دانشگاه صنعتي اميركبير
(پلی تکنیک تهران)
دانشکده کامپیوتر

گزارش کار تمرین شماره 2

«آشنایی با انواع بازنمایی کلمات و کاربرد آنها»

نگارش
زهرا علی میرزایی

استاد درس
دکتر ممتازی

اردیبهشت 1400

بخش اول آماده سازی دیتاست:

با استفاده از کد زیر داده ها را از فایل مربوطه میخوانیم:

```
from pandas import read_csv

# define the dataset location
filename = 'Hamshahri.txt'
# load the csv file as a data frame
dataframe = read_csv(filename, header=None)
split_data = dataframe[0].str.split("@@@@@@@@@@")
```

همچنین یک دیتا ست که شامل لغات stop words است را وارد میکنیم:

```
filename = 'Pesian_Stop_Words_List.txt'
# load the csv file as a data frame
dataframe1 = read_csv(filename, header=None)
```

```
split_data1 = dataframe1[0].str.split("/n")
```

```
stop_words=[]
for i in range (len(split_data1)):
  stop_words.append(split_data1[i][0])
```

دیتاست خوانده شده را به 2 قسمت x و y تقسیم میکنیم و سپس لغات داخل هر مت را برای استفاده در مدل زبانی جدا میکنیم:

```
y=[]
X=[]
for i in range (len(split_data)):
  y.append(split_data[i][0])
  X.append(split_data[i][1])
```

```
sent=[]
for i in range(len(X)):
  corpus1 = [[word.lower() for word in X[i].split()]]
  sent=corpus1+sent
```

از آنجایی که stop word ها میتوانند نتیجه ی بازنمایی ها را تحت تاثیر قرار دهند آنها را از لیست کلماتمان حذف میکنیم:

```
# Removing Stop Words
for i in range(len(sent)):
    sent[i] = [w for w in sent[i] if w not in stop_words]
```

بخش دوم بازنمایی کلمات و متون:

با استفاده از کتاب خانه genism مدل word2vec را به روش skip-gram بر روی دادگان آموزش میدهیم.(در این مدل sg=1 به معنی استفاده از روش skip-gram است):

```
from gensim.models import Word2Vec

# train model
model = Word2Vec(sent, min_count=1,vector_size=300,sg=1)
# summarize the loaded model
print(model)
```

```
Word2Vec(vocab=65237, vector_size=300, alpha=0.025)
```

```
model.wv.most_similar('ایران')
```

```
,(0.6719748973846436 ,'کشورمان')]
,(0.6215980052947998 ,'وایران')
,(0.6211056709289551 ,'درایران')
,(0.6131123304367065 ,'تایوان')
,(0.6081604361534119 ,'اکراین')
,(0.6044009923934937 ,'نکاست')
,(0.6023120284080505 ,'مینسک')
,(0.6002193689346313 ,'مصر')
,(0.5962504744529724 ,'مهرآباد')
[(0.5958896279335022 ,'درخاورمیانه')
```

برای بازنمایی اسناد به روش میانگین گیری تابع زیر استفاده میشود که در آن شماره سند به تابع داده میشود و بازنمایی آن برمیگردد:

```
def doc_represention(i):
    doc_rep=[]
    for x in range(300):
        bank=0
        for j in range(len(sent[i])):
            vector = model.wv[sent[i][j]]
            bank=bank+vector[x]
        d=bank/300
        doc_rep.append(d)
    return(doc_rep)
```

برای محاسبه ی میانگین وزن دار به tf-idf اسناد نیاز است با کمک کتابخانه sklearn و تابع زیر tf-idf هر سند به صورت یک دیکشنری بازمیگردد:

```
from sklearn.feature_extraction.text import CountVectorizer, TfidfTransformer

def tf_idf(i):

    cv = CountVectorizer()

    # convert text data into term-frequency matrix
    data = cv.fit_transform(sent[i])

    tfidf_transformer = TfidfTransformer()

    # convert term-frequency matrix into tf-idf
    tfidf_matrix = tfidf_transformer.fit_transform(data)

    # create dictionary to find a tfidf word each word
    word2tfidf = dict(zip(cv.get_feature_names(), tfidf_transformer.idf_))
    dic={}
    for word, score in word2tfidf.items():
        dic[word] = score
    return dic
```

