TJHSST Computer Systems Lab Senior Research Project Paper Naive Bayes Classification 2008-2009

Christina Wallin
Period 3

January 15, 2009

Abstract

One part of computational linguistics is the classification and comparison of texts into classes. This comparison could be author classification, spam filtering, or in the case of this project, the classification of texts from different genres of news stories. The goal of the project is to be able to classify a text as belonging to one of two classes based on the actual words in the text. This method of classification is called a naive Bayes classifier. In this age of huge amounts of data available online, a classifier which could discriminate between two types of news could be increasingly useful.

Keywords: naive Bayes classifier, computational linguistics, binary independence model, Laplace smoothing, multivariate Bernoulli event model

1 Introduction

The naive Bayes classification is a relatively simple method for classifying texts. It is based on the false assumption that all of the variables, in this case words in the documents, are independent of each other. For example,

if the text was a science story and contained the word "rover," it would most probably also have the word "Mars." This project is done to achieve fundamental understanding concerning the effectiveness of the naive Bayes as compared to other methods, and to find a way of improving upon the performance of this classifier. However, it is also done to try to provide a way for news stories from different genres to be classified. In this age with huge amounts of data popping up on the internet every day, an effective way to sift through the texts with classification will be essential.

The Naive Bayes classifier works by first training the computer with data in order to find the relative probabilities of words for each class of texts. Then, because it makes the assumption that all of the variables are independent, it can find the probability that a specific text is in a class. At first, I have writen a single variable model, called a binary independence model because it compares only two classes, which does not take into account the frequency of the occurance of a word in the text, but merely its presence or absence. In latter quarters I hope to be able to optimize the algorithm by making a multivariate Bernoulli event model (explained later) and discriminating words based on stems instead of the actual form by using a stemmer (probably the Porter stemmer)–e.g. walking and walked would be counted as the same word–and perhaps, if time permits, coding another classification method in order to compare its effectiveness to that of the naive Bayes.

My prospective results would be the number/percentage of texts classified correctly. The 20 Newsgroup database is divided into training and test sets, so I will be able to use that as a constant for training and testing differnet versions of the program. The results would be comparing the effectiveness of the method for two different sets of genres of news (with the data coming from the 20 Newsgroups) and later comparing the effectiveness of the naive Bayes classifier with one using the multivariate Bernoulli model. Another thing currently considered before the other classification system is written is to determine whether or not it helps to stem the words using a Porter stemmer, and to investigate which sorts of files are more effectively classified by the naive Bayes. This study would add more data as to whether or not the Naive Bayes classification system, sometimes even called the "Idiot's Bayes," is effective compared to other methods.

2 Background

There have been many studies which have used the naive Bayesian classification method, for the method itself has been around for a very long time. The first paper that made the Bayesian assumption that all of the variables were independent was written in 1966: "Construction of a model for computer-assisted diagnosis: an application to the problem of non-toxic goitre" by Boyle et al. A reference to the previous paper was found in the paper "Idiot's Bayes–Not So Stupid After All?" by David Hand and Keming Yu, which using theoretical and real data situations shows that the Naive Bayes is not a horrible method because of its false assumption that all of the variables (in my case, occurances of words) are independent. This paper is essentially a review of past uses of naive Bayes and the conclusions of those researchers and a theoretical treatise as to why the naive Bayes is effective. Most of the studies talked about in this paper are medical diagnosis studies, and so text classification is a slightly different way of approaching naive Bayes classifiers.

Another element of this project is the fact that since it deals with the actual text of the words, it is also computational linguistics. So, it is actually the intersection of linguistics and machine learning. For a different linguistic classification problem, dialectology, or the study of how language varies over geographical regions, the study "Linguistic Variation and Composition" by John Nerbonne details how computational methods can be used to more effectively study language variation. One method which was used to increase the effectiveness of the computational linguistic techniques used was the Porter stemmer, which when given a string gives back its stem. The stemmer extracts the essential information out of a string and allows words which are just forms of the same word (e.g. stops, stopping) to be considered as such. This stemming eliminates morphological data, but it helps with understanding the lexical data, and for text classification the morphological part is inessential. Thus, I could see whether or not using a Porter stemmer would help with classification.

Several other studies talk about how best to use the naive Bayes method with text classification. In "Improving the Performance of Naive Bayes for Text Classification," by Yirong Shen and Jing Jiang, the authors explain a way to improve the naive Bayes by combining it with logistic regression, another method. The naive Bayes method itself is a bit overconfident in classification, and so by combinging it with logistic regression and using the Porter stemmer, the classification is improved.

There are also different types of words in a file, as described in "Text as Statistical Mechanics Object" by K. Korutchev and E. Korutcheva. The types are specific terms or keywords, which occur much more often in the file than in the general vocabulary, function words like "the" and "and," which are essential for meaning and occur rather frequently throughout the vocabulary, and common words, which have meaning but are not specific to a specific sort of text (for example, the word "explain"). Thus, if you find both the frequency of words for the general vocabulary and in a specific class, you can calculate some of the keywords of a document.

