20 Newsgroup Document classification Report

Definition

Project Overview

Document classification or document categorization is a problem in library science, information science as well as computer science. The task is to assgin a document to one or more classes. This may be done "manually" (or "intellectually") or algorithmically, and the documents to be classified may be texts, images, music, etc. Each kind of document possesses its special classification problems. Documents may be classified according to their subjects or according to other attributes (such as document type, author, printing year etc.)[1].

This project will focus on algorithmical methods, exactly machine learning algorithms which are widely used in information science and computer science. There are many algorithms to do classification, such as KNN, Naive Bayes, Decision Tree and etc, which have their advantages and disadvantages, it depends on real situation to use which one. Also there are many public labeled text datasets online for classification, for example, Python's NLTK package can contain a lot of free texts. Here, I will apply classification algorithms on 20_newsgroup dataset from CMU Text Learning Group Data Archives, which has a collection of 20,000 messages, from 20 different netnews newsgroups. The news will be classified according to their contents.

Problem Statement

The classification of 20_newsgroup dataset is a supervised problem, each piece of news belongs to one category, the goal of this project is to extract proper features and train a model to assign each piece of news to the correct category. I will explore the dataset in the beginning , then extract features like keywords and build vectors of features from the corpus, based on those vectors I will use several classification methods to do classification, compare the efficiency of these classifiers on the testing data and choose one as final model. Finally I will tune the parameters and validate the performance of the classifiers on different datasets to check the robustness of the selected model.

Metrics

As it is a multi-class classification problem with thousands of samples, I will use two metrics here:

- accuracy: to measure how well the classification algorithms perform, namely rates of samples which are correctly assigned to corresponding categories. Ideal accuracy is 100%.

- timing: how long it takes the algorithm to do classification, a scalable algorithm is expected to work fast because corpus usually includes thousands or more text files.

Analysis

Data Exploration

The dataset can be acquired online directly with built-in Python scikit-llearn tools. There are 20 categories of news in all, here I just choose 4 categories to do analysis, analogously, the other categories can be also handled in this way.

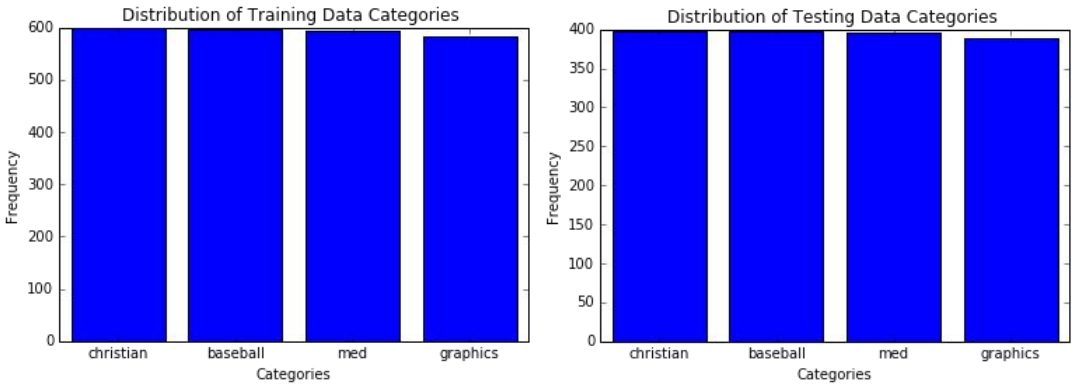
The dataset has been splitted into training part and testing part, each part consists of news of different categories. There are 2374 samples in training set, and 1580 in testing set, and each sample has been labeled by researchers. They belong to four categories involved with computer science, baseball, medical, religion. Let's look at the content of one text.

