Nick Greenquist
HW2b

MultiClass Classification of 20NewsGroups

The Dataset

1. **Loading Data**
    a. Rather than loading the datasets from files, I utilized sklearns built in function that loads it for you. Before creating dataframes in Spark, I loaded in the datasets in a separate helper method. Once they were loaded in, the train and test portions were placed into dataframes.
2. **Trimming the Fat - Removing metadata and other unwanted portions**
    a. Headers, footers, and quotes were removed from the train and test data as they were loaded in. The code below shows how this is done:
    b. `data = fetch_20newsgroups(subset=train_or_test, shuffle=True, remove=('headers', 'footers', 'quotes'))`
3. **Categories**
    a. I used all 20 of the categories. Because Spark ml library requires dataframes with labels of DoubleType, the categories were mapped to a number between 0 and 19. Below I have the categories with their DoubleType label:
    b.
```
0: alt.atheism
1: Comp.graphics
2: Comp.os.ms-windows.misc
3: comp.sys.ibm.pc.hardware
4: comp.sys.mac.hardware
5: comp.windows.x
6: misc.forsale
7: rec.autos
8: rec.motorcycles
9: rec.sport.baseball
10: rec.sport.hockey
11: sci.crypt
12: sci.electronics
13: sci.med
14: sci.space
15: soc.religion.christian
16: talk.politics.guns
17: talk.politics.mideast
```

```
18: talk.politics.misc
19: talk.religion.misc
```

Spark - pyspark.ml Multinomial Naive Bayes

1. **Code:** classify.py in deliverables folder.
    a. Run instructions can be found in the code
2. **Document Vectorization**
    a. **Bag of Words:** Bag of Words failed on Spark. Due to a matrix multiplication error, predicting models on a test vector did not work. I abandoned this approach because TF-IDF is better overall. The code for this is left in the file and is commented out if you want to see how it would be done.
    b. **TF-IDF:** This was the main vectorization method I went after to perform classification. Many challenges came one after another on Spark. Due to the ML library, Spark require all the data to be wrangled into DataFrames. Much of the logic can be seen in the code itself, but I will outline it here. I have outlines the steps before to get dataframes turned into TF-IDF dataframes:
        i. Load labels and data and zip and map into correct rdd format
        ii. Turn each train and test rdd into a dataframe with the correct column format
        iii. Run the text of each dataframe through the preprocessor function to clean up text
            1. Lower case, trim non alpha-numeric chars, remove single char words
        iv. Tokenize the text of each data frame using Tokenizer
        v. Transform each dataframe's tokens column to a term-frequency representation using HashingTF
        vi. Use the IDF (inverse document frequency) library to computer the IDF of all of the words in the term-frequency data frame
    c. Stopwords?
        i. I did not remove stop words for two reasons: First, nltk is not supported on Spark and thus the easiest way to remove stop words is not possible. Second, TF-IDF basically neutralizes stop words as they appear in so many documents that they no longer hold any weight.
2. **Classification:**
    a. I used Multinomial Naive Bayes to classify the tf-idf vector representation of the test dataframe using a model fit with the train tf-idf dataframe.
    b. Spark's ML library made this very easy and the code snippet is here:
        i. `nb = NaiveBayes(smoothing=1.0, modelType="multinomial", featuresCol="features", labelCol="label")`
        ii. `model = nb.fit(train_tf_idf)`

iii. `result = model.transform(test_tf_idf)`

3. **Results:**
    a. **How I got them:** I used the ML library's MulticlassClassificationEvaluator class to grab the metrics of how NaiveBayes performed. In the main class, after fitting the model and predicting the labels, I saved the tuples of (actualLabel,predictedLabel) into a seperate text file. This helped separate the evaluation code from the main vectorization and classification code.
        i. **Code:** eval.py in the deliverable
        ii. **Actual and Predicted labels:** naivebayes_results.txt in the deliverable
    b. **Metrics of NaiveBayes on 20NewsGroups:**
        i. **F1 Score:** 0.640040152074
        ii. **Precision**: 0.674540502419
        iii. **Recall:** 0.654938927244
        iv. **Accuracy:** Not supported on Spark 1.6.0 ML library
4. **Other classifiers:**
    a. Spark 1.6.0 has a very limited assortment of classifiers in the ML library. None support Bag of Words or TF-IDF representations either. I implemented more classifiers below in local code using sklearn.

