)
  cluster_size_counter = [0] * num_of_clusters
  for key in cluster_dict.keys():
    cluster_num = cluster_dict[key]
    z = train_answer_counters[str(key)]
    for i in range(5):
      list1[cluster_num][i] = list1[cluster_num][i] + z[i]
    cluster_size_counter[cluster_num] = cluster_size_counter[cluster_num] + 1
 # print("Number of items per cluster = ")
 # print(cluster_size_counter)

  #print(list1)
  for i in range(num_of_clusters):
    for j in range(5):
      list1[i][j] = list1[i][j] / cluster_size_counter[i]

  #print("PDF = ")
 # for i in range(num_of_clusters):
   # print(list1[i])

  q = [1/5] * 5
  avgKLloss = 0
  for i in range(num_of_clusters):
    avgKLloss = avgKLloss + KLDivergence(list1[i], q)
  avgKLloss = avgKLloss / num_of_clusters

  Klloss_array.append(avgKLloss)
  print("Num of cluster = ",c, " Average KL loss = " , avgKLloss)
  if avgKLloss < avgKLlossGlobal:
    avgKLlossGlobal = avgKLloss
    cluster_size = c
print()

sns.lineplot(np.arange(4,35), Klloss_array)
```

```
        return (fdict, rdict)

    def vectorize(fdict, labels):
        vect = defaultdict(list)
        vect = [0] * len(fdict)
        for name,number in labels.items():
            vect[fdict[name]] = number
        return vect

    def write_model_logs_to_json(MODEL_LOG_DIR, results_dict, output_name):
        with open(MODEL_LOG_DIR +"/"+ output_name + ".json", "w") as fp:
            json.dump(results_dict, fp, sort_keys=True, indent=4)
        print ("Saved to "+MODEL_LOG_DIR +"/"+ output_name + ".json")

    def read_labeled_data_KMeans(filename):
        answer_counters = defaultdict(list)
        JSONfile = read_json(filename)
        message_dict = compile_tweet_dict(JSONfile["data"])
        (fdict, label_dict) = get_data_dict(JSONfile["dictionary"])
        answer_counters = get_feature_vectors_only(fdict, JSONfile["data"])
        return answer_counters,message_dict,label_dict

    def preprocess_data(input_train_file_name,input_dev_file_name,input_test_file_name,folder_

        create_folder(folder_name)
        create_folder(folder_name + "/logs")
        create_folder(folder_name + "/logs/models")

        train_answer_counters,train_message_dict,label_dict = read_labeled_data_KMeans(input_t

        dev_answer_counters,dev_message_dict,label_dict = read_labeled_data_KMeans(input_dev_f

        test_answer_counters,test_message_dict,label_dict = read_labeled_data_KMeans(input_tes

        return train_answer_counters,dev_answer_counters,label_dict,train_message_dict,dev_mes
```

Pre-Processing Data and Loading them up for your pipeline

```
train_file,dev_file,test_file,output_file,folder_name = load_default_parameters()
#Reading Data
train_answer_counters,dev_answer_counters,label_dict,train_message_dict,dev_message_dict,t
print(train_answer_counters)
```

```
defaultdict(<class 'list'>, {'10154485216228132': [0.9727272727272728, 0.01818181818
```

Q2

```
import regex as re
from sklearn.cluster import KMeans
embedder = SentenceTransformer('bert-base-nli-mean-tokens')
```

```
for i in train_message_dict.keys():

  train_message_dict[i]= re.sub(r"[^A-Za-z0-9]+", " ",train_message_dict[i])
#print(corpusList)

corpus_embeddings = embedder.encode(train_message_dict.values())

num_of_clusters = 5
clustering_model = KMeans(n_clusters=num_of_clusters)
clustering_model.fit(corpus_embeddings)
cluster_assignment = clustering_model.labels_

cluster_dict = {}
index = 0
for i in train_message_dict.keys():
  cluster_dict[i] = cluster_assignment[index]
  index = index + 1
print("The cluster distribution of the data space is = ")
print(cluster_assignment)
#print(cluster_dict)
```