From: lyford@dagny.webo.dg.com (Lyford Beverage)
Subject: Re: Notes on Jays vs. Indians Series
Distribution: na
Organization: Data General Corporation, Research Triangle Park, NC
Lines: 22
In article <1993Apr13.202037.9485@cs.cornell.edu>, tedward@cs.cornell.edu (Edward [Ted] Fischer) writes:
|> In article <rudyC5Fr3q.1CL@netcom.com> rudy@netcom.com (Rudy Wade) writes:
|> >In article <C5FMxD.2pM@cs.dal.ca> niguma@ug.cs.dal.ca (Gord Niguma) writes:
|> >>reference to history because he certainly didn't have the best season for
|> >>second basemen in history. He probably didn't even have as good a season as
|> >>Alomar last year.
|> >
|> >What? Do you have some measure (like popularity in Toronto doesn't count)
|> >that you are basing this statement on?
|>
|> Uh, yes. Baerga has a lot of flash, but Alomar was the better hitter
|> last year.
|>
|> BATTERS BA SLG OBP G AB R H TB 2B 3B HR RBI BB SO SB CS E
|> BAERGA,C .312 .455 .354 161 657 92 205 299 32 1 20 105 35 76 10 2 19
|> ALOMAR,R .310 .427 .405 152 571 105 177 244 27 8 8 76 87 52 49 9 5
|>
This is fascinating. You say that Alomar was the better hitter last year, and immediately follow that up with numbers showing that Baerga had a better year. The only category that I see which shows an advantage for Alomar is OBP.

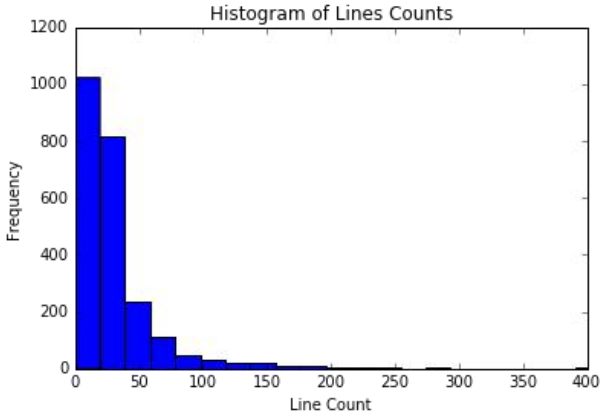
It looks like an email, there are a subject, an organization, an email address and a main body, the subject and content can contain some useful words in terms of classification, for example, this article mentioned 'basemen', 'hitter', terms often refer to baseball. Aside from these, there are many signs such as punctuations, digits in the content, which may be redundant for classification.

Exploratory Visualization

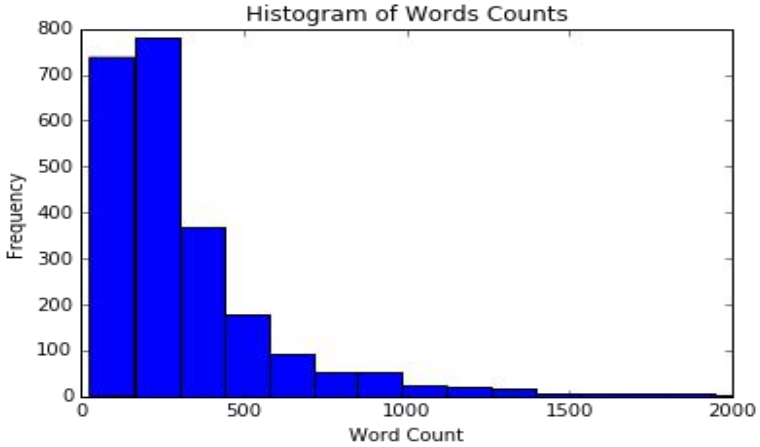
I have checked some statistical characteristics of the corpus through barplots, histograms and etc.



Each category has almost the same articles, both the training and testing data are balanced. We need not reshuffle and split them.



