20 Newsgroup Document Classification Report

Problem Statement

This project will focuss on classifying the 20 news data using machine learning algorithms such as Naïve bayes, Random forest, SVM etc. It is a supervised classification problem, there are news of 20 categories where each piece of news belongs to one category.

The goal is to extract proper features and build an effective model to assign each piece of news to the correct category.

I will explore the dataset in the beginning on the training part by extracting useful keywords and build vectors of features from the texts of news. Then I will use several classification methods to do classification based upon those vectors and then compare the efficiency of these classifiers on the test data and choose one. I will also use gridsearch to find the best parameters for classifiers.

Dataset

I will apply classification algorithms on 20 newsgroup dataset from CMU Text Learning Group Data Archives. It has a collection of 20,000 messages, collected from 20 different newsgroups. The news will be classified according to their contents

The data is organized into 20 different newsgroups, each corresponding to a different topic. Some of the newsgroups are very closely related to each other
(e.g. comp.sys.ibm.pc.hardware / comp.sys.mac.hardware), while others are highly unrelated (e.g misc.forsale / soc.religion.christian). Here is a list of the 20 newsgroups,

comp.graphics	rec.autos	sci.crypt
comp.os.ms-windows.misc	rec.motorcycles	sci.electronics
comp.sys.ibm.pc.hardware	rec.sport.baseball	sci.med
comp.sys.mac.hardware	rec.sport.hockey	sci.space
comp.windows.x		
	talk.politics.misc	talk.religion.misc
misc.forsale	talk.politics.guns	alt.atheism
	talk.politics.mideast	soc.religion.christian

`20news-bydate.tar.gz` unpacks into two top-level directories: `20news-bydate-train` and `20news-bydate-test`. Both of these contain 20 subdirectories of newsgroup data, where the directory name is the same as the newsgroup name, and each of these contains a set of newsgroup messages, roughly 400—600 posts for the training sets and roughly 250—400 posts for the test sets.

Analysis

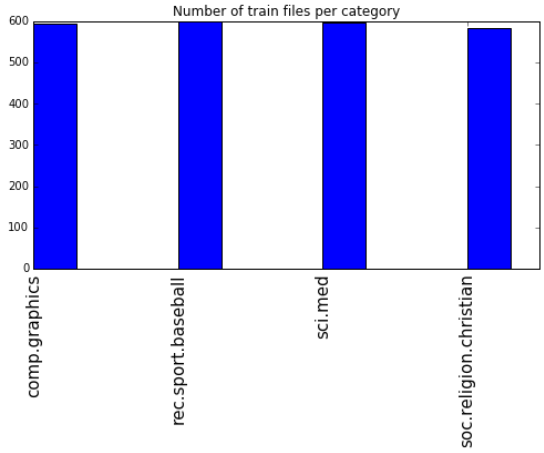
In this project, I am showing 4 classes for data exploration and visualization, mainly, **['comp.graphics','rec.sport.baseball','sci.med','soc.religion.christian'].** There are 2374 samples in training set and 1580 in testing set and each sample is labeled.

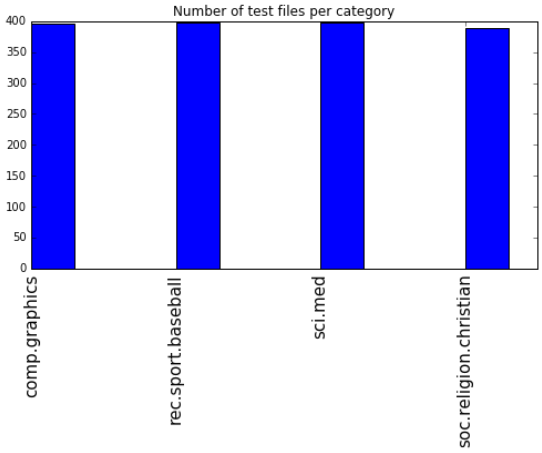
Similar analysis can be done for other classes. **The Classification is carried over both 4 classes and 20 classes.**

