Document Classification using different ML Algorithms

Naitik Dodia – 201501177

Dhirubhai Ambani Institute of Information & Communication Technology,

Gandhinagar, Gujarat 382007, India

IT 563, Data Mining

Introduction:

Document classification is generally defined as content-based assignment of one or more predefined categories to documents. The increasing volume of the documents and their important role in the product life cycle requires an automatic classification system in order to categorize them in the proper pre-defined classes and to facilitate the information retrieval. Some documents contain structural components like schemas and tables for ex. Technical documents. Document classification can be used for document filtering and routing to topic-specific processing mechanisms such as information extraction and machine translation.

This project will focus on Machine Learning algorithms like Naïve Bayes Classifier (especially Multinomial Naïve Bayes) and Support Vector Machines for automatic Document classification.

The Dataset used for this project is the classic 20newsgroup dataset which contains 20000 news articles spread across having 20 different categories. The two main phases of this project are feature extraction and learning. As we have results for the training data, this problem is classified as supervised classification problem.

The metric used here is accuracy. Based on accuracy we will check that which algorithm is better. Time to run the algorithm also matters but these algorithms take similar amount of time so we will not consider time as useful metric.

Model:

The library used here is scikit-learn in Python. Sklearn provides the 20newsgroup dataset in its built-in datasets and hence it is downloaded directly from the code.

1. Analysis of dataset:

Takin into consideration the first four categories, I have done some exploratory visualization and the results are as follows.

Here Fig 2.1.1 shows us the distribution of documents in the first four categories. We can see that the document frequency is almost same. This means that we don’t need to mix, reshuffle and split both the training and test data.

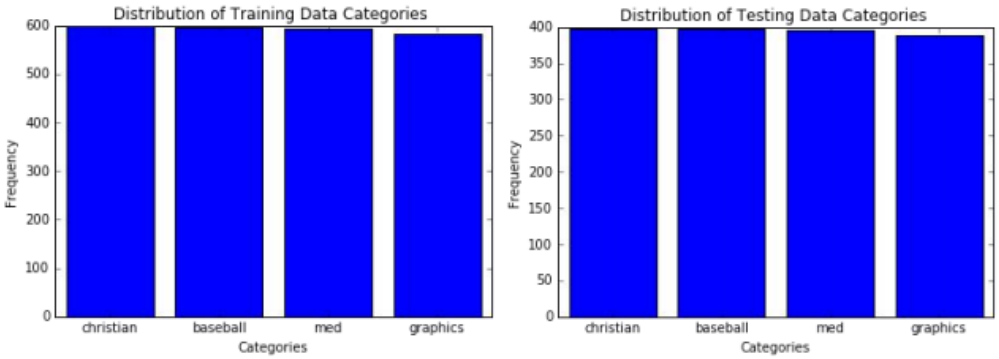


Fig 2.1.1: The distribution of documents among the first four categories in the training dataset(left) and test dataset(right)

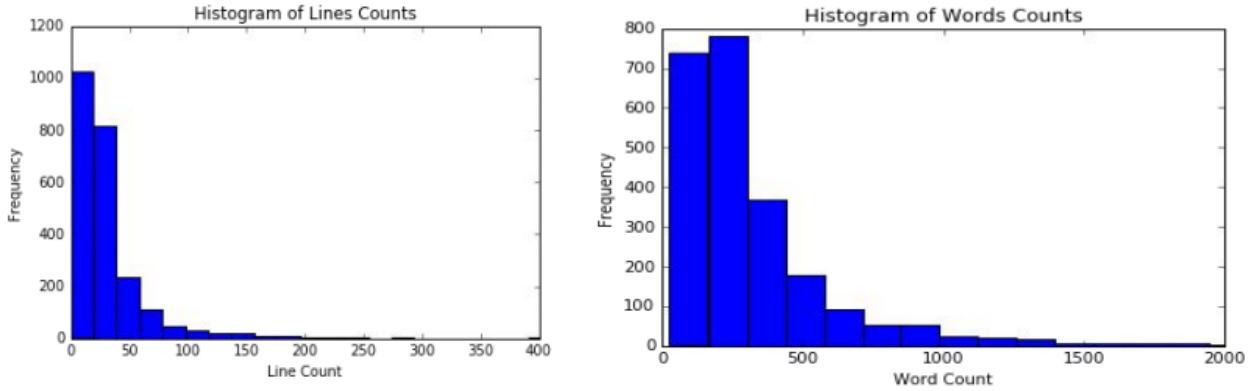


Fig 2.1.2: The distribution of line count(left) and word count(right) in the sample taken for analysis

Fig 2.1.2 shows us that the line count and word count are normally distributed about zero. So we don't have to normalize the data. And the number of outliers is also very less. So removing the outliers is also not necessary.

2. Feature Extraction:

I have considered the count of each word in each document as the feature set of each document. So out feature set consists of all the words with their frequency in their corresponding documents. This counting is done using CountVectorizer that is already present is scikit-learn.

As this count is very high and these two counts for two different documents cannot be compared directly, I have used TFIDF format for the feature set. TFIDF stands for Term Frequency Inverse Document Frequency. In this format the frequency of each words in each document is taken and it is divided by the frequency of the class of the document in which it is present. This makes it possible to compare the features of each document with other irrespective of the class of the document.

In the first case I have taken 1-word group as one feature and run the algorithms on it. But we ca take 2,3,4,5 - word count as feature and then apply the classification algorithms. This type of featuring technique is called n-gram featuring. In my case the n in n-gram corresponds to 1,2,3,4,5. Basically n-gram featuring groups the adjacent words which are present in the range and then CountVectorizer and TFIFD is applied.

