Homework 7. Naive Baysian - Spam or Ham

Double Click here to edit this cell

- Name: 전기범
- Student ID: 201703091
- Submission date: 2019/06/03

In [1]:

```
%matplotlib inline
import numpy as np
import pandas as pd
import re
from sklearn.metrics import confusion_matrix, classification_report
from collections import Counter
```

In [2]:

```
messages = pd.read_csv("spam_utf8.csv")
```

We have 5572 text messages (747 spams or 4825 hams)

Problem 1 (5 pts): Most Common Words

- Use the following for word-splitting:

```
re.findall("[a-z0-9']+", text.lower())
```

- Find 20 most common spam words

In [3]:

```
messages.head()
```

Out[3]:

	category	text
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...

```
In [4]: spam_text=np.sum(messages.loc[messages['category']=='spam'].text.va
        lues+'\n')
        # print(spam_text)
        spam=re.findall("[a-z0-9']+", spam_text.lower())
        # print(spam)
        spam_cnt=Counter(spam)

        print([(key,value) for key,value in sorted(dict(spam_cnt).items(),
        key =lambda x:x[1], reverse=True)][:20])
        # YOUR CODE MUST BE HERE
```

```
[('to', 688), ('a', 378), ('call', 355), ('you', 290), ('your', 26
4), ('free', 224), ('2', 206), ('the', 206), ('for', 204), ('now',
202), ('or', 188), ('u', 169), ('txt', 163), ('is', 158), ('on', 1
44), ('ur', 144), ('4', 137), ('have', 135), ('from', 131), ('mobi
le', 127)]
```

Your output should be like the following:

```
[('to', 688), ('a', 378), ('call', 355), ('you', 290), ('your', 264), (
'free', 224), ('2', 206), ('the', 206), ('for', 204), ('now', 202), ('o
r', 188), ('u', 169), ('txt', 163), ('is', 158), ('on', 144), ('ur', 14
4), ('4', 137), ('have', 135), ('from', 131), ('mobile', 127)]
```

- Find 20 most common ham words

```
In [5]: ham_text="\n".join(str(h) for h in messages.loc[messages['category'
        ]=='ham'].text.values)
        ham=re.findall("[a-z0-9']+", ham_text.lower())
        ham_cnt=Counter(ham)
        print([(key,value) for key,value in sorted(dict(ham_cnt).items(), k
        ey =lambda x:x[1], reverse=True)][:20])
        # YOUR CODE MUST BE HERE
```

```
[('i', 2281), ('you', 1836), ('to', 1518), ('the', 1100), ('a', 10
46), ('u', 980), ('and', 843), ('in', 802), ('me', 758), ('my', 73
4), ('is', 709), ('it', 602), ('of', 513), ('for', 490), ('that',
486), ('have', 430), ('but', 427), ('so', 422), ('are', 408), ('yo
ur', 407)]
```

Your output should be like the following:

```
[('i', 2281), ('you', 1836), ('to', 1518), ('the', 1100), ('a', 1046),
('u', 980), ('and', 843), ('in', 802), ('me', 758), ('my', 734), ('is',
709), ('it', 602), ('of', 513), ('for', 490), ('that', 486), ('have', 4
30), ('but', 427), ('so', 422), ('are', 408), ('your', 407)]
```

Problem 2 (10 pts): My First Simplest Spam Filter

I designed a very simple spam classifer:

- If the text contains one of the spam words, it is a spam
- otherwise, it is a ham

Note that:

- Use the following for word-splitting:

```
re.findall("[a-z0-9']+", text.lower())
```

For each spam words:

```
['call']
['call', 'free']
['call', 'free', 'txt']
```

- **Compute confusion matrix, precision, recall, f1-score using all data as a test dataset**

In [6]:

```python
import re
texts = [str(text) for text in messages['text']]

class MySimplestSpamFilter(object):

    def fit(self, spam_words):
        # FILL OUT
        self.spam_words = spam_words

    def predict(self, texts):
        res = []
        # FILL OUT
        for text in texts:
            words = [word for word in re.findall("[a-z0-9']+", text.lower())]
            if set(self.spam_words) & set(words):
                res.append('spam')
            else:
                res.append('ham')
        return res
```