Data Exploration and Visualization

I checked some stats for corpus through barplots, histograms:

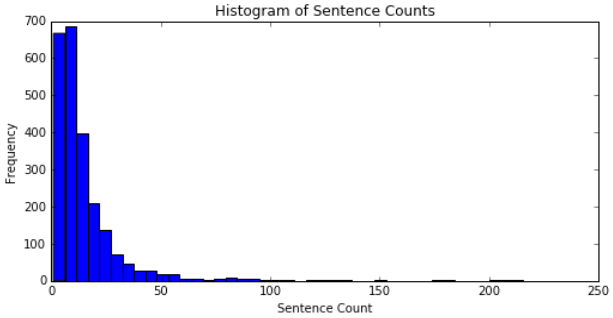
1. Below is distribution of train and test files per class. It can be seen that each class has almost same number of files and thus data seems to be balance.

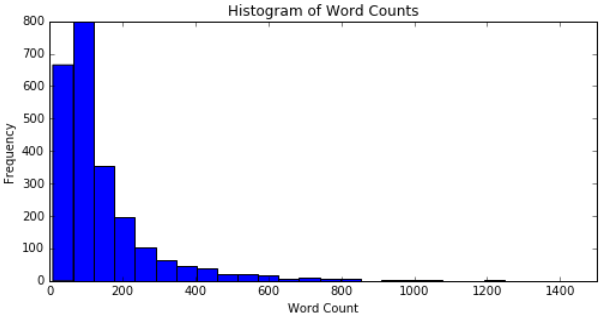




2. Histogram of Sentence and Word counts.

   Below is histogram of sentence and word counts of training data.



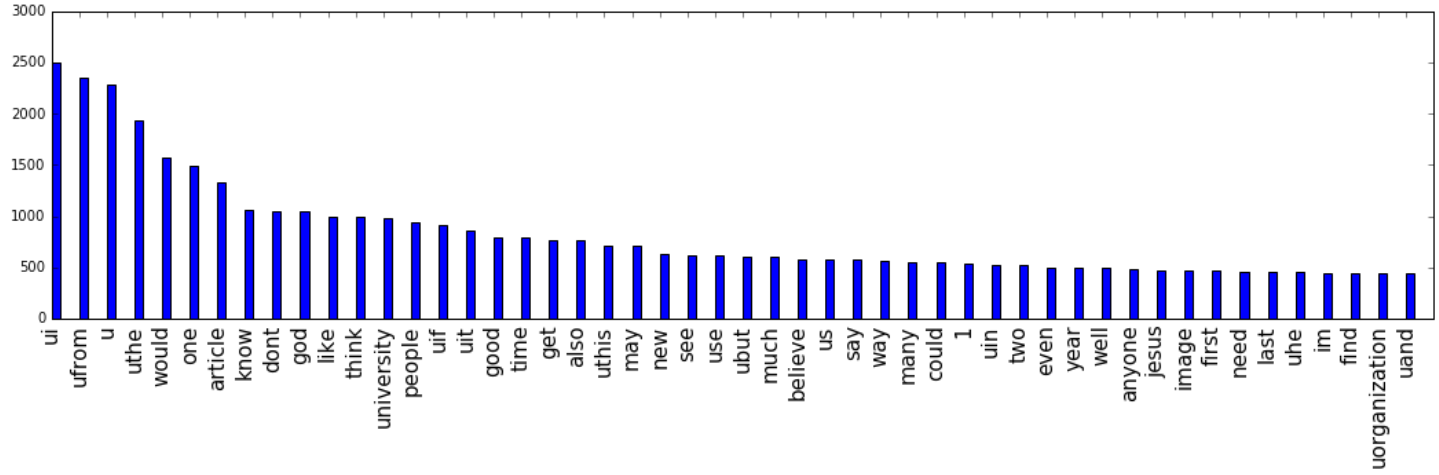


   I have used nltk library for sentence and word tokenization. I have removed the punctuation and removed the stop words.

