CSE 158, Fall 2019: Homework 4

Name: Hongjian Cui   PID: U08398995

In [1]:
```
import json
import numpy as np
import random
import string

from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn import linear_model
from collections import defaultdict
```

In [2]:
```
def parseData(fname):
    for l in open(fname):
        yield eval(l)
```

In [3]:
```
path = "/home/cui/Projects/PycharmProjects/CSE-158/data/train_Category.
```

In [4]:
```
dataset = list(parseData(path))
```

In [5]:
```
data = dataset[:10000]
```

In [6]:
```
punctuation = set(string.punctuation)
corpus = []

for d in data:
    text = d['review_text']
    text = text.lower()
    text = [c for c in text if not (c in punctuation)]
    text = ''.join(text)
    corpus.append(text)
```

In [7]:
```
corpus[1]
```

Out[7]:
```
'pretty decent the ending seemed a little rush but a good ending to th
e first trilogy in this series the fact that most of the time it is a
military fantasy makes it interesting also all of the descriptions of
food just make me hungry'
```

Question 1:

In [8]:
```
vec = CountVectorizer(ngram_range=(2, 2), dtype=np.uint16)
```

In [9]:
```
X = vec.fit_transform(corpus)
```

```
In [10]: bigram_dic = vec.vocabulary_
```

```
In [11]: bigram_dic
```

```
Out[11]: {'genuinely enthralling': 179383,
          'enthralling if': 143297,
          'if collins': 218676,
          'collins or': 93268,
          'or bernard': 333461,
          'bernard did': 58842,
          'did invent': 121687,
          'invent this': 232352,
          'this out': 465827,
          'out of': 337953,
          'of whole': 325661,
          'whole cloth': 515212,
          'cloth they': 91985,
          'they deserve': 462487,
          'deserve medal': 118119,
          'medal for': 285952,
          'for imagination': 167965,
          'imagination lets': 220783,
          'lets leave': 262384,
          'leave the': 260265
```

```
In [12]: len(bigram_dic)
```

```
Out[12]: 533839
```

```
In [13]: wordCount = sorted(bigram_dic.items(),key = lambda x:x[1], reverse = Tr
```

```
In [14]: wordCount[:5]
```

```
Out[14]: [('zzzaps continuing', 533838),
          ('zywych lub', 533837),
          ('zyndu is', 533836),
          ('zyndu are', 533835),
          ('zydy hwslh', 533834)]
```

Question 2:

```
In [15]: vec = CountVectorizer(ngram_range=(2, 2), dtype=np.uint16,
                                      max_features=1000)
         X = vec.fit_transform(corpus)
         X = X.toarray()
         y = [d['rating'] for d in data]
```

```
In [16]: clf = linear_model.Ridge(1.0, fit_intercept=False)
         clf.fit(X, y)
         predictions = clf.predict(X)
```

In [17]:

```
def MSE(predictions, targets):
    return ((predictions - targets) ** 2).mean()
```

In [18]:

```
print ("The MSE of the training set is {:.3f}."
                       .format(MSE(predictions, y)))
```

```
The MSE of the training set is 6.283.
```

Question 3:

In [19]:

```
vec = CountVectorizer(ngram_range=(1, 2), max_features=1000)
X = vec.fit_transform(corpus)
X = X.toarray()
y = [d['rating'] for d in data]
```

In [20]:

```
clf = linear_model.Ridge(1.0, fit_intercept=False)
clf.fit(X, y)
predictions = clf.predict(X)
```

In [21]:

```
print ("The MSE of the training set is {:.3f}."
                       .format(MSE(predictions, y)))
```

```
The MSE of the training set is 5.364.
```

Question 4:

In [22]:

```
tfidf_vec = TfidfVectorizer(ngram_range=(1, 1), dtype=np.float32)
X_tfidf = tfidf_vec.fit_transform(corpus)
```

In [23]:

```
idf = tfidf_vec.idf_ - 1
```

In [24]:

```
dict_idf = dict(zip(tfidf_vec.get_feature_names(), idf))
```

In [25]:

```
dict_idf['stories']
```

Out[25]:

```
2.5718725
```

In [26]:

```
dict_idf['magician']
```

Out[26]:

```
6.074946
```

In [27]:

```
dict_idf['psychic']
```

Out[27]:

```
5.952344
```

In [28]:

```
dict_idf['writing']
```

Out[28]:

```
2.296703
```

```
In [29]: dict_idf['wonder']

Out[29]: 4.062946
```

```
In [30]: X_tfidf = X_tfidf.toarray()
```

```
In [31]: dict_tfidf_first = dict(zip(tfidf_vec.get_feature_names(), X_tfidf[0]))
```

```
In [32]: dict_tfidf_first['stories']

Out[32]: 0.04845343
```

```
In [33]: dict_tfidf_first['magician']

Out[33]: 0.09597358
```

```
In [34]: dict_tfidf_first['psychic']

Out[34]: 0.18862091
```

```
In [35]: dict_tfidf_first['writing']

Out[35]: 0.044720683
```

```
In [36]: dict_tfidf_first['wonder']

Out[36]: 0.06868025
```