In [7]:

```
# RUN THIS CELL
model = MySimplestSpamFilter()
model.fit(['call'])

y_pred = model.predict(texts)
y_true = messages['category']

print(confusion_matrix(y_true, y_pred, labels=['spam', 'ham']).T)
print(classification_report(y_true, y_pred, labels=['spam', 'ham'])
)
```

```
[[ 328  218]
 [ 419 4607]]
              precision    recall  f1-score   support

        spam       0.60      0.44      0.51       747
         ham       0.92      0.95      0.94      4825

   micro avg       0.89      0.89      0.89      5572
   macro avg       0.76      0.70      0.72      5572
weighted avg       0.87      0.89      0.88      5572
```

YOUR OUTPUT SHOULD BE:

```
[[ 328  218]
 [ 419 4607]]
             precision    recall  f1-score   support

       spam       0.60      0.44      0.51       747
        ham       0.92      0.95      0.94      4825

avg / total       0.87      0.89      0.88      5572
```

```
In [8]: # RUN THIS CELL
        model = MySimplestSpamFilter()
        model.fit(['call', 'free'])

        y_pred = model.predict(texts)
        y_true = messages['category']

        print(confusion_matrix(y_true, y_pred, labels=['spam', 'ham']).T)
        print(classification_report(y_true, y_pred, labels=['spam', 'ham'])
        )
```

```
[[ 438  265]
 [ 309 4560]]
              precision    recall  f1-score   support

        spam       0.62      0.59      0.60       747
         ham       0.94      0.95      0.94      4825

   micro avg       0.90      0.90      0.90      5572
   macro avg       0.78      0.77      0.77      5572
weighted avg       0.89      0.90      0.90      5572
```

YOUR OUTPUT SHOULD BE:

```
[[ 438  265]
 [ 309 4560]]
             precision    recall  f1-score   support

       spam       0.62      0.59      0.60       747
        ham       0.94      0.95      0.94      4825

avg / total       0.89      0.90      0.90      5572
```

In [9]:

```
# RUN THIS CELL
model = MySimplestSpamFilter()
model.fit(['call', 'free', 'txt'])

y_pred = model.predict(texts)
y_true = messages['category']

print(confusion_matrix(y_true, y_pred, labels=['spam', 'ham']).T)
print(classification_report(y_true, y_pred, labels=['spam', 'ham'])
)
```

```
[[ 532  277]
 [ 215 4548]]
              precision    recall  f1-score   support

        spam       0.66      0.71      0.68       747
         ham       0.95      0.94      0.95      4825

   micro avg       0.91      0.91      0.91      5572
   macro avg       0.81      0.83      0.82      5572
weighted avg       0.92      0.91      0.91      5572
```

YOUR OUTPUT SHOULD BE:

```
[[ 532  277]
 [ 215 4548]]
             precision    recall  f1-score   support

       spam       0.66      0.71      0.68       747
        ham       0.95      0.94      0.95      4825

avg / total       0.92      0.91      0.91      5572
```

Problem 3 (5 pts): My Another Simplest Spam Filter

I designed another very simple spam classifer:

- If the text contains **all of the spam words**, it is a spam
- otherwise, it is a ham

Note that:

- Use the following for word-splitting:

```
re.findall("[a-z0-9']+", text.lower())
```

For this set of spam words:

```
['call', 'free', 'txt']
```

- **Compute confusion marix, precision, recall, f1-score using all data as a test dataset**

In [10]:

```
import re

class MyAnotherSimplestSpamFilter(object):
    def fit(self, spam_words):
        self.spam_words = spam_words

    def predict(self, texts):
        res = []
        for text in texts:
            words = [word for word in re.findall("[a-z0-9']+", text
.lower())]
            if set(self.spam_words) <= set(words):
                res.append('spam')
            else:
                res.append('ham')
        return res
```

In [11]:

```
# RUN THIS CELL
model = MyAnotherSimplestSpamFilter()
model.fit(['call', 'free', 'txt'])

y_pred = model.predict(texts)
y_true = messages['category']

print(confusion_matrix(y_true, y_pred, labels=['spam', 'ham']).T)
print(classification_report(y_true, y_pred, labels=['spam', 'ham'])
)
```

```
[[   6    0]
 [ 741 4825]]
              precision    recall  f1-score   support

        spam       1.00      0.01      0.02       747
         ham       0.87      1.00      0.93      4825

   micro avg       0.87      0.87      0.87      5572
   macro avg       0.93      0.50      0.47      5572
weighted avg       0.88      0.87      0.81      5572
```