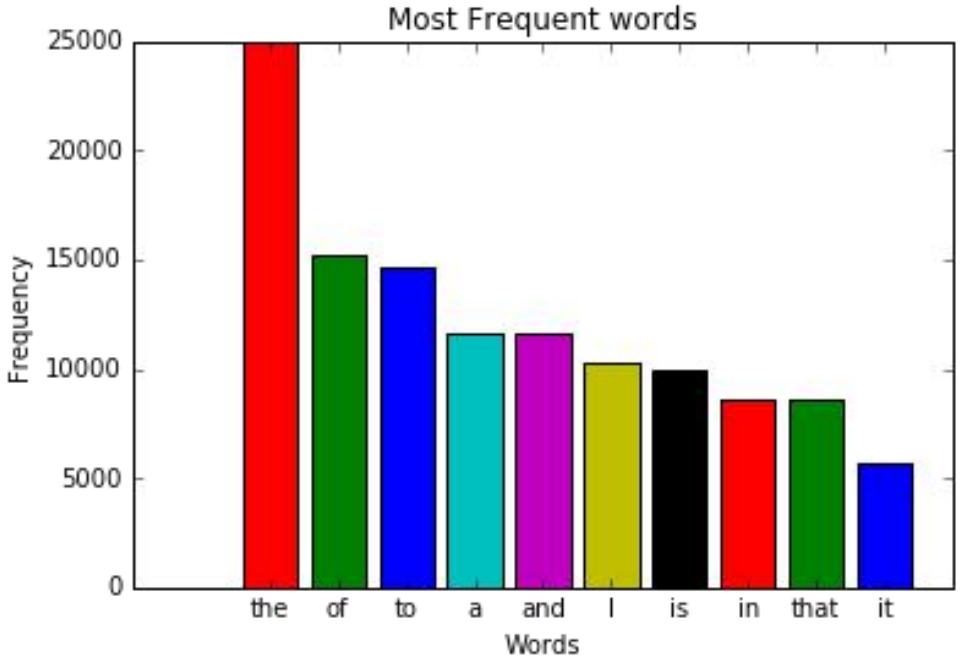
Most articles have less than 50 lines, the median is 22, so these news consist of quite short articles indeed. Note: there are some articles with 0 line, perhaps they do not contain much information, but only one piece of news has 0 line, the influence is minor, we can keep them in the corpus .



I have tokenized those articles, namely split the texts into words using textblob package. Most articles have less than 500 words, max word count is 11002, min word count is 26. There are no obvious outliers. Note: the header of each article accounts for over 20 words, so the contents should contain more than 20 words, otherwise the articles are not useful. Actually there are only 3 articles with less than 30 words. It is not neccessary to remove them.

The total number of unique words can be 671564, a very huge number if we treat each one as a feature, and it is not easy to handle such a large feature matrix on a single computer, the training process may go far beyond the computation capability of a desktop. Therefore, It is impossible to represent each article with all the words as features in the corpus. Besides, if we use all the words, maybe the model will not be generalized, it will not perform well on new corpus which has new words. So we need extract only a proper proportion of words as keywords.

Anyway, in this project, it is likely that the features(selected words) outnumber samples, so we need pay attention.



I have also found out the most frequent words in the given corpus. It is clear that the most frequent words are stop words such as ‘the’, ‘of’, commonly seen almost in every article, but without much meaning in term of classification, they can be removed to reduce the dimension of our problem.

Algorithms and Techniques

First of all, I need extract features from the news, these features can be words, sentences or phrases or combination of them all. But here, I only consider using words as features o represent an article allowing for the computation scale. If a special word such as 'cancer' apear

often in an article, it is likely to be classified as science related, but we can not simply use frequencies of words because most frequent words do not provide much useful information as the figure show above, i.e, you can find 'the' very often in each piece of news, but it gives nothing useful about the category. In fact, some specific words such as 'basemen', 'hitter', they can provide very useful information, and they do not appear often. Therefore,I'll use Tf-Idf algorithm to extract features. It suffices to divide the number of occurrences of each word in a document by the total number of words in the document: these new features are called tf, short for TermFrequencies. Another refinement on top of tf is to down scale weights for words that occur in many documents in the corpus, and are therefore less informative than those that occur only in a smaller portion of the corpus.

In this problem, it has over 2000 training samples with four categories, a multi-class classification problem with medium scale samples, and a large number of features. KNN is not suitable here because of computing efficiency in high dimension. I will try *Multinomial Naive Bayes*[2], *Decision Tree*[3] and *Deep learning*[4] methods, they can work even on large datasets, the characteristics are as following:

- Multinomial Naive Bayes
  Pros: only requires a small amount of training data for classification, highly scalable, used very often in text retrieval.
  Cons: Oversimple for certain data, it needs assumption that the value of a particular feature is independent of the value of any other feature.
- Decision Tree
  Pros: easy to interpret the results, scalable
  Cons: the accuracy may be not excellent, prone to overfitting.
- Deep neural network
  Pros: can produce state-of-the-art results on various tasks
  Cons: computationally expensive

I will use *scikit-learn* package to do the feature extraction, Naive Bayes learning and Decision Tree learning. As for Deep neural network, I will use *tensorflow*[5]. The corpus of news do not provide default parameters for *NB* and *Decision Tree* models, the parameters should be set in advance, and need tuned if necessary. But for *deep learning*, the number of features can determine the size of weights of the first layer, as for the hidden lay, the size should also be set in advance.