Question 5:

```
In [37]: tfidf_vec = TfidfVectorizer(ngram_range=(1, 1), dtype=np.float32,
                                     max_features=1000)
         X_tfidf = tfidf_vec.fit_transform(corpus)
         X_tfidf = X_tfidf.toarray()
         y = [d['rating'] for d in data]
```

```
In [38]: clf = linear_model.Ridge(1.0, fit_intercept=False)
         clf.fit(X_tfidf, y)
         predictions = clf.predict(X_tfidf)
```

```
In [39]: print ("The MSE of the training set is {:.3f}."
                                .format(MSE(predictions, y)))

         The MSE of the training set is 1.523.
```

Question 6:

```
In [40]: def cosineSimilarity(s1, s2):
             numer = sum(s1 * s2)
             denom = np.sqrt(sum(s1 ** 2)) * np.sqrt(sum(s2 ** 2))

             if denom == 0:
                 return 0
             else:
                 return numer / denom
```

```
In [41]: similarity = [cosineSimilarity(X_tfidf[0], d) for d in X_tfidf]
```

```
In [42]: similarity[0] = 0
```

```
In [43]: maxSimilarity = max(similarity)
```

```
In [44]: maxSimilarity

Out[44]: 0.5295288962174183
```

```
In [45]: index = similarity.index(maxSimilarity)
```

```
In [46]: data[index]['review_id']

Out[46]: 'r64325341'
```

Question 7:

In [1]:

```
import json
import numpy as np
import random
import string

from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn import linear_model
from collections import defaultdict
```

In [2]:

```
def parseData(fname):
    for l in open(fname):
        yield eval(l)
```

In [3]:

```
path = "/home/cui/Projects/PycharmProjects/CSE-158/data/train_Category.
```

In [4]:

```
dataset = list(parseData(path))
```

In [5]:

```
random.shuffle(dataset)
```

In [6]:

```
y_training = [d['rating'] for d in dataset[:10000]]
y_validation = [d['rating'] for d in dataset[10000:20000]]
y_test = [d['rating'] for d in dataset[20000:30000]]
```

In [7]:

```
def MSE(predictions, targets):
    return ((predictions - targets) ** 2).mean()
```

In [8]:

```
def function(ngrams, removePunctuation, tfidf, wordCounts):
    punctuation = set(string.punctuation)
    corpus = []
    max_features = 20000

    if removePunctuation:
        for d in dataset[:30000]:
            text = d['review_text']
            text = text.lower()
            text = [c for c in text if not (c in punctuation)]
            text = ''.join(text)
            corpus.append(text)
    else:
        for d in dataset[:30000]:
            tmp = d['review_text']
            tmp = tmp.lower()
            text = []
            for c in tmp:
                if c in punctuation:
                    text.append(" ")
                    text.append(c)
                    text.append(" ")
                else:
                    text.append(c)
            text = ''.join(text)
            corpus.append(text)

    X = []

    # if parameter tfidf is True, using tf-idf vectorizer
    if tfidf:
        vec = TfidfVectorizer(ngram_range=(ngrams, ngrams), Unigrams
                              max_features=max_features)
        X = vec.fit_transform(corpus)
        X = X.toarray()

    # if parameter wordCounts is True, using count vectorizer
    if wordCounts:
        vec = CountVectorizer(ngram_range=(ngrams, ngrams),
                              max_features=max_features)
        X = vec.fit_transform(corpus)
        X = X.toarray()

    X_training = X[:10000]
    X_validation = X[10000:20000]
    X_test = X[20000:30000]

    MSE_list = []
    regularization = [0.01, 0.1, 1, 10, 100]
    for r in regularization:
        clf = linear_model.Ridge(r, fit_intercept=False)
        clf.fit(X_training, y_training)
        predictions = clf.predict(X_validation)
        MSE_list.append((r, MSE(predictions, y_validation)))

        del clf
    return MSE_list, X_training, X_validation, X_test
```

In [9]:

```
def MSE_testSet(MSE_list, X_training, X_validation, X_test):
    r = min(MSE_list, key = lambda x:x[1])[0]

    clf = linear_model.Ridge(r, fit_intercept=False)
    clf.fit(X_training, y_training)
    predictions = clf.predict(X_test)

    mse = round(MSE(predictions, y_test), 3)

    print ("The MSE of the test set is {:.3f}.".format(mse))

    return mse
```

In [10]:

```
performance = []
```

1. Unigrams & Removing punctuation & tfidf scores

In [11]:

```
MSE_list, X_training, X_validation, X_test = function(1, True, True, Fa
```

In [12]:

```
MSE_list
```

Out[12]:

```
[(0.01, 4.231921387378723),
 (0.1, 2.530514873651722),
 (1, 1.980777863708946),
 (10, 2.26540381409375),
 (100, 3.3078669950698605)]
```

In [13]:

```
performance.append(("Unigrams, remove punctuation and using tf-idf scor
                    MSE_testSet(MSE_list, X_training, X_validation, X_t
```

```
The MSE of the test set is 1.940.
```