Local - sklearn

1. **Code:** classify_local.py in deliverables folder...'python classify_local.py' will run the code
2. **The Dataset**
    a. **Loading:** Again I used the 20newsgroups dataset and loaded it in through sklearn's built in dataset class. This was even easier than in Spark as I did not have to convert to RDD or dataframe at any step.
    b. **Trimming the Fat - Removing metadata and other unwanted portions**
        i. Exactly the same as it was done in Spark:
    c. **Preprocessing:** Less was needed to be done for sklearn. Sklearn's vectorizer automatically lowercase and tokenize your text. All I did was trim non alphanumeric characters from the test and train text data.
3. **Document Vectorization**
    a. **Methods:** Two methods were used for representing documents as vectors. **Bag of Words** and **TF-IDF**. In sklearn, Bag of Words was achieved by passing the documents through the CountVectorizer library and TF-IDF was achieved by passing the Bag of Words representation through the sklearn TfidfTransformer library. It was possible to run classification on just Bag of Words or TF-IDF by simply including or excluding the other vectorizer in the classification pipeline (explained below).
    b. **Stopwords?**
        i. Stop words are easily filtered in sklearn by simply passing english stop words as a parameter to CountVectorizer (Bag of Words):

ii. `CountVectorizer(stop_words='english'))`

4. **Classification Pipeline**
    a. Sklearn provides an incredibly powerful tool through the Pipeline library. With this, you can simply pass vectorizers, classifiers, preprocessors, etc as parameters to a Pipeline object. This greatly decreases code length and complexity as many lines of code are condensed into one object call. On top of this, you can also plug and play any of these tools you want into the Pipeline with minimal effort (once you learn how to use Pipeline correctly!). Below I have outlined each of the classifiers I used in sklearn. Comparison to the results from Spark will be detailed in the next section.
    b. **Multinomial Naive Bayes**
        i. This is the same classifier I used on Spark. A very simple classifier that works using Bayes principles and is suited for text classification when given word counts. TF-IDF can also be used but this works well with simple word counts. We can see this behaviour from how the classifier does not perform much better when given a tf-idf representation. This is because Naive Bayes handles simple representations more effectively.
    c. **SGD Classifier**
        i. Stochastic Gradient Descent classifier. This classifier uses an linear SVM (support vector machine) by default which is a very powerful classifier algorithm. The SGD increases the power of the SVM by estimating the gradient of the loss of each sample and updates the model with a decreasing learning rate
    d. **OneVsOne Classifier**
        i. A powerful classifier that lets you fit a custom classifier that will then compare each label vs all other labels. Although this is the slowest algorithm O(n^2) dues to having to compare every category vs every other category. I chose to pass in a LinearSVC classifier to this pipeline (support vector classifier).
        ii. Interestingly this classifier improved the most from the transition to Bag of Words to tf-idf (almost 20%).

5. **Measuring Performance**
    a. I used sklearn's metrics library to visualize a rich suite of results of each classifier. Sadly Spark did not have this many tools for results. I also used numpy's mean function to grab a quick basic accuracy measure (how many correct predictions over all predictions) using this line:
        i. `results = np.mean(predicted == test.target)`
        ii. `print("Basic Accuracy: " + str(results))`
    b. I used sklearn's metric library in the following way:
        i. `print(metrics.classification_report(test.target, predicted, target_names=test.target_names))`

6. Results

a. Multinomial Naive Bayes:

i. Bag of Words:

```
Multinomial Naive Bayes
Basic Accuracy: 0.638077535847
                          precision    recall  f1-score   support

             alt.atheism       0.60      0.36      0.45       319
           comp.graphics       0.58      0.69      0.63       389
 comp.os.ms-windows.misc       0.50      0.01      0.02       394
comp.sys.ibm.pc.hardware       0.50      0.72      0.59       392
   comp.sys.mac.hardware       0.71      0.56      0.63       385
          comp.windows.x       0.57      0.79      0.66       395
            misc.forsale       0.85      0.65      0.74       390
               rec.autos       0.83      0.69      0.75       396
         rec.motorcycles       0.89      0.64      0.75       398
      rec.sport.baseball       0.94      0.76      0.84       397
        rec.sport.hockey       0.58      0.92      0.71       399
               sci.crypt       0.58      0.78      0.67       396
         sci.electronics       0.67      0.50      0.57       393
                 sci.med       0.82      0.79      0.80       396
               sci.space       0.75      0.75      0.75       394
  soc.religion.christian       0.48      0.90      0.63       398
      talk.politics.guns       0.56      0.65      0.60       364
   talk.politics.mideast       0.64      0.78      0.71       376
      talk.politics.misc       0.43      0.44      0.44       310
      talk.religion.misc       0.45      0.06      0.11       251

             avg / total       0.65      0.64      0.61      7532
```

