Z604-Project-2

by Qiwen Zhu

contents

Project Introduction

This project tries to implement a text classifier by employing Naive Bayes method and using Yelp review dataset. I use NLTK's Naive Bayes package to build the classifier.

Data Processing

The Yelp review dataset can be downloaded from:

Yelp Dataset Challenge

The dataset is in the format of JSON. It looks like:

```
{"votes": {"funny": 0, "useful": 0, "cool": 0},
"user_id": "PUFPaY9KxDAcGqfsorJp3Q",
"review_id": "Ya85v4eqdd6k9Od8HbQjyA",
"stars": 4,
"date": "2012-08-01",
"text": "Mr Hoagie is an institution. Walking in, it does seem like a
throwback to 30 years ago, old fashioned menu board, booths out of
the 70s, and a large selection of food. Their speciality is the
Italian Hoagie, and it is voted the best in the area year after year.
I usually order the burger, while the patties are obviously cooked
from frozen, all of the other ingredients are very fresh. Overall,
its a good alternative to Subway, which is down the road.", "type":
"review",
"business_id": "5UmKMjUEUNdYWqANhGckJw"}
```

To build a classifier, I only need the __* review texts *__ and __* coresponding stars *__. Because the project is implemented via Python, I use the following code to extract reviews and their corresponding labels/classes:

```python
import json
f = open(r"yelp_academic_dataset_review.json","r")
f_pos = open(r"review/positive_reviews","w")
f_neg = open(r"review/negative_reviews","w")
f_neu = open(r"review/neutral_reviews","w")
f_all = open(r"review/all_reviews","w")

for l in f:
    entry = json.loads(l)
    # put review into positive set
    if entry['stars'] > 3:
        line = entry['text'].replace('\n','')+'\t'+'positive'+'\n'
        line = line.encode('utf8','replace')
        f_pos.write(line)
        f_all.write(line)
    # put review into negative set
    elif entry['stars'] < 3:
        line = entry['text'].replace('\n','')+'\t'+'negative'+'\n'
        line = line.encode('utf8','replace')
        f_neg.write(line)
        f_all.write(line)
    else:
        # put review into neutral set
        line = entry['text'].replace('\n','')+'\t'+'neutral'+'\n'
        line = line.encode('utf8','replace')
        f_neu.write(line)
        f_all.write(line)

f.close()
f_pos.close()
f_neu.close()
f_neg.close()
f_all.close()
```

This script produces files looking like:

```
Yes this place is a little out dated and not opened on the weekend.
But other than that the staff is always pleasant and fast to make your
order. Which is always spot on fresh veggies on their hoggies and
other food. They also have daily specials and ice cream which is
really good. I had a banana split they piled the toppings on. They win
pennysaver awards ever years i see why. positive
All the food is great here. But the best thing they have is their
wings. Their wings are simply fantastic!!  The "Wet Cajun" are by the
best & most popular.  I also like the seasoned salt wings.  Wing Night
is Monday & Wednesday night, $0.75 whole wings!The dining area is
nice. Very family friendly! The bar is very nice is well.  This place
is truly a Yinzer's dream!!  "Pittsburgh Dad" would love this place
n'at!!  positive
```

Each line in the files is a review text and its corresponding label (positive/negtive/neutral), seperated by a *** Tab ***

This script produces 4 files:

- all_reviews
- negative_reviews (# stars <3)
- neutral_reviews (# stars = 3)
- positive_reviews (# start >3)

I only use *** negative_reviews***, *** positive_reviews ***, and *** neutral_reviews*** for the project, because *** all_reviews*** is very skewed, with 1492558 positive, 450540 negative and 282115 neutral in 2225213 reviews.

I use the following unix command to retrieve only the first 2000 reviews in each file and concatenate them into one file, named *** balanced_reviews_6000***:

```
$ awk 'NR <= 2000 {print $0}' positive_reviews >
balanced_reviews_6000
$ awk 'NR <= 2000 {print $0}' negative_reviews >>
balanced_reviews_6000
$ awk 'NR <= 2000 {print $0}' neutral_reviews >>
balanced_reviews_6000
```

This produces a file of 6000 lines, with each line consisting of one review and its label.