2. Unigrams & Removing punctuation & Word counts

In [14]:

```
MSE_list, X_training, X_validation, X_test = function(1, True, False, T
```

In [15]:

```
MSE_list
```

Out[15]:

```
[(0.01, 100.56290076827047),
 (0.1, 54.292731348707385),
 (1, 22.805745018853557),
 (10, 10.85740713173832),
 (100, 7.388132286034386)]
```

In [16]:

```
performance.append(("Unigrams, remove punctuation and using word counts
                    MSE_testSet(MSE_list, X_training, X_validation, X_t
```

```
The MSE of the test set is 7.116.
```

3. Unigrams & Preserving punctuation & tfidf scores

```
In [17]: MSE_list, X_training, X_validation, X_test = function(1, False, True, Fa
```

```
In [18]: MSE_list
```

```
Out[18]: [(0.01, 4.316604607819873),
          (0.1, 2.5358068742106763),
          (1, 1.9652322732523957),
          (10, 2.2433897060794266),
          (100, 3.2837694424099646)]
```

```
In [19]: performance.append(("Unigrams, preserve punctuation and using tf-idf sco
                             MSE_testSet(MSE_list, X_training, X_validation, X_te
```

```
The MSE of the test set is 1.932.
```

4. Unigrams & Preserving punctuation & Word counts

```
In [20]: MSE_list, X_training, X_validation, X_test = function(1, False, False, T
```

```
In [21]: MSE_list
```

```
Out[21]: [(0.01, 104.49586690530538),
          (0.1, 55.371576744287566),
          (1, 22.85794513506134),
          (10, 10.999730770757104),
          (100, 7.454292447464862)]
```

```
In [22]: performance.append(("Unigrams, preserve punctuation and using word count
                             MSE_testSet(MSE_list, X_training, X_validation, X_te
```

```
The MSE of the test set is 7.220.
```

5. Bigrams & Removing punctuation & tfidf scores

```
In [23]: MSE_list, X_training, X_validation, X_test = function(2, True, True, Fal
```

```
In [24]: MSE_list
```

```
Out[24]: [(0.01, 4.896307534843344),
          (0.1, 3.8012622958526743),
          (1, 3.0363105832826585),
          (10, 3.6130739101107743),
          (100, 6.587131482229626)]
```

```
In [25]: performance.append(("Bigrams, remove punctuation and using tf-idf scores
                             MSE_testSet(MSE_list, X_training, X_validation, X_te
```

```
The MSE of the test set is 2.889.
```

6. Bigrams & Removing punctuation & Word counts

```
In [26]: MSE_list, X_training, X_validation, X_test = function(2, True, False, Tı
```

```
In [27]: MSE_list
```

```
Out[27]: [(0.01, 61.28342585489422),
          (0.1, 33.54553164355451),
          (1, 19.63217253144386),
          (10, 10.230048234295362),
          (100, 7.455529943741971)]
```

```
In [28]: performance.append(("Bigrams, remove punctuation and using word counts"
                             MSE_testSet(MSE_list, X_training, X_validation, X_tє
```

```
The MSE of the test set is 7.127.
```

7. Bigrams & Preserving punctuation & tfidf scores

```
In [29]: MSE_list, X_training, X_validation, X_test = function(2, False, True, Fa
```

```
In [30]: MSE_list
```

```
Out[30]: [(0.01, 4.863034411156802),
          (0.1, 3.758555054253984),
          (1, 3.002637063205719),
          (10, 3.5770305692365634),
          (100, 6.538793974606483)]
```

```
In [31]: performance.append(("Bigrams, preserve punctuation and using tf-idf sco
                             MSE_testSet(MSE_list, X_training, X_validation, X_tє
```

```
The MSE of the test set is 2.856.
```

8. Bigrams & Preserving punctuation & Word Counts

```
In [32]: MSE_list, X_training, X_validation, X_test = function(2, False, False, ٦
```

```
In [33]: MSE_list
```

```
Out[33]: [(0.01, 61.10764171329383),
          (0.1, 34.60855769334343),
          (1, 19.747199837154405),
          (10, 10.290871247045763),
          (100, 7.420340762065661)]
```

```
In [34]: performance.append(("Bigrams, preserve punctuation and using word counts
                             MSE_testSet(MSE_list, X_training, X_validation, X_tє
```

```
The MSE of the test set is 7.089.
```

In [35]:

```python
for i in performance:
    print(i[0] + ": " + str(i[1]))

print("The best performance on test set is using " +
                  min(performance, key = lambda x:x[1])[0])
```

```
Unigrams, remove punctuation and using tf-idf scores: 1.94
Unigrams, remove punctuation and using word counts: 7.116
Unigrams, preserve punctuation and using tf-idf scores: 1.932
Unigrams, preserve punctuation and using word counts: 7.22
Bigrams, remove punctuation and using tf-idf scores: 2.889
Bigrams, remove punctuation and using word counts: 7.127
Bigrams, preserve punctuation and using tf-idf scores: 2.856
Bigrams, preserve punctuation and using word counts: 7.089
The best performance on test set is using Unigrams, preserve punctuati
on and using tf-idf scores
```