OFFICE HOUR

- Wednesday 3 - 5 pm
- DBH 4013
Return the training and test dataset, including only the top 5 categories “in training data”

First load the whole ‘training’ dataset
**PROBLEM 1**

```python
def load_news_dataset(n_categories):
    n_categories = 5
```

from sklearn.datasets import fetch_20newsgroups
from nltk import FreqDist

twenty_train_all = fetch_20newsgroups(subset='train')

```python
>>> type(twenty_train_all)
<class 'sklearn.datasets.base.Bunch'>

>>> twenty_train_all.data
["From: lerxst@wam.umd.edu (where's my thing)\nSubject: WHAT car is this!?\nNntp-Posting-Host: rac3.wam.umd.edu\nOrganization: University of Maryland, College Park\nNLines: 15\n\nI was wondering if anyone out there could enlighten . . . . .

>>> twenty_train_all.target
array([7, 4, 4, ..., 3, 1, 8])

>>> twenty_train_all.target_names
['alt.atheism', 'comp.graphics', 'comp.os.ms-windows.misc',
 'comp.sys.ibm.pc.hardware', 'comp.sys.mac.hardware', 'comp.windows.x',
 'misc.forsale', 'rec.autos', 'rec.motorcycles', 'rec.sport.baseball',
 'rec.sport.hockey', 'sci.crypt', 'sci.electronics', 'sci.med', 'sci.space',
 'soc.religion.christian', 'talk.politics.guns', 'talk.politics.mideast',
 'talk.politics.misc', 'talk.religion.misc']
```
PROBLEM 1

def load_news_dataset(n_categories):
    n_categories = 5

    # Return the training and test dataset, including only the top 5 categories “in training data”
    # First load the whole 'training' dataset
    # Get the 5 most frequent categories
    # Load training and test data, including only those categories

    train_data = fetch_20newsgroups(subset='train', categories=category_names, ...)

>>> category_names
['rec.sport.hockey', 'soc.religion.christian', 'rec.motorcycles', 'rec.sport.baseball', ..]
PROBLEM 2

```python
def extract_text_features(train_data, test_data):

    tfidf_transformer = TfidfTransformer()

    tfidf_transformer = tfidf_transformer.fit(X_train_counts)
    X_train_tfidf = tfidf_transformer.transform(X_train_counts)

    X_test_tfidf = tfidf_transformer.transform(X_test_counts)
```

- Use CountVectorizer, TfidfTransformer
PROBLEM 2  

```python
def extract_text_features(train_data, test_data):

    tfidf_transformer = TfidfTransformer()

    tfidf_transformer = tfidf_transformer.fit(X_train_counts)
    X_train_tfidf = tfidf_transformer.transform(X_train_counts)

    X_train_tfidf = tfidf_transformer.fit_transform(X_train_counts)

    X_test_tfidf = tfidf_transformer.transform(X_test_counts)
```
PROBLEM 2

```python
def extract_text_features(train_data, test_data):

    tfidf_transformer = tfidf_transformer.fit(X_train_counts)
    X_train_tfidf = tfidf_transformer.transform(X_train_counts)

>>> X_train_tfidf
<2989x39851 sparse matrix of type '<class 'numpy.float64'>'
    with 377580 stored elements in Compressed Sparse Row format>

>>> X_train_tfidf[0,:]
<1x39851 sparse matrix of type '<class 'numpy.float64'>'
    with 48 stored elements in Compressed Sparse Row format>
```
def fit_and_predict_BernoulliNB(X_train, Y_train, X_test):

X_train: scipy sparse matrix
Data for training (matrix with features, e.g. BOW or tf-idf)

Y_train: numpy.ndarray
Labels for training data (target value)

X_test: scipy sparse matrix
Test data used for prediction
PROBLEM 4

def pipeline_BernoulliNB(train_data, Y_train, X_test_data):

def extract_text_features(train_data, test_data):

+ 

def fit_and_predict_BernoulliNB(X_train, Y_train, X_test):

Code for pipeline_BernoulliNB() is given on Piazza
BERNOULLI NAIVE BAYES CLASSIFIER

- TF-IDF feature is not ideal for Bernoulli NB
  - Binarize = 0.5 (Prob 3)
    - It’s normal if you get misclassified result
  - Binarize = 0.0 (Prob 4)
    - Thresholding with 0 is ignoring IDF weights
    - Same as binarizing Count features!
**BERNOULLI NAIVE BAYES CLASSIFIER**

- TF-IDF feature is not ideal for Bernoulli NB
  - Binarize = 0.5 (Prob 3)
    - Accuracy with threshold 0.5: 0.2163
  - Binarize = 0.0 (Prob 4)
    - Accuracy with threshold 0.0: 0.9079

- Bernoulli NB with Count features (BOW)
  - Accuracy with threshold 0: 0.9079