It can be seen that most files have less than 40 sentences and median is close to 20. Also most files have less than 400 words with max word count of 5613 and minimum word count of 9. (Please note that this is after removing the punctuation and stop words)

3. Histogram of top 50 frequent words

   This is plotted across full training data.



The total number of unique words after removing stop words are 73534. This is a huge feature set as compared to number of sample in training data. I am going to reduce this number by setting parameters while doing vectorization of text.

Classification Algorithms and Methodology

Data Preprocessing

First of all, I extract features from the news data. These features can be words, sentences or phrases or combination of them all. I have considered using single words as features allowing for model complexity. I have **used Tf-Idf algorithm** to extract features rather than simple word frequencies. The tf-idf value increases proportionally to the number of times a word appears in the document, but is offset by the frequency of the word in the corpus, which helps to adjust for the fact that some words appear more frequently in general.

Classification - 4 class

This problem has over 2000 training samples with four categories, a multi-class classification problem with high dimension.

After removing the punctuation, turned letter into lowercase and removing the stopwords, there were a total number of 73534 words. I further reduced this feature set by setting the parameters (min_df = 5) in **Tfidfvectorizer.**

The final feature vector was (2374, 8521) –

```
<2374x8521 sparse matrix of type '<type 'numpy.float64'>'
       with 183053 stored elements in Compressed Sparse Row format>
```

Benchmark Classifier – Naïve Bayes

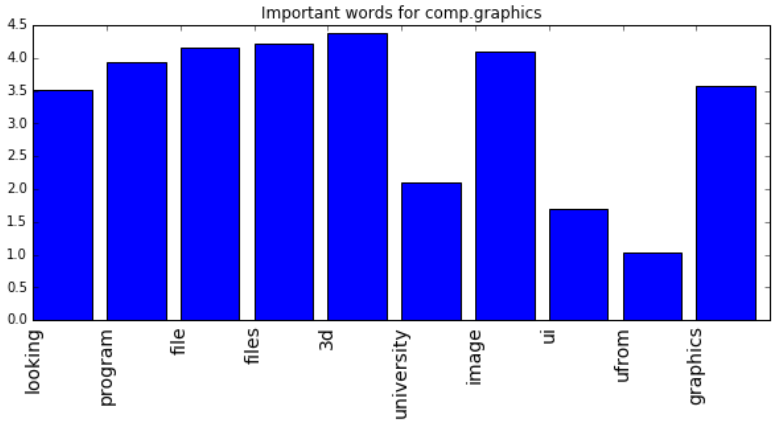
The accuracy of NB classifier was 94.36%

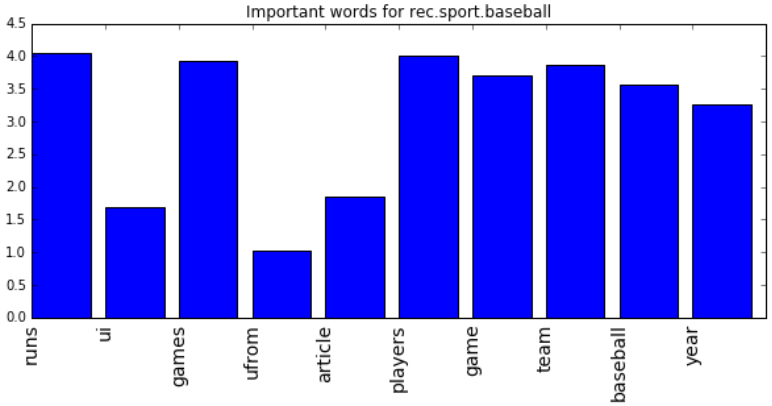
```
Training data Accuracy: 98.7784330244 %
Test data Accuracy: 94.3670886076 %
```