Benchmark

Here we consider two factors: bias and variance.

- Bias: the ideal model should assign each testing article to the right category, namely, the accuracy 100%. The higher accuracy is, the better the algorithm is likely to be. Here I set the accuracy threshold as 90%.

- Variance: the model performance should not vary much on different datasets, we should try to avoid overfitting. Also, with different parameters, the model should be robust. Here I limit the accuracy difference between training and testing data is below 8%.

Furthermore, the models should not take too much time if we have limited hardware resources.

Methodology

Data Preprocessing

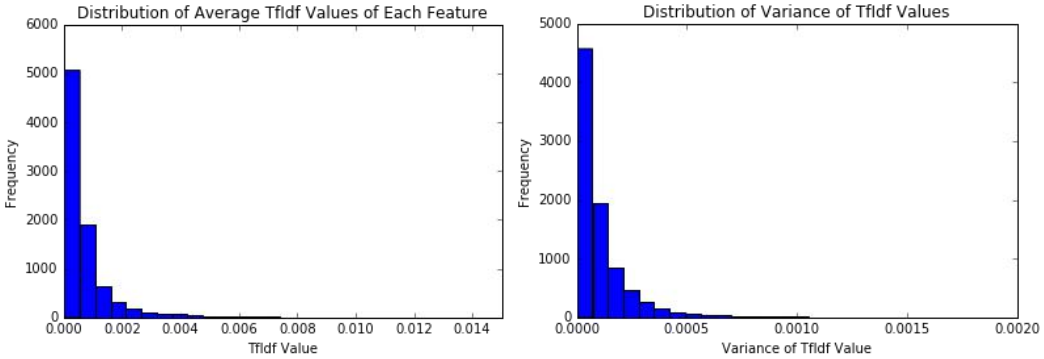
As I analyzed above, I did not consider the punctuation, digits and stopwords in the contents of news. I wrote a function and removed them before feature extraction, after that there were 41948 words left in all, but many words were repetitions of letters like 'AAAA', which seemed like typos, which could be removed by setting a minimum of frequency.

I ran the feature extracting function again, this time there were only 8554 words, which were more meaningful rather than typos. After feature extraction, the vector looked like:

```
array([[ 0.        ,  0.        ,  0.09797326,  0.        ,  0.        ,
         0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
         0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
         0.        ,  0.38527571,  0.        ,  0.        ,  0.        ,
         0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
         0.        ,  0.        ,  0.        ,  0.        ,  0.        ]])
```

Actually most values are zeros in the matrix, so *sklearn* package transformed them into sparse matrix to save space. In later analysis, I need transform them into normal matrix again.

There were still over 8000 words, and each article could be represent by the TfIdf values of selected words. My classification work would be based on these sparse matrix. I did not transform uppercase to lowercase here, because in certain cases, some abbreviations in uppercase had different meanings from lowercase ones.



Both the means and variances of each column(feature) were in similiar scales, there were no very large or extremely small figures, so no need to normalize them.

Implementation

I used NB, Decision Tree and deep neural network methods to train the data, the metrics were accuracy and timing. The parameters of NB and Decision Tree were default, the classifiers were derived from scikit-learn package. First 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. Then, I imported tree.DecisionTreeClassifier() from *sklearn,* trained them on the extracted feature matrix and predicted the categories on testing feature matrix.

For the deep neural network, I designed a deep neural network classifier based on *tensorflow.* There was only 1 hidden layer of 512 units in the neural network, and the output of the hidden layer was processed by *Rectifier Linear Units*[5][7]. And during the process I used soft max cross entropy as the loss function , in order to speed up learning, I applied stochastic gradient descending method to the transformed dataset. Ahead of learning, I wrote a function to transform format of the feature matrix, and . To get a good result, I trained neural network for 1001 loops.