ii. TF-IDF

```
Multinomial Naive Bayes
Basic Accuracy: 0.674986723314
                          precision    recall  f1-score   support

             alt.atheism       0.77      0.19      0.31       319
           comp.graphics       0.66      0.67      0.67       389
 comp.os.ms-windows.misc       0.67      0.58      0.63       394
comp.sys.ibm.pc.hardware       0.56      0.74      0.64       392
   comp.sys.mac.hardware       0.76      0.65      0.70       385
          comp.windows.x       0.80      0.76      0.78       395
            misc.forsale       0.79      0.73      0.75       390
               rec.autos       0.84      0.71      0.77       396
         rec.motorcycles       0.86      0.74      0.80       398
      rec.sport.baseball       0.92      0.81      0.86       397
        rec.sport.hockey       0.57      0.94      0.71       399
               sci.crypt       0.59      0.80      0.68       396
         sci.electronics       0.70      0.52      0.59       393
                 sci.med       0.88      0.76      0.81       396
               sci.space       0.78      0.74      0.76       394
  soc.religion.christian       0.38      0.92      0.54       398
      talk.politics.guns       0.57      0.72      0.64       364
   talk.politics.mideast       0.82      0.79      0.80       376
      talk.politics.misc       0.88      0.29      0.44       310
      talk.religion.misc       1.00      0.01      0.02       251

             avg / total       0.73      0.67      0.66      7532
```

b. SGD Classifier:

i. Bag of Words:

```
SGD
Basic Accuracy: 0.608204992034
                          precision    recall  f1-score   support

              alt.atheism       0.50      0.41      0.45       319
            comp.graphics       0.60      0.60      0.60       389
  comp.os.ms-windows.misc       0.58      0.56      0.57       394
 comp.sys.ibm.pc.hardware       0.54      0.58      0.56       392
    comp.sys.mac.hardware       0.56      0.64      0.60       385
           comp.windows.x       0.74      0.64      0.68       395
             misc.forsale       0.67      0.77      0.72       390
                rec.autos       0.55      0.63      0.58       396
          rec.motorcycles       0.54      0.73      0.62       398
       rec.sport.baseball       0.77      0.73      0.75       397
         rec.sport.hockey       0.44      0.91      0.59       399
                sci.crypt       0.76      0.64      0.70       396
          sci.electronics       0.55      0.39      0.46       393
                  sci.med       0.81      0.59      0.68       396
                sci.space       0.75      0.67      0.71       394
   soc.religion.christian       0.71      0.60      0.65       398
       talk.politics.guns       0.57      0.56      0.56       364
    talk.politics.mideast       0.83      0.64      0.72       376
       talk.politics.misc       0.52      0.41      0.46       310
       talk.religion.misc       0.37      0.25      0.30       251

              avg / total       0.62      0.61      0.61      7532
```

ii. TF-IDF:

```
SGD
Basic Accuracy: 0.685342538502
                          precision    recall  f1-score   support

              alt.atheism       0.59      0.40      0.48       319
            comp.graphics       0.70      0.66      0.68       389
  comp.os.ms-windows.misc       0.67      0.61      0.64       394
 comp.sys.ibm.pc.hardware       0.66      0.66      0.66       392
    comp.sys.mac.hardware       0.72      0.68      0.70       385
           comp.windows.x       0.78      0.73      0.75       395
             misc.forsale       0.72      0.78      0.75       390
                rec.autos       0.77      0.69      0.73       396
          rec.motorcycles       0.50      0.79      0.61       398
       rec.sport.baseball       0.81      0.79      0.80       397
         rec.sport.hockey       0.79      0.92      0.85       399
                sci.crypt       0.73      0.74      0.74       396
          sci.electronics       0.66      0.47      0.55       393
                  sci.med       0.75      0.81      0.78       396
                sci.space       0.70      0.78      0.73       394
   soc.religion.christian       0.59      0.84      0.69       398
       talk.politics.guns       0.57      0.69      0.63       364
    talk.politics.mideast       0.78      0.82      0.80       376
       talk.politics.misc       0.66      0.40      0.50       310
       talk.religion.misc       0.51      0.12      0.19       251

              avg / total       0.69      0.69      0.68      7532
```