After this, I *** remove every punctuation *** in the review text, producing a file of 6000 linesls:

```
import string



f = open('balanced_reviews_6000')
f_out = open('balanced_reviews_6000_no_punctuation','w')
for l in f:
    words, sentiment = l.split('\t')
    for p in string.punctuation:
        words = words.replace(p, '')
    words_filtered = [w.lower() for w in words.split() if len(w) >=
3]
    f_out.write(' '.join(w for w in words_filtered)+'\t'+sentiment)

f.close()
f_out.close()
```

Then I *** serialize/pickle the data*** that my classifier can use directly, by using the following code:

```
try:
    import cPickle as pickle
except ImportError:
    import pickle

def get_data():
    f = open('balanced_reviews_6000_no_punctuation')
    all_reviews = []
    for l in f:
        words, label = l.split('\t')
        words = words.split()
        label = label.replace('\n','')
        all_reviews.append((words,label))
    return all_reviews

# print get_data()[:3]
def pickle_data(data):
    f = open('pickled_6000_reviews','wb')
    pickle.dump(data,f)
    f.close()
    return

if __name__ == '__main__':
    pickle_data(get_data())
```

Now the data for the project is prepared.

Machine Leaning

I use Naive Bayes algorithm package provided via NLTK.

First, I retrieve pickled data:

```
import nltk
import pprint
try:
    import cPickle as pickle
except ImportError:
    import pickle

pickled_file = 'pickled_6000_reviews'

def read_pickled_reviews(pickled_file):
    f = open(pickled_file,'rb')
    data = pickle.load(f)
    f.close()
    return data

data  = read_pickled_reviews()
```

Then, I put every word in the 6000 reviews into a list, named *** all_words*** :

```
def get_words_in_data(data):
    all_words = []
    for words, label in data:
        all_words.extend(words)
    return all_words

all_words = get_words_in_data(data)
```

After this, I use the all_words list to produce word features,producing a list of each word in all_words without repetition:

```
word_features = list(set(all_words))
```

Next, I *** define a function to extract features ***, by simply counting frequency of each word in all_words, so that when I build the training set with nltk's apply_feature function, I can use the extract feature function:

```
def extract_features(doc):
    doc_words = set(doc) # get all words unrepeatedly in doc, a doc
is a list of words in review.
    features = {}
    for word in word_features:
        features['contains(%s)' % word] = (word in doc_words)
    return features
```

Now, I apply the function to data of 6000 reviews, in order to extract features to train a classifier. the feature data set for NLTK Naive Bayes classifier is a list of tuples of two; tuple[0] is a dictionary showing whether a review contains feature words; and tuple[1] is the label for the review. the data set looks like:

```
[({'contains(007)': False,
   'contains(00s)': False,
   'contains(00way)': False,
   .......
   'contains(zuchinni)': False,
   'contains(zuma)': False,
   'contains(zuppa)': False,
   'contains(\xe2\x82\xac20)': False},
  'positive')]
```

As we will see, the size of features list (the feature data set) can be very huge. For 6000 reviews, there are 28553 unique words, and thus the feature data set will be represented by a matrix of 6000 X 28553. This is *** too big *** .

The huge size of the feature matrix is because of *** two major reasons ***:

1. There are *** too many words that do not provide much useful information for classification ***, such as *** "the","this","at","is" and etc ***, *** stop words ***
2. There are *** too many words from same radical word ***, such as "zesty", "zestier", "zest", and etc.

Thus, I perform the following operations the all_words list:

1. Remove stop words
2. Lemmatize the remaining words

Code for above operations:

```
# initially, len(all_words) is 630904
from nltk.corpus import stopwords
from nltk.stem.porter import *
all_words = [w for w in all_words if w.lower() not in
stopwords.words('english')]
# after removing stop words, there remain 400458 words
ps = PorterStemmer()
all_words = map(ps.stem, all_words)
word_features = list(set(all_words))
# after stemming every words in all_words, len(word_features) is only
21793
# not huge improvement, but this can be further explored in the
future.
```

Now that I have a feature extraction function and data of 6000 reviews and their labels, I produce the feature data set (list of tuples):

```
feature_data_set = [(extract_features(review),label) for review,
label in data]
```

For *** 10-fold cross-validation***, I use ***scikit-learn *** :

```
from sklearn import cross_validation
# this will produce lists of index, with which the feature_data_set
list can be devided into training set and test set.
c_v = cross_validation.KFold(len(feature_data_set), n_folds = 10,
shuffle = False, random_state = None)
# if shuffle = True, then the feature_data_set will be shuffled
before split
# do the 10_fold cross_validation and print accuracy of each
classifier.
for cv_train, cv_test in c_v:
    classifier =
nltk.NaiveBayesClassifier.train(feature_data_set[cv_train[0]:cv_train
[len(cv_train)-1]])
    print 'accuracy: %.2f' % nltk.classify.util.accuracy(classifier,
feature_data_set[cv_test[0]:cv_test[len(cv_test)-1]])
```

Results

Because of the length of feature vector and the cpu time limit of Karst server, the dataset of 6000 reviews didn't work. Thus I use a dataset of 3000 reviews, with 1000 reviews from each of the three review files (positive/negative/neutral).