All the computing time were recorded. The result turned out to be in this way: the more simple, the better it performed. The accuracy of *NB* classifier was 0.946, and the computing time was 0.006s. The accuracy of *Decision Tree* classifier was 0.737 and the computing time was 0.578s. The accuracy of *neural network* was just 0.72 and the computing time reached 44s. It seemed *NB model* worked pretty well on the dataset in terms of both accuracy and timing, whereas deep neural network did not perform well in this case.

The results of three classifier indicated that the assumption of feature independence was quite strong, namely those words were independent from each other, so the articles could be classified on the basis of probabilities. And it was clear from the result that overfitting happened when using deep neural network, the training accuracy has been 100%, after 500 loops, yet the testing accuracy was only 0.72, perhaps there should be more training samples allowing for so many features.

Next, I would only tune the parameters of NB and Decision Tree models.

Refinement

I used grid search method to tune parameters of Naive Bayes, exactly the values of alpha . Here, I consider accuracy only because the timing was very small indeed. I set the *alpha*

ranged from 0.01 to 1, and when alpha = 0.01, the model had a best accuracy of 0.957 on testing data, an improvement of default one whose accuracy was 0.946. It seemed that smaller alpha could lead to a better accuracy.

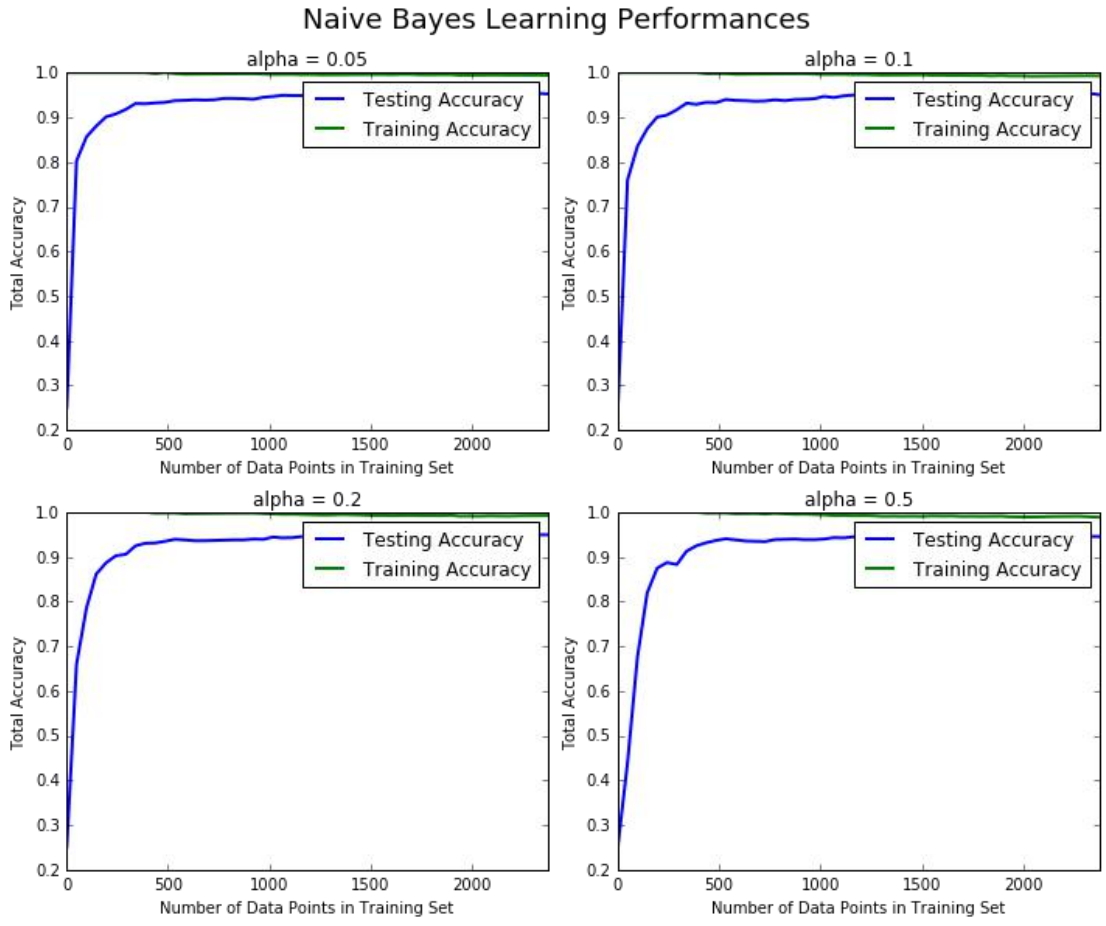
In addition, I also tuned *max_depth* parameter of Decision Tree classifier. I set the max_depth as (3, 4 ,6 ,8, 10). Overfitting happened if I tried to improve training accuracy, the grid search method chose *max*_depth=10, while the accuracy on the testing data was less than 0.6. Therefore, I would only discuss NB model later.