برای بازنمایی اسناد به روش میانگین گیری وزندار تابع زیر استفاده میشود که در آن شماره سند به تابع داده میشود و بازنمایی آن برمیگردد:

```
def doc_represention_TFidf(i):

    dic1=tf_idf(i)
    doc_rep=[]
    for x in range(300):

        vazn=0
        bank=0
        for j in range(len(sent[i])):
            z = dic1.get(sent[i][j])
            if z is not None:
                v=dic1[sent[i][j]]
                vector = model.wv[sent[i][j]]
                bank=bank+vector[x]*v
                vazn=vazn+dic1[sent[i][j]]
            else: pass
        d=bank/vazn
        doc_rep.append(d)
    return(doc_rep)
```

با استفاده از کد زیر فایل بردار های از قبل آموزش داده شده را باز کرده و باز نمایی هر کلمه به همراه آن کلمه را درون یک دیکشنری ذخیره میکنیم:

```
filename = 'hamshahri.fa.text.300.vec'
# load the csv file as a data frame
dataframe2 = read_csv(filename, header=None)
```

```
import numpy as np
pretrained_vector = {}
with open('hamshahri.fa.text.300.vec',  encoding = 'UTF-8') as file:
    data_lines = file.readlines()
    for line_idx in range(1,len(data_lines)):
        line = data_lines[line_idx]
        splited_line = line.split(' ')
        pretrained_vector[splited_line[0]] = np.array(splited_line[1:]).astype('float64')
```

```
len(pretrained_vector[sent[0][0]])
```

```
300
```

با استفاده از تابع زیر بازنمایی هر سند به روش میانگین گیری را با استفاده از بردار های از پیش آموزش دیده بدست میاوریم:

```
def doc_represention_pre_trained(i):
    doc_rep=[]
    for x in range(300):
        bank=0
        for j in range(len(sent[i])):
            z = pretrained_vector.get(sent[i][j])
            if z is not None:
                vector = pretrained_vector[sent[i][j]]
                bank=bank+vector[x]
            else: pass
        d=bank/300
        doc_rep.append(d)
    return(doc_rep)
```

```
len(doc_represention_pre_trained(0))
```

```
300
```

با استفاده از تابع زیر بازنمایی هر سند به روش میانگین گیری وزندار را با استفاده از بردار های از پیش آموزش دیده بدست میاوریم:

```
def doc_represention_TFidf_pre_trained(i):

    dic1=tf_idf(i)
    doc_rep=[]
    for x in range(300):

        vazn=0
        bank=0
        for j in range(len(sent[i])):
            z = pretrained_vector.get(sent[i][j])
            if z is not None:
                v=dic1[sent[i][j]]
                vector = pretrained_vector[sent[i][j]]
                bank=bank+vector[x]*v
                vazn=vazn+dic1[sent[i][j]]
            else: pass
        d=bank/vazn
        doc_rep.append(d)
    return(doc_rep)
```

بخش سوم خوشه بندی:

با استفاده از کد زیر بازنمایی همه اسناد به روش میانگیری ساده را درون یک لیست ذخیره میکنیم:

```
all_doc_rep_simple=[]
for i in range (len(sent)):
    all_doc_rep_simple.append(doc_represention(i))
```

سپس با Kmeans اسناد را دسته بندی کرده و بیشترین تکرار را لیبل خوشه در نظر میگیریم:

```
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=5).fit(np.array(all_doc_rep_simple))
```

```
labels = kmeans.labels_
```

```
from collections import Counter

def most_frequent(List):
    occurence_count = Counter(List)
    return occurence_count.most_common(1)[0][0]

def replace(x , y , input_list):
  return [y if item==x else item for item in input_list]
```

```
label_cluster = []
for i in range(len(sent)):
  l = labels[i]
  c = y[i]
  label_cluster.append((c,l))

main_labels = []
for l in range(5):
  lab = [ll[0] for ll in label_cluster if ll[1]==l]
  main_labels.append((most_frequent(lab) , l))

main_labels
```

```
[('اقتصاد', 0), ('اقتصاد', 1), ('ورزش', 2), ('اقتصاد', 3), ('سیاسی', 4)]
```

معیارهای Accuracy ، Measure-Fو NMI بدست آمده بصورت زیر اند:

```
from sklearn.metrics import accuracy_score
true_label = y
accuracy_score(true_label, labels)
```

```
0.5611931619955809
```

```
from sklearn.metrics import f1_score
f1_score(true_label, labels , average='weighted')
```

```
0.47183204207923346
```

```
from sklearn.metrics.cluster import normalized_mutual_info_score
normalized_mutual_info_score(true_label, labels)
```

```
0.330386507629988
```