To know which of the n-gram will best suit for document classification, I have used Grid Search method. “Grid search” method applies all the different parameters given to it and gives us the best parameters that we can use for our classification. It does this on the basis of statistical significance and decreasing the overfitting. So to reduce overfitting and to choose the values of parameters “Grid Search” is used.

3. Algorithms and Techniques:

In the following description of techniques, the feature set is considered as 1gram feature set and the learning rate used is 0.001

1. **Multinomial Naïve Bayes Classification**

   In this algorithm it is assumed that there is no relation between individual features of the dataset. The conditional probability for each class given the set of features value of the document is calculated on the basis of the following formula.

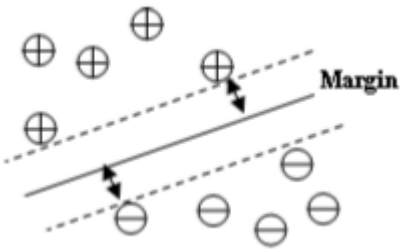
$$P(C = c|z_1, z_2, \cdots, z_n) = \frac{P(z_1, z_2, \cdots, z_n|C = c)P(C = c)}{P(z_1, z_2, \cdots, z_n)}$$

$$P(z_1, z_2, \ldots z_n|c_j) = \prod_{i=1}^{n} P(z_i|c_j)$$

   And the class having maximum probability is predicted. Here the second formula is valid only under the assumption of this algorithm. But in reality the words are always internally related by some factor. Though not considering this fact the Naïve Bayes classifier works well and the accuracy obtained is **77.39%** which is not bad.

2. **Multinomial Support Vector Machines:**

   Support vector machines can be considered as extended part of logistic regression. A simple support vector machine is a binary linear classifier which separates the positive and negative examples in a training set. The method looks for the hyperplane that separates positive examples from negative examples, ensuring that the margin between the nearest positives and negatives is maximal. The effectiveness of SVM is superior to other methods of text classification. SVM makes a model representing the training examples as the points in a dimensional space separated by the hyperplane, and it uses this model to predict a new example belongs to which side of this hyperplane. The examples used in searching the hyperplane are no longer used and only these support vectors are used to classify new case. This makes a very fast method.



1 Maximum margin Hyperplane

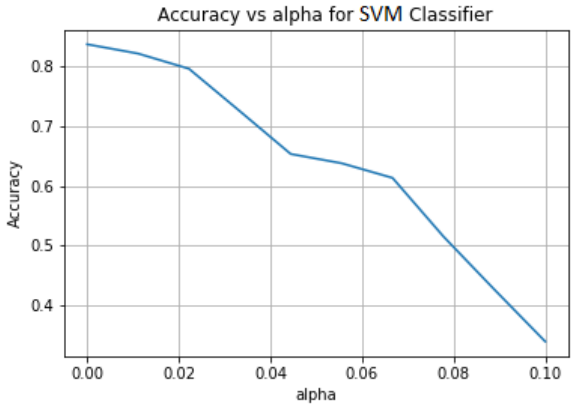
   Two types of SVMs are available: 1) linear – the margin created corresponds to linear vector and 2) nonlinear – the margin created corresponds to a non-linear vector. In multinomial linear SVM, many margins are created and each section separated by the combination of these margins correspond to a specific class of the document.

   Using the linear SVM and the parameters mentioned above, the accuracy obtained is **82.38%.** Comparing with Naïve Bayes Classifier, SVM gives more accuracy in less time.

3. Further Analysis (n-gram and alpha):

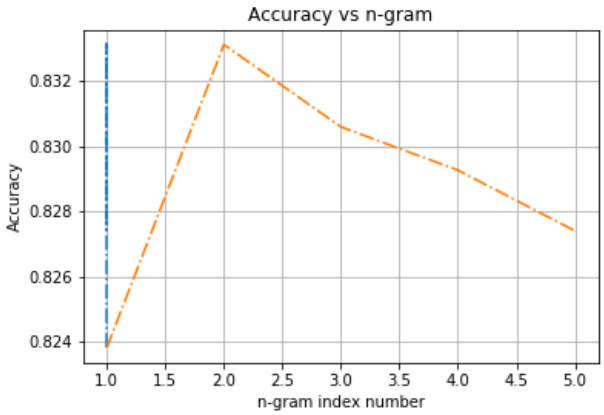
1. Changing the value of alpha:

As we change the value of alpha i.e. learning rate from lower to higher values we can see that the accuracy of the SVM classifier decreases. This is obvious and the result of the grid-search methods suggests us to use the value of alpha to be 0.001 as best considering the trade-off between time of execution and accuracy.



2. Changing the feature set (n-gram):

In this analysis n-gram corresponds to grouping of n words and taking their count in the CountVectorizer and this analysis corresponds to SVM classifier. Here the n values are taken in the order [1,2,3,4,5].
From the graph of the accuracy we can see that the accuracy is maximum at n =2. But the value that is suggested by Grid-Search algorithm is n=4. This is because at n = 4 it takes much less time as there are less number of 4-words group. Also the difference between the accuracies for n=2 and n=4 is not much significant (O(0.01%)).
It means that grouping of two words is very much favourable and it gives more accurate results than any other n-gram.



Conclusion:

Based on the implementation and analysis of above mentioned algorithms we can conclude that SVM is more suitable method than Naïve Bayes Classifier for Document classification. For SVM, the maximum efficiency for the trade-off between time and parameters is obtained when we take 4-gram words as features and learning rate(alpha) as 0.001.

References:

[1] DOCUMENT CLASSIFICATION Combining Structure and Content -Samaneh Chagheri, Sylvie Calabretto

[2] Automatic Document Classification - C. Goller, J. Löning, T. Will1, W. Wolff