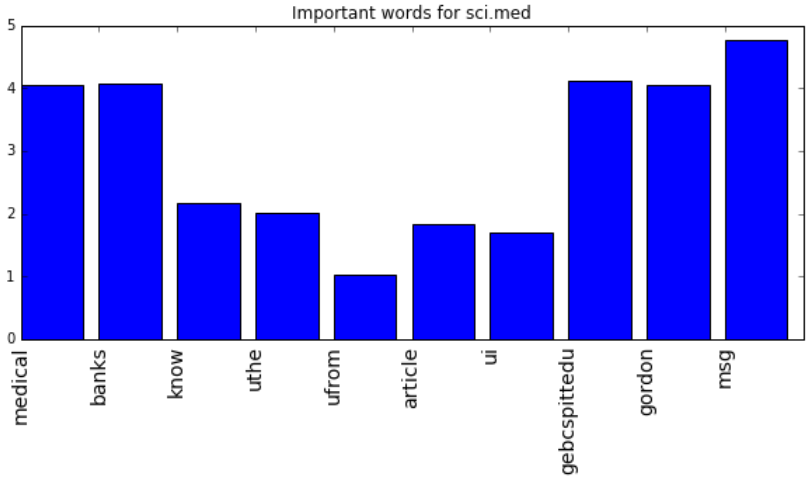
LinearSVC

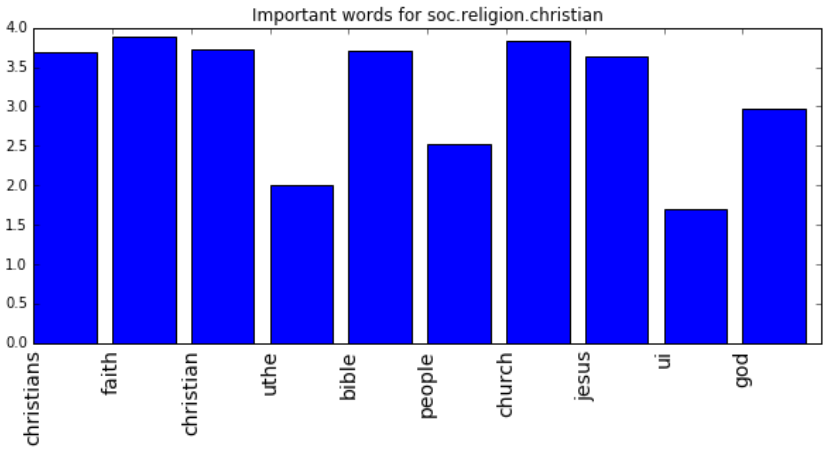
```
Training data Accuracy: 100.0 %
Test data Accuracy: 95.0 %
```

I extracted word weights for each class from Mutinomial NB model, and listed the largest Weights(from TFidfvectorizer.idf_)along with corresponding words as important features for each class. Below are the plots:









It looks like NB model performed well in classifying the 4 classes.

The barplots show that certain important words were indeed related to specific categories.For example, in the first figure, ‘3d','image’, graphics' are very important for 'comp.graphics' topics like and ‘faith, Christian, ‘god’, ‘bible’ are important topics for ‘soc.religion.christian’. These also proved that the Multinomial model works well on this dataset.

Classification - 20 classes

This problem has over 10000 training samples with four categories, a multi-class classification problem with high dimension.

After removing the punctuation, turned letter into lowercase and removing the stopwords, there were a total number of 259744 words. I further reduced this feature set by setting the parameters (max_df = .5) in Tfidfvectorizer.

I have used tfidf vectorizer with following parameters for my 20 class problem.
vectorizer = TfidfVectorizer(max_df=0.5,sublinear_tf=True,stop_words='english')

The feature vector after vectorization was

```
<11314x259541 sparse matrix of type '<type 'numpy.float64'>'
      with 1305864 stored elements in Compressed Sparse Row format>
```

I have used Python scikit-learn package to do the feature extraction, as well as Naive Bayes Learning, Decision Tree learning and SVM. I imported Multinomial Naive Bayes classifer from sklearn.naive_bayes, trained them on the extracted feature matrix and predicted the categories on testing feature matrix.

