Lab 9 Project: Naive Bayes for Natural Language Processing

In this project, we show how to handle text data with scikit-learn. Working with text requires careful preprocessing and feature extraction. It is also quite common to deal with highly sparse matrices.

ATTENTION: In this project, you will need to read the documents of several very important modules, such as grid_search, train_test_split, etc. I've already given you the code for the most difficult part, if you don't understand some of the terminologies, try googling it, or quora it. Good Luck!

We will learn to recognize whether a comment posted during a public discussion is considered insulting to one of the participants. We will use a labeled dataset from Impermium (https://impermium.com), released during a Kaggle competition (https://www.kaggle.com/c/detecting-insults-in-social-commentary).

1. Let's import our libraries.

In [4]:
```
import numpy as np
import pandas as pd
import sklearn
import sklearn.model_selection as ms

#import sklearn.grid_search as gs
#from sklearn.model_selection import GridSearchCV as gs

import sklearn.feature_extraction.text as text
import sklearn.naive_bayes as nb
import matplotlib.pyplot as plt
%matplotlib inline
```

2. Let's open the csv file with Pandas.

In [6]:
```
df = pd.read_csv("troll.csv")
```

3. Each row is a comment. There are three columns: whether the comment is insulting (1) or not (0), the data, and the unicode-encoded contents of the comment. In the next cell, **print several entries of the 'Insult' and 'Comment' column**

In [9]:
```
#Finish me
df[['Insult', 'Comment']].head()
```

Out[9]:

	Insult	Comment
0	1	"You fuck your dad."
1	0	"i really don't understand your point.\xa0 It ...
2	0	"A\\xc2\\xa0majority of Canadians can and has ...
3	0	"listen if you dont wanna get married to a man...
4	0	"C\xe1c b\u1ea1n xu\u1ed1ng \u0111\u01b0\u1edd...

4. Now, we are going to define the feature matrix $\mathbf{X}$ and the labels $\mathbf{y}$. First, define the label `y` from the dataframe:

In [10]:
```
#Finish me
y = df['Insult']
```

Obtaining the feature matrix from the text is not trivial. Scikit-learn can only work with numerical matrices. How to convert text into a matrix of numbers? A classical solution is to first extract a **vocabulary**: a list of words used throughout the corpus. Then, we can count, for each sample, the frequency of each word. We end up with a **sparse matrix**: a huge matrix containiny mostly zeros.

However in a large text corpus, some words will be very present (e.g. “the”, “a”, “is” in English) hence carrying very little meaningful information about the actual contents of the document. If we were to feed the direct count data directly to a classifier those very frequent terms would shadow the frequencies of rarer yet more interesting terms. In order to re-weight the count features into floating point values suitable for usage by a classifier it is very common to use the tf–idf transform.

For more explanation: http://scikit-learn.org/stable/modules/feature_extraction.html#tfidf-term-weighting (http://scikit-learn.org/stable/modules/feature_extraction.html#tfidf-term-weighting)

Here, we can use the module in `text.TfidfVectorizer`, use it to define the data `X`, and print the shape of it, you may use the `TfidfVectorizer.fit_transform` method

In [11]:
```
tf = text.TfidfVectorizer()
X = tf.fit_transform(df['Comment']) #finish me
print(X.shape)
```

```
(3947, 16469)
```

5. There are 3947 comments and 16469 different words. Let's estimate the sparsity of this feature matrix.

In [12]:
```
print("Each sample has ~{0:.2f}% non-zero features.".format(
          100 * X.nnz / float(X.shape[0] * X.shape[1])))
```

```
Each sample has ~0.15% non-zero features.
```

6. Now, we are going to train a classifier as usual. **We first split the data into a train and test set, using `ms.train_test_split` , the size of testset is 20% of the whole data**

In [30]:
```
(X_train, X_test, y_train, y_test) = ms.train_test_split(X, y, test_size = 0.20) #finish me
```

7. Now, we are going to train the model with cross-validation. We use a **Bernoulli Naive Bayes classifier** with a **grid search cross-valiation** (use `gs.GridSearchCV`), which means we want to use cross-validation to find the best parameter for this BernoulliNB model. And the only parameter of Bernoulli NB we want to tune here is 'alpha' (which is a parameter related to Laplace Smoothing, so you should set the `para_grid` to be `{'alpha':np.logspace(-2., 2., 50)}` , here we search the best `alpha` in the range of (-2, 2)

In [31]:
```
bnb = ms.GridSearchCV(nb.BernoulliNB(), param_grid = {'alpha': np.logspace(-2., 2., 50)})
bnb.fit(X_train, y_train);
```

```
C:\Users\Sarah\Anaconda3\lib\site-packages\sklearn\model_selection\_split.py:2053: FutureWarning: You should specify a value for 'cv' instead of relying on the default value. The default value will change from 3 to 5 in version 0.22.
  warnings.warn(CV_WARNING, FutureWarning)
```

8. What is the performance of this classifier on the test dataset? Print the performance by Using `GridSearchCV.score` , it shows the classification accuracy of this Naive Bayes Model on the given test dataset.

In [32]:
```
bnb.score(X_test, y_test) #Finish me
```

Out[32]:
```
0.7810126582278482
```

9. Let's take a look at the words corresponding to the largest coefficients (the words we find frequently in insulting comments).

In [33]:
```
# We first get the words corresponding to each feature.
names = np.asarray(tf.get_feature_names())
# Next, we display the 50 words with the largest
# coefficients.
print(','.join(names[np.argsort(bnb.best_estimator_.coef_[0,:])[::-1][:50]]))
```

```
you,your,are,the,to,and,of,that,is,it,in,like,on,have,re,not,for,just,so,xa0,all,an,idiot,with,be,get,fuck,do,what,go,this,don,up,no,as,can,stupid,or,but,know,if,ass,bitch,who,about,because,little,me,was,people
```

10. Finally, let's test our estimator on a few test sentences. the result should be `[0 1 1]`

In [34]:
```
print(bnb.predict(tf.transform([
    "I totally agree with you.",
    "You are so stupid.",
    "I love you."
    ])))
```

```
[0 1 0]
```

Congratulations! You just finished a project on Natural Language Processing!