YOUR OUTPUT SHOULD BE:

```
[[   6    0]
 [ 741 4825]]
             precision    recall  f1-score   support

       spam       1.00      0.01      0.02       747
        ham       0.87      1.00      0.93      4825

avg / total       0.88      0.87      0.81      5572
```

Problem 4. Discussion (5 pts)

- From the Problem 2, 3 experiments, discuss about precision and recall trade-off.
- WRITE HERE (To edit, double click this cell)

recall은 내가 모델로부터 얻은 결과중에 얼마나 정답이 포함되었는가 여부이고, precision은 모델에서 얻은 결과중에 얼마나 정답일 가능성이 높은가이다. 두 가지 용어는 서로 trade off 관계이다.

단어의 수가 증가함에 따라 Spam 판정의 확률이 높아지는것을 보아 Spam Filter를 위해 단어의 수를 증가시키면 더 정확도가 높은 필터링이 되는것을 확인 할 수 있었습니다

Problem 5. SMS Spam Filter (20 pts)

- Using sklearn modules, implement your SMS spam filter
- Print confusion matrix, classification report, **f0.5 score**
- This homework will be graded based on **f0.5 score**

In [12]:

```
# DO NO EDIT THIS CELL
import numpy as np

np.random.seed(0)
```

In [13]:

```python
# YOU CODE MUST BE HERE

# import whatever module as you need
from sklearn.naive_bayes import BernoulliNB
from sklearn.metrics import fbeta_score
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer

# You must have 30% stratified sample from dataset for test dataset like the following:

vectorizer = CountVectorizer(stop_words = "english", lowercase=True, binary=False)  # default actually
vecs = vectorizer.fit_transform(texts)  # it is a sparse matrix representation

targets = messages["category"].apply(lambda c: 1 if c == 'spam' else 0)

X_train, X_test, y_train, y_test = train_test_split(vecs, targets, stratify = targets, test_size = 0.3)

bernoulli_NB = BernoulliNB()
bernoulli_NB.fit(X_train, y_train)

y_pred = bernoulli_NB.predict(X_test)   # default binarize=0.0

print(confusion_matrix(y_test, y_pred, labels=[1, 0]).T)
print(classification_report(y_test, y_pred, labels=[1, 0]))
beta = 0.5
print('f_{} score is {:6.2f}%'.format(beta, fbeta_score(y_test, y_pred, beta = beta)*100))
```

```
[[ 183    8]
 [  41 1440]]
              precision    recall  f1-score   support

           1       0.96      0.82      0.88       224
           0       0.97      0.99      0.98      1448

   micro avg       0.97      0.97      0.97      1672
   macro avg       0.97      0.91      0.93      1672
weighted avg       0.97      0.97      0.97      1672

f_0.5 score is  92.61%
```

Problem 6. MyBernoulliNaiveBayesClassifier (40 pts)

- Implement your own Bernoulli Naive Bayes Classifier

In [ ]:

```
class MyBernoulliNaiveBayesClassifier(object):
    def fit(self, X, Y, smoothing=1):
    # FILL OUT

    def predict(self, X):
        res = []
    # FILL OUT
        return res
```

In [ ]:

```
# RUN THIS CELL

import numpy as np
np.random.seed(0)
X = np.random.randint(2, size=(6, 100))
y = np.array([1, 2, 3, 3, 1, 0])
my_nbc = MyBernoulliNaiveBayesClassifier()
my_nbc.fit(X, y)

y_pred = my_nbc.predict(X)

print(confusion_matrix(y, y_pred).T)
print(classification_report(y, y_pred))
```

YOUR OUTPUT MUST BE LIKE THE FOLLOWING:

```
[[1 0 0 0]
 [0 2 0 0]
 [0 0 1 0]
 [0 0 0 2]]
             precision    recall  f1-score   support

          0       1.00      1.00      1.00         1
          1       1.00      1.00      1.00         2
          2       1.00      1.00      1.00         1
          3       1.00      1.00      1.00         2

avg / total       1.00      1.00      1.00         6
```

Ethics:

If you cheat, you will get negatgive of the total points. If the homework total is 22 and you cheat, you get -22.

What to submit

- Run all cells
- Goto "File -> Print Preview"
- Print the page
- Submit in class
- No late homeworks accepted
- Your homework will be graded on the basis of correctness and programming skills

Deadline: 6/3