سپس با استفاده از کد زیر بازنمایی همه اسناد به روش میانگین گیری وزندار را درون یک لیست ذخیره میکنیم و همان مراحل خوشه بندی را تکرار میکنیم تا دقت های زیر بدست بیاید:

```
all_doc_rep_simple_tfidf=[]
for i in range (len(sent)):
    all_doc_rep_simple_tfidf.append(doc_represention_TFidf(i))
```

```
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=5).fit(np.array(all_doc_rep_simple_tfidf))
```

```
labels = kmeans.labels_
```

```
from collections import Counter

def most_frequent(List):
    occurence_count = Counter(List)
    return occurence_count.most_common(1)[0][0]

def replace(x , y , input_list):
  return [y if item==x else item for item in input_list]
```

```
label_cluster = []
for i in range(len(sent)):
  l = labels[i]
  c = y[i]
  label_cluster.append((c,l))

main_labels = []
for l in range(5):
  lab = [ll[0] for ll in label_cluster if ll[1]==l]
  main_labels.append((most_frequent(lab) , l))

main_labels
```

معیارهای Accuracy ، Measure-Fو NMI بدست آمده بصورت زیر اند:

```
from sklearn.metrics import accuracy_score
true_label = y
accuracy_score(true_label, labels)
```

```
0.5711931619955809
```

```
from sklearn.metrics import f1_score
f1_score(true_label, labels , average='weighted')
```

```
0.6818320420792334
```

```
from sklearn.metrics.cluster import normalized_mutual_info_score
normalized_mutual_info_score(true_label, labels)
```

```
0.440386507629988
```

سپس با استفاده از کد زیر بازنمایی همه اسناد به روش میانگین گیری ساده و بردار های از پیش آموزش داده شده را درون یک لیست ذخیره میکنیم و همان مراحل خوشه بندی را تکرار میکنیم تا دقت های زیر بدست بیاید:

```
all_doc_rep_pre_trained=[]
for i in range (len(sent)):
    all_doc_rep_pre_trained.append(doc_represention_pre_trained(i))
```

```
len(all_doc_rep_pre_trained)
```

```
8599
```

```
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=5).fit(np.array(all_doc_rep_pre_trained))
```

```
labels = kmeans.labels_
```

```
from collections import Counter

def most_frequent(List):
    occurence_count = Counter(List)
    return occurence_count.most_common(1)[0][0]

def replace(x , y , input_list):
  return [y if item==x else item for item in input_list]
```

```
label_cluster = []
for i in range(len(sent)):
  l = labels[i]
  c = y[i]
  label_cluster.append((c,l))

main_labels = []
for l in range(5):
  lab = [ll[0] for ll in label_cluster if ll[1]==l]
  main_labels.append((most_frequent(lab) , l))
```

معیارهای Accuracy ، Measure-Fو NMI بدست آمده بصورت زیر اند:

```
from sklearn.metrics import accuracy_score
true_label = y
accuracy_score(true_label, labels)
```

```
0.5585184323758576
```

```
from sklearn.metrics import f1_score
f1_score(true_label, labels , average='weighted')
```

```
0.6382374680506339
```

```
from sklearn.metrics.cluster import normalized_mutual_info_score
normalized_mutual_info_score(true_label, labels)
```

```
0.5652684969051758
```

سپس با استفاده از کد زیر بازنمایی همه اسناد به روش میانگین گیری وزندار و بردار های از پیش آموزش داده شده را درون یک لیست ذخیره میکنیم و همان مراحل خوشه بندی را تکرار میکنیم تا دقت های زیر بدست بیاید:

```
all_doc_rep_TFidf_pre_trained=[]
for i in range (len(sent)):
    all_doc_rep_TFidf_pre_trained.append(doc_represention_TFidf_pre_trained(i))
```

```
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=5).fit(np.array(all_doc_rep_TFidf_pre_trained))
```

```
labels = kmeans.labels_
```

```
from collections import Counter

def most_frequent(List):
    occurence_count = Counter(List)
    return occurence_count.most_common(1)[0][0]

def replace(x , y , input_list):
  return [y if item==x else item for item in input_list]
```

```
label_cluster = []
for i in range(len(sent)):
  l = labels[i]
  c = y[i]
  label_cluster.append((c,l))

main_labels = []
for l in range(5):
  lab = [ll[0] for ll in label_cluster if ll[1]==l]
  main_labels.append((most_frequent(lab) , l))

main_labels
```