Results

Model Evaluation and Validation

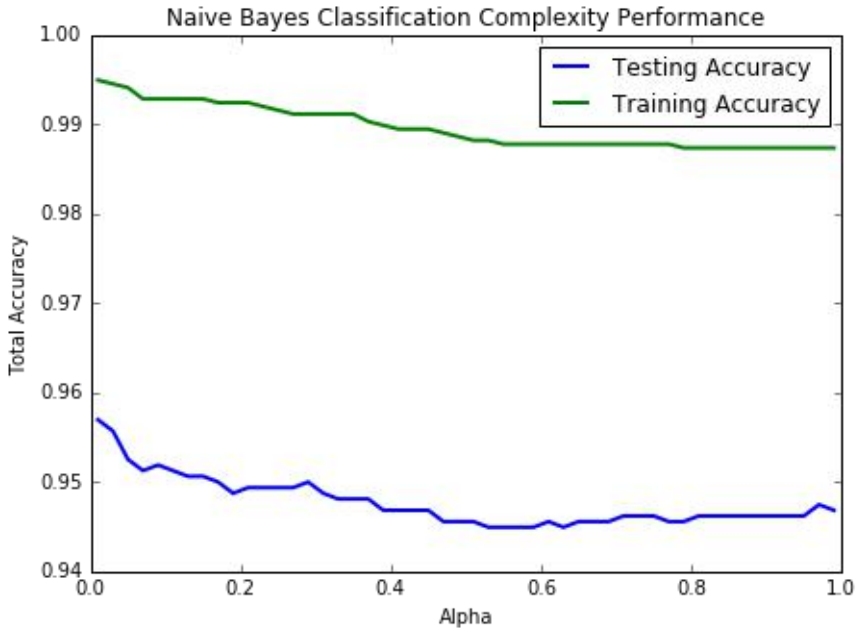
In this part, I took a look at several models' learning and testing accuracies on various subsets of training data, to check bias and variances of the models. In addition, I investigated NB classifier with an increasing alpha parameter on the full training set to observe how model complexity affects learning and testing accuracies.

Based on my previous study, I wrote functions to calculate the performance of several models with varying sizes of training data. The learning and testing accuracies for each model were then plotted.



Those figures indicate that NB classifier works well on those datasets in terms of variance and bias. The accuracy grows as the number of samples increases initally, When the number of samples are over 500, the accuracies become quite stable. The training accuracy reaches almost 100%, and testing accuracy can reach as high as 95%. No overfitting here. The classifier is robust in terms of sample scales.

I wrote a function to calculate the performance of the model as model complexity increased as well. The learning and testing accuracies were then plotted.



As the alpha increases, the accuracies of training dataset and testing dataset go down slowly. Then after 0.4, the accuracies seem to be stable. The testing accuracy is around 95%, and when alpha is close to zero, it has large accuracy. The differences between training dataset and testing dataset are small, no overfitting here.

Above all, NB classifier is quite robust to different training datasets and alpha. I will set alpha= 0.01 as the parameter of our final model.

Justification

As analyzed above, Multinomial NB model performed well on the classification tasks in terms of both bias and variance. The accuracy of classification could reach 95%, and the variancewas also small, that meaned the model generalizes well.

Furthermore, the NB model assumes independence between features, I'd like to check whether those features are independent in terms of linear relationship. I have computed the correlation coefficients between each feature, and the maximal value was only 0.00012, very

small, it was almost true that those words are linearly independent in terms of TfIdf values.