```
The cluster distribution of the data space is =
[1 2 4 3 1 3 1 0 1 3 2 0 4 3 2 3 2 1 0 1 1 3 3 0 0 3 3 0 0 2 2 0 3 3 4 3 3
 1 0 0 3 2 3 3 0 1 2 0 4 3 2 4 0 4 1 3 0 2 3 0 2 0 1 4 0 2 3 1 0 3 0 3 4 3
 0 2 1 3 0 3 0 2 2 1 1 3 4 3 4 3 4 3 4 3 0 0 1 0 2 1 0 3 1 0 1 3 3 4 3 3 3
 1 4 2 3 4 1 2 0 2 2 3 3 3 3 3 2 1 1 0 1 0 3 3 1 4 1 2 3 2 3 2 1 3 0 4 2 4
 3 0 1 3 2 3 1 1 0 4 0 1 1 3 1 4 3 4 0 3 3 0 3 4 3 3 3 3 1 0 0 3 1 3 2 1 3
 3 3 0 3 0 1 1 2 4 2 4 4 0 3 4 1 0 0 0 3 3 3 0 4 0 3 0 3 3 0 2 4 2 0 0 3 0
 1 2 0 0 3 2 3 1 0 3 1 3 4 3 0 3 1 0 2 3 4 2 2 2 0 0 0 4 0 3 1 3 4 0 2 0 2
 4 1 4 1 1 3 1 3 1 1 3 2 1 1 3 4 3 2 0 2 0 3 1 2 2 0 2 0 0 0 4 4 0 1 1 1 1
 4 1 4 2 1 3 3 2 0 3 0 1 1 3 3 2 2 1 0 3 0 4 2 3 2 3 2 3 3 4 0 0 0 2 0 3 1
 0 4 3 4 4 1 3 3 0 1 3 3 0 4 0 3 0 0 3 1 3 2 1 3 3 2 0 4 4 0 0 0 1 3 0 3 4
 3 1 3 0 3 4 3 1 1 4 1 3 2 3 3 2 4 0 4 0 3 1 2 0 2 2 2 0 0 3 3 3 2 3 3 3 1
 1 4 4 3 1 1 2 1 4 3 1 3 4 2 1 3 2 2 0 3 0 0 2 2 2 3 0 2 3 1 2 1 2 3 2 4 3
 1 3 2 3 0 1 1 3 2 0 0 0 3 3 4 2 1 4 0 3 1 3 1 4 3 2 1 1 3 1 4 3 2 4 0 3 1
 1 0 4 3 0 3 1 0 3 1 1 2 3 2 2 3 2 3 3 3 4 2 0 3 0 3 4 3 3 1 3 3 3 2 1 2 3
 3 3 4 3 0 3 3 2 1 1 4 1 3 0 2 1 2 2 4 2 3 1 3 4 1 1 0 0 3 3 4 3 3 0 3 2 2
 0 4 3 3 0 0 1 4 3 1 3 3 1 3 4 0 3 3 2 1 3 4 0 1 4 0 1 2 1 1 3 4 3 2 1 3 2
 0 4 4 3 3 4 4 1 1 2 0 2 3 3 4 1 0 0 3 4 3 2 4 3 0 3 4 1 3 4 4 4 4 1 3 3 1
 2 1 0 0 3 1 1 2 0 2 3 3 1 1 2 1 3 3 3 0 0 0 4 3 1 0 4 4 3 3 0 3 2 1 3 4 4
 3 3 4 3 1 1 0 3 3 1 2 3 3 4 3 3 2 0 4 4 4 3 0 1 3 3 3 2 3 3 2 4 1 2 2 2 1
 4 0 3 1 4 0 4 4 0 2 4 4 2 1 0 1 2 2 4 4 3 1 3 0 3 0 1 4 0 3 3 1 0 1 1 0 2
 0 4 2 2 1 0 1 1 3 0 2 4 3 0 3 3 4 1 0 4 2 3 1 1 0 0 1 3 4 1 1 2 3 3 0 0 4
 3 0 0 2 0 3 3 4 2 3 0 1 1 1 3 3 3 1 0 0 2 0 0 0 0 3 4 4 2 1 0 4 0 1 2 0 0
 4 0 0 1 3 1 1 4 3 4 3 4 3 1 3 1 1 3 4 3 1 4 0 3 3 2 0 1 3 3 3 1 0 4 3 0 0
 3 0 1 3 0 0 4 1 1 1 2 3 3 0 1 2 4 4 4 3 4 0 3 3 4 3 1 3 4 3 2 2 4 2 3 4 3
 0 4 1 2 1 0 3 4 3 3 3 0 3 3 4 0 0 2 4 0 4 1 1 2 3 4 2 0 2 0 1 1 0 1 3 0 0
 3 4 1 4 0 0 4 4 3 0 1 4 3 2 1 2 3 3 0 2 2 0 1 0 1 0 3 2 0 0 1 1 3 3 1 3 4
 1 1 3 1 1 2 4 2 3 4 0 3 0 1 2 3 0 0 0 4 0 3 2 4 0 3 0 3 0 3 1 3 4 3 3 3 4]
```

# Q3

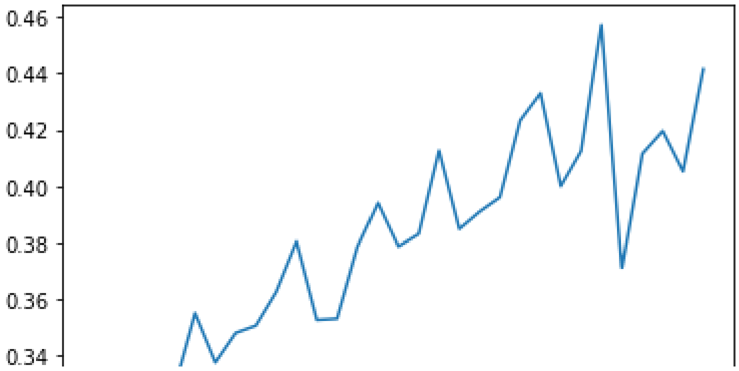
```
list1 = []
for i in range(num_of_clusters):
  l = [0]*num_of_clusters
  l = [0]*num_of_clusters
  list1.append(l)