The 10-fold cross validation results, *** feature set reduced***, and *** without *** shuffle:

```
$ time -p python train_test.py
give me the pickled file:
pickled_1500_reviews
9298
start to train a new classifier
accuracy: 0.30872
start to train a new classifier
accuracy: 0.95973
start to train a new classifier
accuracy: 0.93960
start to train a new classifier
accuracy: 0.77852
start to train a new classifier
accuracy: 0.68456
start to train a new classifier
accuracy: 0.73154
start to train a new classifier
accuracy: 0.72483
start to train a new classifier
accuracy: 0.59060
start to train a new classifier
accuracy: 0.63087
start to train a new classifier
accuracy: 0.21477
real 616.93
user 598.01
sys 11.27
```

Note: the time -p command measures how much time the script consumes

- real indicates the total time spent executing the script.

- user indicates the amount of time the CPU spent executing the script
- sys indicates the amount of time spent in kernel-level functions.

After I changed the shuffle switch in KFold validation, results are:

10-fold, *** feature set reduced ***, and *** with*** shuffle

```
[qiwzhu@h3 review]$ time -p python train_test.py
give me the pickled file:
pickled_1500_reviews
9298
start to train a new classifier
accuracy: 0.77
start to train a new classifier
accuracy: 0.78
start to train a new classifier
accuracy: 0.78
start to train a new classifier
accuracy: 0.78
start to train a new classifier
accuracy: 0.78
start to train a new classifier
accuracy: 0.77
start to train a new classifier
accuracy: 0.78
start to train a new classifier
accuracy: 0.77
start to train a new classifier
accuracy: 0.77
start to train a new classifier
accuracy: 0.78
real 1508.51
user 1489.18
sys 9.97
```

10-fold, *** feature set not reduced*** , *** without*** shuffle:

```
[qiwzhu@h3 review]$ time -p python train_test.py
give me the pickled file:
pickled_1500_reviews
12463
start to train a new classifier
accuracy: 0.49
start to train a new classifier
accuracy: 1.00
start to train a new classifier
accuracy: 0.97
start to train a new classifier
accuracy: 0.80
start to train a new classifier
accuracy: 0.71
start to train a new classifier
accuracy: 0.71
start to train a new classifier
accuracy: 0.77
start to train a new classifier
accuracy: 0.59
start to train a new classifier
accuracy: 0.65
start to train a new classifier
accuracy: 0.23
real 917.08
user 891.64
sys 1.61
```

10-fold, *** feature set not reduced*** , *** with*** shuffle:

```
[qiwzhu@h3 review]$ time -p python train_test.py
give me the pickled file:
pickled_1500_reviews
12463
start to train a new classifier
accuracy: 0.80
start to train a new classifier
accuracy: 0.80
start to train a new classifier
accuracy: 0.81
start to train a new classifier
accuracy: 0.80
start to train a new classifier
accuracy: 0.80
start to train a new classifier
accuracy: 0.80
start to train a new classifier
accuracy: 0.80
start to train a new classifier
accuracy: 0.79
start to train a new classifier
accuracy: 0.80
start to train a new classifier
accuracy: 0.80
real 1970.73
user 1844.01
sys 1.60
```

5-fold cross validation results, *** feature set reduced *** , *** with shuffle *** :

```
[qiwzhu@h3 review]$ time -p python train_test.py
give me the pickled file:
pickled_1500_reviews
9298
start to train a new classifier
accuracy: 0.77
start to train a new classifier
accuracy: 0.78
start to train a new classifier
accuracy: 0.77
start to train a new classifier
accuracy: 0.77
start to train a new classifier
accuracy: 0.78
real 907.14
user 888.72
sys 11.03
```

The Most Informative Features:

```
Most Informative Features
         contains(worst) = True           negati : positi =     30.3 :
1.0
          contains(rude) = True           negati : positi =     15.8 :
1.0
         contains(grill) = True           positi : negati =     15.7 :
1.0
          contains(paid) = True           negati : positi =     12.3 :
1.0
         contains(woman) = True           negati : positi =     10.3 :
1.0
        contains(brunch) = True           neutra : negati =      9.7 :
1.0
           contains(gab) = True           positi : neutra =      9.0 :
1.0
         contains(guess) = True           negati : positi =      8.7 :
1.0
          contains(pass) = True           negati : positi =      8.3 :
1.0
       contains(hostess) = True           negati : positi =      8.2 :
1.0
```