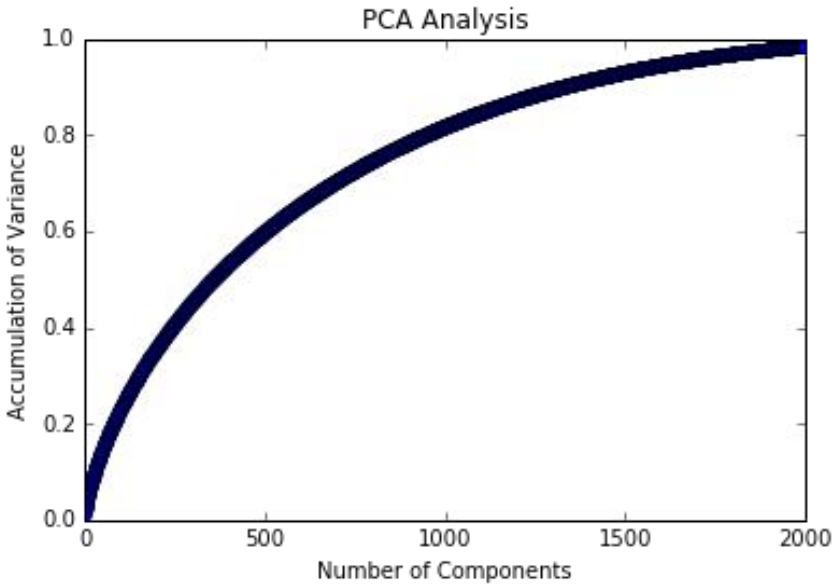
Remember, in our original datasets, many datasets of other categories remain left, I could use them to test my model. This time I would choose datasets of another four categories(‘rec.motorcycles', 'talk.politics.misc', 'comp.windows.x', 'sci.space') to validate this model.

The model still performed quite well. The accuracy was above 0.96 and the timing was still very small. So Multinomial Naive Bayes model could classify these articles precisely and quickly.