cluster_size_counter = [0] * num_of_clusters
for key in cluster_dict.keys():
  cluster_num = cluster_dict[key]
  z = train_answer_counters[str(key)]
  for i in range(5):
    list1[cluster_num][i] = list1[cluster_num][i] + z[i]
  cluster_size_counter[cluster_num] = cluster_size_counter[cluster_num] + 1
print("Number of items per cluster = ")
print(cluster_size_counter)

#print(list1)
for i in range(num_of_clusters):
  for j in range(5):
    list1[i][j] = list1[i][j] / cluster_size_counter[i]

print("PDF = ")
for i in range(num_of_clusters):
  print(list1[i])
```

```
Number of items per cluster = 
[209, 193, 146, 299, 152]
PDF = 
[0.2287825198300072, 0.17124861739506148, 0.15833602246223089, 0.25124381256292766,
[0.6302601430462572, 0.12807319494295039, 0.1415914850006423, 0.059282235480391604,
[0.45486689675324743, 0.1692993829977977, 0.279287229802749, 0.059360334811931204, 0
[0.49498061049740977, 0.17708356386660748, 0.1422480685088225, 0.07997300864836163,
[0.33214243427362805, 0.14223102587791295, 0.2655289652311941, 0.09359426194287461,
```

# Q4

```
from math import log2
import seaborn as sns

def KLDivergence(p, q):
  sum = 0
  for i in range(len(p)):
    if(p[i]!=0 and q[i]!=0):
      sum = sum + p[i] * log2(p[i]/q[i])
  return sum

avgKLlossGlobal = 99999
cluster_size = 0
Klloss_array = []
for c in range(4,35,1):
  num_of_clusters = c
  clustering_model = KMeans(n_clusters=num_of_clusters)
  clustering_model.fit(corpus_embeddings)
  cluster_assignment = clustering_model.labels_

 # print("Number of clusters = " , num_of_clusters)
  cluster_dict = {}
  index = 0
```

```
Num of cluster =  4  Average KL loss =  0.3297501033503399
Num of cluster =  5  Average KL loss =  0.3291055185270969
Num of cluster =  6  Average KL loss =  0.3329805394359627
Num of cluster =  7  Average KL loss =  0.3168235186904753
Num of cluster =  8  Average KL loss =  0.3287492015925444
Num of cluster =  9  Average KL loss =  0.35508466621180507
Num of cluster =  10  Average KL loss =  0.33752496536330645
Num of cluster =  11  Average KL loss =  0.34801641091902
Num of cluster =  12  Average KL loss =  0.35063652798651584
Num of cluster =  13  Average KL loss =  0.36258925597919417
Num of cluster =  14  Average KL loss =  0.38045558122290263
Num of cluster =  15  Average KL loss =  0.35262635235102496
Num of cluster =  16  Average KL loss =  0.3530831965064551
Num of cluster =  17  Average KL loss =  0.37855156285648695
Num of cluster =  18  Average KL loss =  0.3940179064371898
Num of cluster =  19  Average KL loss =  0.37865139204247145
Num of cluster =  20  Average KL loss =  0.3831180088523859
Num of cluster =  21  Average KL loss =  0.4125967368713487
Num of cluster =  22  Average KL loss =  0.3848441578946426
Num of cluster =  23  Average KL loss =  0.3909478521255912
Num of cluster =  24  Average KL loss =  0.3960267476795479
Num of cluster =  25  Average KL loss =  0.42332232668022884
Num of cluster =  26  Average KL loss =  0.4329698584925034
Num of cluster =  27  Average KL loss =  0.4000076209557135
Num of cluster =  28  Average KL loss =  0.4124344249078486
Num of cluster =  29  Average KL loss =  0.4572135688916327
Num of cluster =  30  Average KL loss =  0.37119015207716305
Num of cluster =  31  Average KL loss =  0.4115265493507645
Num of cluster =  32  Average KL loss =  0.4195571003567065
Num of cluster =  33  Average KL loss =  0.4052280103694281
Num of cluster =  34  Average KL loss =  0.4415485364392753