5-fold cross validation results, *** feature set not reduced*** and *** with *** shuffle:

```
[qiwzhu@h3 review]$ time -p python train_test.py
give me the pickled file:
pickled_1500_reviews
12463
start to train a new classifier
accuracy: 0.80
start to train a new classifier
accuracy: 0.79
start to train a new classifier
accuracy: 0.80
start to train a new classifier
accuracy: 0.79
start to train a new classifier
accuracy: 0.80
real 1300.89
user 1291.69
sys 1.55
```

The Most Informative Features:

```
Most Informative Features
         contains(worst) = True           negati : positi =     30.3 :
1.0
     contains(fantastic) = True           positi : negati =     22.3 :
1.0
          contains(rude) = True           negati : positi =     15.8 :
1.0
         contains(grill) = True           positi : negati =     15.7 :
1.0
      contains(carnegie) = True           positi : neutra =     14.2 :
1.0
       contains(grilled) = True           neutra : negati =     12.4 :
1.0
         contains(awful) = True           negati : positi =     12.3 :
1.0
          contains(paid) = True           negati : positi =     12.3 :
1.0
        contains(crispy) = True           neutra : negati =     11.1 :
1.0
      contains(attitude) = True           negati : positi =     10.3 :
1.0
```

5-fold cross validation results, *** feature set not reduced *** and *** without *** shuffle:

```
[qiwzhu@h3 review]$ time -p python train_test.py
give me the pickled file:
pickled_1500_reviews
12463
start to train a new classifier
accuracy: 0.00000
start to train a new classifier
accuracy: 0.88963
start to train a new classifier
accuracy: 0.70903
start to train a new classifier
accuracy: 0.68227
start to train a new classifier
accuracy: 0.06689
real 586.04
user 555.47
sys 1.54
```

The Most Informative Features:

```
Most Informative Features
         contains(worst) = True           negati : positi =     30.3 :
1.0
     contains(fantastic) = True           positi : negati =     22.3 :
1.0
          contains(rude) = True           negati : positi =     15.8 :
1.0
         contains(grill) = True           positi : negati =     15.7 :
1.0
      contains(carnegie) = True           positi : neutra =     14.2 :
1.0
       contains(grilled) = True           neutra : negati =     12.4 :
1.0
         contains(awful) = True           negati : positi =     12.3 :
1.0
          contains(paid) = True           negati : positi =     12.3 :
1.0
        contains(crispy) = True           neutra : negati =     11.1 :
1.0
      contains(attitude) = True           negati : positi =     10.3 :
1.0
```

5-fold cross validation results, *** feature set reduced *** and *** without *** shuffle:

```
[qiwzhu@h3 review]$ time -p python train_test.py
give me the pickled file:
pickled_1500_reviews
9298
start to train a new classifier
accuracy: 0.30872
start to train a new classifier
accuracy: 0.95973
start to train a new classifier
accuracy: 0.93960
start to train a new classifier
accuracy: 0.77852
start to train a new classifier
accuracy: 0.68456
start to train a new classifier
accuracy: 0.73154
start to train a new classifier
accuracy: 0.72483
start to train a new classifier
accuracy: 0.59060
start to train a new classifier
accuracy: 0.63087
start to train a new classifier
accuracy: 0.21477
real 601.67
user 562.05
sys 11.65
```

The Most Informative Features:

```
Most Informative Features
         contains(worst) = True           negati : positi =     30.3 :
1.0
          contains(rude) = True           negati : positi =     15.8 :
1.0
         contains(grill) = True           positi : negati =     15.7 :
1.0
          contains(paid) = True           negati : positi =     12.3 :
1.0
         contains(woman) = True           negati : positi =     10.3 :
1.0
        contains(brunch) = True           neutra : negati =      9.7 :
1.0
           contains(gab) = True           positi : neutra =      9.0 :
1.0
         contains(guess) = True           negati : positi =      8.7 :
1.0
          contains(pass) = True           negati : positi =      8.3 :
1.0
       contains(hostess) = True           negati : positi =      8.2 :
1.0
```

Summary:

	10-f-reduced	**10-f-not-reduced**	**5-f-reduced**	**5-f-not-reduced**
shuffle	0.77~0.78;1508	0.79-0.81;1970	0.77~0.78;907	0.79~0.80;1300
no-shuffle	0.31~0.96;617	0.23~1.00;917	0.21~0.96;601	0.00~0.89;586

Conclusion

- without feature set shuffled, the accuracy distribution is very skewed
- when feature set is shuffled, the accuracy distribution is well balanced
- when word features is reduced by stemming and stop word removal, accuracy

is a little bit lower

- 5-fold validation saves much time
- reduced feature set saves much time, especially when feature set is shuffled
- shuffled feature set consums more time to train.