Conclusion

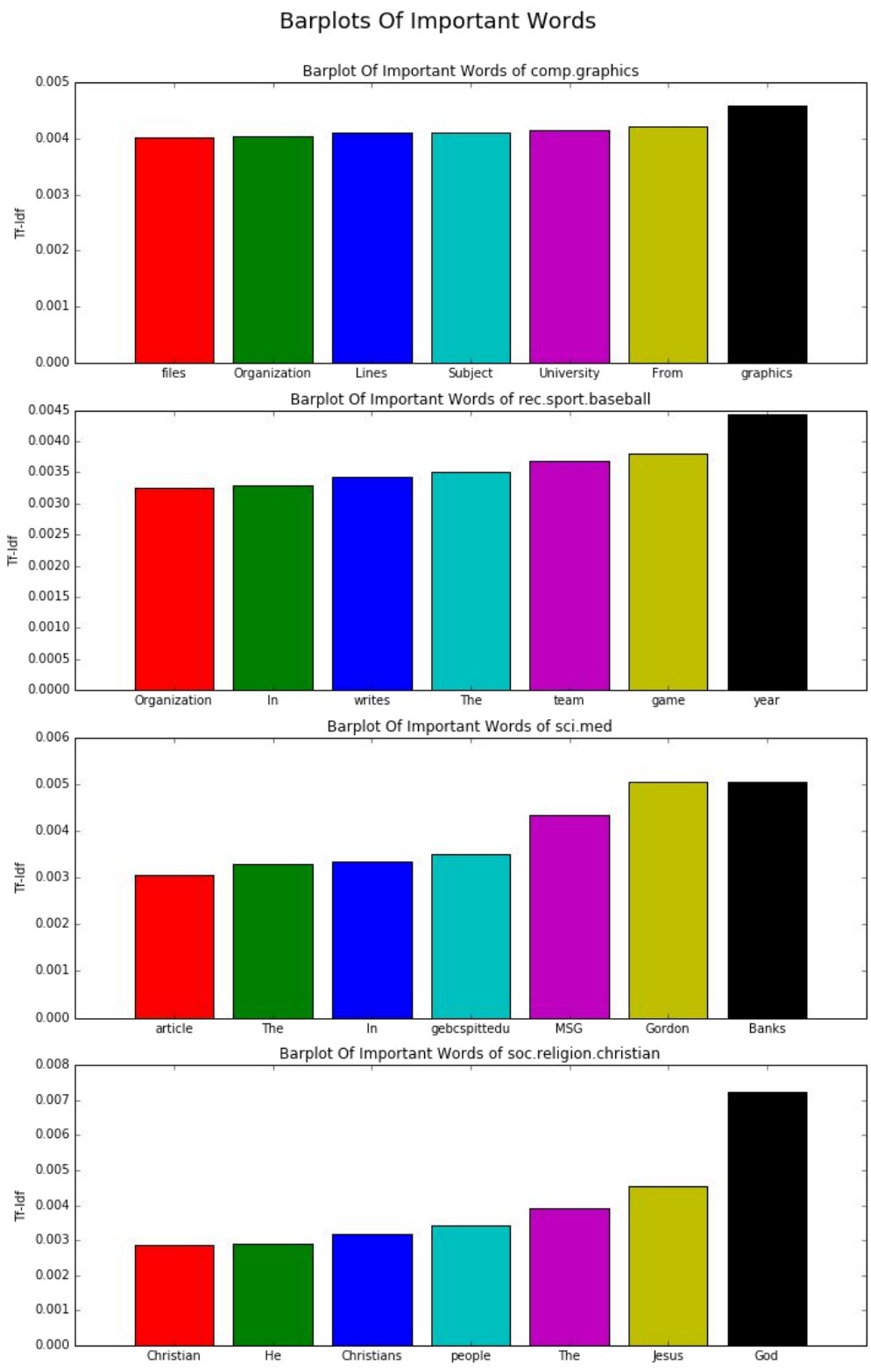
Free-Form Visualization

As we have thousands of features, I wondered whether it was possible to reduce the dimension of the features, so I used PCA tool(Principal Component Analysis) from *sklearn.decomposition* package on the feature matrix.



From the figure above, it was clear that only 2000 components of 8000 could contain almost all the variance of original features, and it was possible to use PCA to reduce the dimension of features if we could accept certain loss of accuracy.

In addition, I explored how the features influence the classification results, to check certain words are related to articles of certan categories. I will try to find out some of important words for each category.



It showed that these listed important words were related to certain categories, which wascoherent to the classification. For example, in the first figure, 'graphics' is very important for 'comp.graphics', and in the last figure, 'God', 'Christians', 'Jesus' point to religion. But there are also stopwords such as 'The', 'From', perhaps the default English stopwords in sklearn package are not sufficient, if neccessary we can enlarge the stopwords vocabulary.

Reflection

This project was to solve a multi-class document classification problem. I downloaded 20newsgroups datasets, analysed the characteristcs of the datasets, extracted features using TfIdf from the corpus. The features outnumbers training samples, so I applied Multinomial Naive Bayes, Decision Tree and Deep learning classifier to the datasets, and found that simple classifiers worked better, NB performed best in terms of accuracy and timing on the testing datasets. Then I tuned parameters of NB classifier to improve the performance with grid search method. Afterwards, I analysed robustness of NB by exploring models' learning and testing accuracies on various subsets of training data, and comparing accuracies as changing values of alpha. NB classifier has balanced bias and variance, it generalized well on different datasets if the sample number is over 500, and the variance of classifier is not sensitive to parameter changes. In addition, I validated the classifer on other catergories of datasets, it still worked well. Finally, I explored the weights of important words of each category, which were coherent in terms of real world meanings.

Improvement

In this project, I did not transform all the words into upper or lowercase, neither did I consider different forms of the same words, for example, 'Christian', 'Christians', 'christian' are the same in meaning, if I combine them into one word, I can reduce the redundance of features and improve efficiency of the model. Apart from these, I can also enlarge the stopwords vocabulary to reduce redundance. Furthermore, if I take phrases such bigrams or trigrams into consideration, i.e., 'big data', 'machine learning', it will be likely to increase the classification accuracy, but require more computing capabilities and memories.

Reference

[1]Document Classification: en.wikipedia.org/wiki/Document_classification
[2]Naive Bayes Classifier: en.wikipedia.org/wiki/Naive_Bayes_classifier
[3]Decision Tree: en.wikipedia.org/wiki/Decision_tree
[4]Deep Learning: en.wikipedia.org/wiki/Deep_learning
[5]Deep Learning Class: cn.udacity.com/course/deep-learning--ud730
[6]scikit-learn userguide: scikit-learndevelopers, Release0.16.1
[7]Relu function: en.wikipedia.org/wiki/Rectifier_(neural_networks)