The cluster size for which the loss is minimum is =  7
The minimum loss is =  0.3168235186904753
```

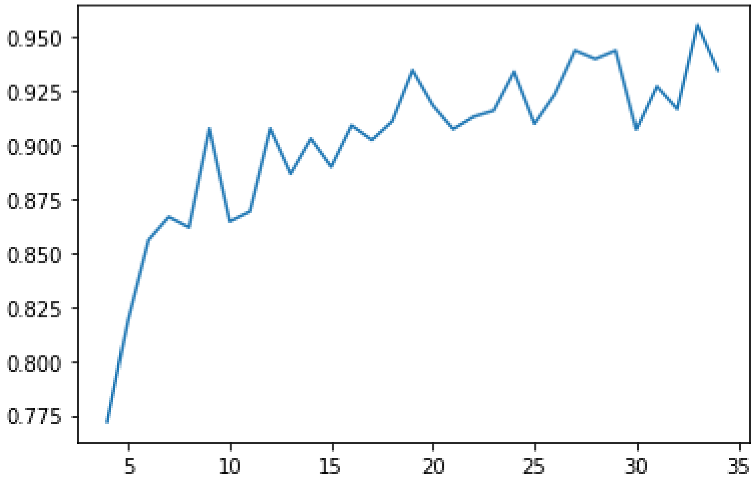


Q5

```
label_list = []

for key in train_answer_counters.keys():
  label_list.append(train_answer_counters[key])

avgKLlossGlobal = 99999
cluster_size = 0
Klloss_array = []
for c in range(4,35,1):
```

```
sns.lineplot(np.arange(4,35), KLloss_array)

print()
print("The cluster size for which the loss is minimum is = ", cluster_size)
print("The minimum loss is = ", avgKLlossGlobal)
```

```
Num of cluster =  4  Average KL loss =  0.7719809416673107
Num of cluster =  5  Average KL loss =  0.8187231882914097
Num of cluster =  6  Average KL loss =  0.8557736962883206
Num of cluster =  7  Average KL loss =  0.8664931139189449
Num of cluster =  8  Average KL loss =  0.8615752269604809
Num of cluster =  9  Average KL loss =  0.9074113016694579
Num of cluster =  10  Average KL loss =  0.8643274444130069
Num of cluster =  11  Average KL loss =  0.8689976443825338
Num of cluster =  12  Average KL loss =  0.9073937279336065
Num of cluster =  13  Average KL loss =  0.886402609101993
Num of cluster =  14  Average KL loss =  0.9027039363606064
Num of cluster =  15  Average KL loss =  0.889628366515904
Num of cluster =  16  Average KL loss =  0.9088673139541701
Num of cluster =  17  Average KL loss =  0.9020562093230318
Num of cluster =  18  Average KL loss =  0.9107107554501765
Num of cluster =  19  Average KL loss =  0.9342346189131023
Num of cluster =  20  Average KL loss =  0.9180789269125251
Num of cluster =  21  Average KL loss =  0.907115800861216
Num of cluster =  22  Average KL loss =  0.912845844176752
Num of cluster =  23  Average KL loss =  0.915652891619461
Num of cluster =  24  Average KL loss =  0.9335486580523354
Num of cluster =  25  Average KL loss =  0.9095424376276917
Num of cluster =  26  Average KL loss =  0.923088323432202
Num of cluster =  27  Average KL loss =  0.9433624020863746
Num of cluster =  28  Average KL loss =  0.9394689923224878
Num of cluster =  29  Average KL loss =  0.9433354784829232
Num of cluster =  30  Average KL loss =  0.9070581460918151
Num of cluster =  31  Average KL loss =  0.9267609421118836
Num of cluster =  32  Average KL loss =  0.916363432428849
Num of cluster =  33  Average KL loss =  0.9551420611924925
Num of cluster =  34  Average KL loss =  0.9342411837115199


The cluster size for which the loss is minimum is =  4
The minimum loss is =  0.7719809416673107
```


0.950
0.925
0.900
0.875
0.850
0.825
0.800
0.775
5
10
15
20
25
30
35