Classifiers with default parameters - Accuracy results:

Naïve Bayes:

```
Training data Accuracy: 96.9948736079 %
Test data Accuracy: 80.8815719596 %
```

Random Forest:

```
Training data Accuracy: 99.8055506452 %
Test data Accuracy: 61.1524163569 %
```

SVM

```
Training data Accuracy: 99.9823227859 %
Test data Accuracy: 84.9707912905 %
```

Refinement - Parameter tuning using Gridsearch

I have used gridsearch to find the best parameters for these estimators.

The parameters tuned are as below:
Feature extraction - Count vectorizer with one grams and bigrams and TFidf transformer
Parameters tuned for Naïve bayes – alpha
Decision trees – No. of estimators and max depth
SVM – C and gamma (tol)

Naïve Bayes

```
Best parameters:{'vect__ngram_range': (1, 1), 'tfidf__use_idf': T
rue, 'clf__alpha': 0.05}

Training data Accuracy: 99.3282658653 %
Test data Accuracy: 83.6431226766 %%
```

LinearSVC

```
Best parameters: {'vect__ngram_range': (1, 2), 'clf__tol': 0.01,
'clf__C': 1, 'tfidf__use_idf': True}

Training data Accuracy: 99.9027753226 %
Test data Accuracy: 83.0855018587 %
```

Random Forest

```
Best parameters: {'vect__ngram_range': (1, 1), 'clf__max_depth':
50, 'tfidf__use_idf': True, 'clf__n_estimators': 30}

Training data Accuracy: 91.4972600318 %
Test data Accuracy: 67.5252257037 %
```

From the boave results it looks like:

- NB with alpha = 0.05 gives best performance.
- Linear SVX with C =1 and tol = .01 gives best performance.
- Random forest does not seem to work well on this data set and is overfitting with 99% accuracy on training data and only 67% accuracy on test data.

Ensemble of Classifiers:

To further improve the accuracy I did ensemble of three classifiers. I used the best parameters found using gridsearch method.

MultinomialNB(alpha = 0.05)
RandomForestClassifier(n_estimators = 30,max_depth=50)
SGDClassifier(alpha=0.0001,loss='modified_huber',n_iter=50)

I created my own definition of ensemble classifier with fit, predict and predict_proba methods. I used cross validation to compute accuracy scores.

Training data score:

```
Accuracy: 0.88 (+/- 0.02) [Naive Bayes]
Accuracy: 0.71 (+/- 0.03) [Random Forest]
Accuracy: 0.90 (+/- 0.02) [SVM]
Accuracy: 0.90 (+/- 0.02) [Ensemble]
```

Test data scores:

```
Accuracy: %0.2f [%s] 0.835767392459 Naive Bayes
Accuracy: %0.2f [%s] 0.656664896442 Random Forest
Accuracy: %0.2f [%s] 0.849442379182 SVM
Accuracy: %0.2f [%s] 0.857142857143 Ensemble
```

Ensemble of Naïve Bayes, Random forest and SVM improved test accuracy little bit to 85.71%

Conclusions

- TFIDF with SVM works best for both 4 class and 20 class problem
- Naïve Bayes model also performed comparable to SVM.
- On the 4 class problem the model is less complex and hence higher accuracy is seen than the 20 class problem.
- Random forest did not work well on this dataset.
- Ensemble of Naïve Bayes, Random forest and SVM improved test accuracy little bit to 85.71%

I believe for this particular dataset the accuracy results are bit lower because in the bydate version, data is sorted. The test set is all from a later time period than the training set. Topics under discussion shift over time, and so there's enough extra similarity between documents close in time versus temporal movement in what gets posted over time. Model is not able to generalize well because the content of a particular newsgroup drifts over time.