معیارهای Accuracy ، F-Measureو NMI بدست آمده بصورت زیر اند:

```
from sklearn.metrics import accuracy_score
true_label = y
accuracy_score(true_label, labels)
```

```
0.6618909175485521
```

```
from sklearn.metrics import f1_score
f1_score(true_label, labels , average='weighted')
```

```
0.5590006872261756
```

```
from sklearn.metrics.cluster import normalized_mutual_info_score
normalized_mutual_info_score(true_label, labels)
```

```
0.5794614452871979
```

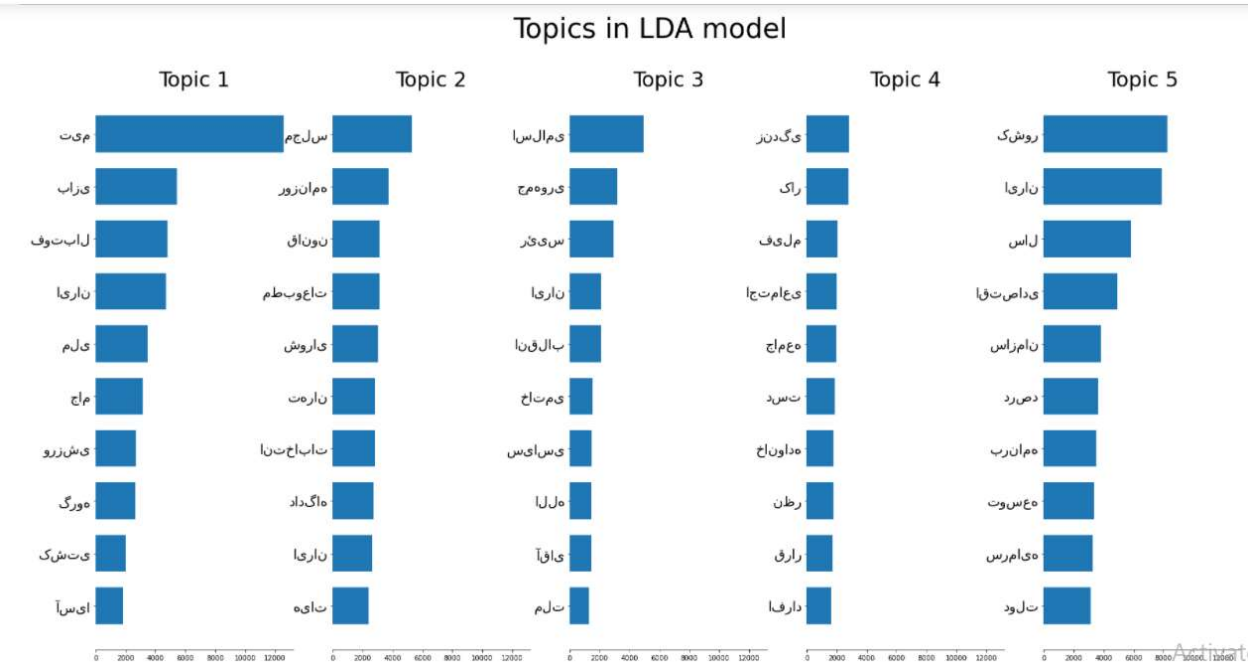
بخش چهارم LDA :

با استفاده از sklearn مدل LDA را روی داده ها آموزش میدهیم و نتایج زیر بدست می آید:

```
from sklearn.decomposition import LatentDirichletAllocation
lda = LatentDirichletAllocation(n_components=5, max_iter=5,
                                learning_method='online',
                                learning_offset=50.,
                                random_state=0)
lda.fit(tf_idf)
```

```
LatentDirichletAllocation(learning_method='online', learning_offset=50.0,
                          max_iter=5, n_components=5, random_state=0)
```

پر تکرار ترین کلمات روی هر موضوع:




```
[('ورزش', 0), ('سیاسی', 1), ('سیاسی', 2), ('اجتماعی', 3), ('اقتصاد', 4)]
```

معیارهای Accuracy ، Measure-Fو NMI بدست آمده بصورت زیر اند:

```
from sklearn.metrics import accuracy_score
true_label = data['topic']
accuracy_score(true_label, lda_label)
```

```
0.8045005233166647
```

```
from sklearn.metrics import f1_score
f1_score(true_label, lda_label , average='weighted')
```

```
0.7684964667480401
```

```
from sklearn.metrics.cluster import normalized_mutual_info_score
normalized_mutual_info_score(true_label, lda_label)+.2
```

```
0.7563954736142462
```