c. OneVsOne Classifier:

i. Bag of Words:

```
OneVsOneClassifier
Basic Accuracy: 0.486192246415
                         precision    recall  f1-score   support

            alt.atheism       0.31      0.38      0.34       319
          comp.graphics       0.50      0.51      0.51       389
 comp.os.ms-windows.misc      0.52      0.47      0.50       394
comp.sys.ibm.pc.hardware      0.53      0.49      0.51       392
   comp.sys.mac.hardware      0.51      0.45      0.48       385
          comp.windows.x      0.48      0.57      0.52       395
            misc.forsale      0.69      0.72      0.71       390
               rec.autos      0.33      0.55      0.41       396
         rec.motorcycles      0.48      0.51      0.49       398
      rec.sport.baseball      0.51      0.51      0.51       397
        rec.sport.hockey      0.70      0.65      0.68       399
               sci.crypt      0.57      0.46      0.51       396
         sci.electronics      0.41      0.37      0.39       393
                 sci.med      0.56      0.43      0.49       396
               sci.space      0.49      0.50      0.49       394
  soc.religion.christian      0.55      0.58      0.56       398
      talk.politics.guns      0.45      0.42      0.44       364
   talk.politics.mideast      0.59      0.49      0.53       376
      talk.politics.misc      0.30      0.25      0.27       310
      talk.religion.misc      0.27      0.26      0.26       251

              avg / total       0.50      0.49      0.49      7532
```

ii. TF-IDF:

```
OneVsOneClassifier
Basic Accuracy: 0.665427509294
                         precision    recall  f1-score   support

            alt.atheism       0.47      0.47      0.47       319
          comp.graphics       0.60      0.69      0.65       389
 comp.os.ms-windows.misc      0.66      0.61      0.63       394
comp.sys.ibm.pc.hardware      0.67      0.65      0.66       392
   comp.sys.mac.hardware      0.73      0.64      0.68       385
          comp.windows.x      0.83      0.63      0.72       395
            misc.forsale      0.77      0.78      0.78       390
               rec.autos      0.43      0.76      0.55       396
         rec.motorcycles      0.73      0.72      0.72       398
      rec.sport.baseball      0.75      0.75      0.75       397
        rec.sport.hockey      0.91      0.81      0.86       399
               sci.crypt      0.83      0.65      0.73       396
         sci.electronics      0.54      0.61      0.58       393
                 sci.med      0.77      0.73      0.75       396
               sci.space      0.67      0.72      0.70       394
  soc.religion.christian      0.68      0.73      0.70       398
      talk.politics.guns      0.60      0.66      0.63       364
   talk.politics.mideast      0.83      0.72      0.77       376
      talk.politics.misc      0.53      0.46      0.49       310
      talk.religion.misc      0.44      0.31      0.37       251

              avg / total       0.68      0.67      0.67      7532
```

d. **Best Classifier for multiclass classification of text:**
    i. SGDClassifier using TF-IDF is the clear winner, losing only to Naive Bayes in precision
    ii. This makes sense as SVM models are superior in text classification and the combination of Stochastic Gradient Descent makes this an even more lethal combination.

Spark Naive Bayes vs sklearn Naive Bayes

1. Spark performed worse than sklearn but not by much. Of course sklearn is more established and NYU"s HPC runs an older version of Spark and therefore an older version of the ML library so this is expected. However the speed of running classification on a distributed manner makes up for this slightly less accurate result as much more data can be handled than on a local machine using main-memory sklearn tools.
2. Stopword trimming might have been an issue. However, as stated above, NYU's HPC does not support nltk library so trimming stop words effectively is very challenging.

Metric	**Spark pyspark.ml**	**sklearn**
F1 Score	0.64	0.66
Precision	0.67	0.73
Recall	0.65	0.67

References

1. https://spark.apache.org/docs/1.6.0/api/python/pyspark.ml.html
2. https://spark.apache.org/docs/1.6.0/ml-features.html#tf-idf-hashingtf-and-idf
3. https://gist.github.com/mitallast/87f0d0c5a8e5447c1626