3 Procedures and methodology

3.1 Programming Language and Other Tools

The programming language used for these manipulations is Python. An aid for the linguistic sorts of operations is the nltk package, the Natural Language Toolkit. This package provides capabilities like the Porter stemmer programmed into it, and also has methods for displaying the data with the PyLab package. The PyLab package is a method of making graphs and charts, very much like matlab. Visuals can be made of the probabilities that selected words occur in each class, as compared to a text to be classified. With the multivariate model, a graph of selected frequencies of words in the sample text would also be beneficial. These charts would help the viewer understand the processes the computer is going through as it compares the text and the probability vectors.

The database which I will be using to classify texts is the 20 Newsgroups database. There are 20 different genres of news stories, and it is divided into training and testing sets. The training and testing sets are separated in time by a little, and so it is more realistic. Selected genres of news are comp.graphics, misc.forsale, rec.autos, rec.motorcycles, rec.sport.baseball, rec.sport.hockey, talk.politics.misc, talk.politics.guns, talk.politics.mideast, sci.crypt, sci.electronics, sci.med, sci.space, talk.religion.misc, alt.atheism, and soc.religion.christian.

3.2 First quarter version

For the first quarter version, I have focused on researching how to make a naive Bayes classifier and creating the first section of it. The first quarter version reads in a text and parses the string to remove all punctuation and capitalization. Then, it creates a dictionary of words occurring in that text and their frequencies. This step is in anticipation of the next part of the program, which is to create a probability vector (PFX) for each class. The program has been tested by using small data sets and checking manually whether or not the frequencies were correct. The portion which used regular expressions to remove punctuation was tested by printing out the parsed strings and seeing if the regular expression did the correct operation.

With this program, the frequencies of words in different sorts of texts can be compared. The differences in words in different genres of news is the basic premise behind the Naive Bayes classifier, and a visual representation with pylab of the top 20 frequent words is in Figures 1 and 2. The two news stories which were used were about space science and baseball respectively, and they were approximately the same length. One difference which is striking is that the baseball text has quite a few words which are the most frequent which deal with baseball, while the only word in the top 20 with respect to frequence which is even remotely science-related in the space article is "black". This difference suggests that the scientific writing is more verbose and less to the point than the baseball writing. This perhaps would not be the case with different articles in the same genre, but the correlation is interesting.

3.3 Current 2nd quarter version

In the 2nd quarter, I have completed writing a simple Naive Bayes classifier, the overall design of which can be found in Figure 3. The first step is to train the program as to what characteristics each class has. I did this with a probability vector (PFX) for each class based on the words occurring in them. This first model is not multivariate, which means that it does not take into account multiple occurances of a word in a document. So, in my project, the probability vector is a dictionary of all of the words occurring in the test cases with the probability for each variable (word) being the number of texts of the class which contain the word divided by the total number of texts in the class. Then, 1 needs to be added to both the top and the bottom, so that there is never a probability of zero that a term occurs. That process is called

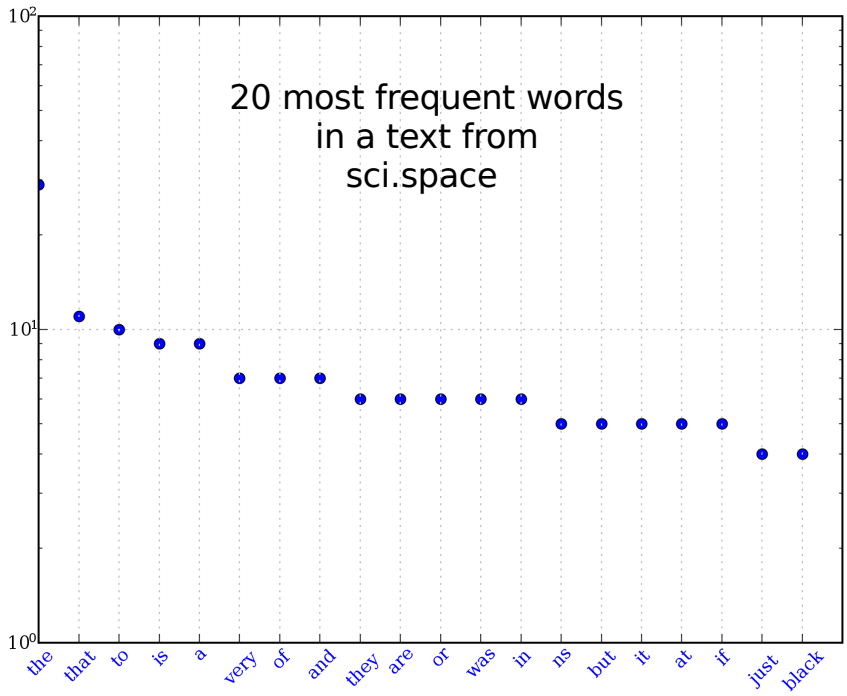

Figure 1: The frequency of the top twenty most frequent words in the sci.space genre from the 20 Newsgroups database, plotted with PyLab.

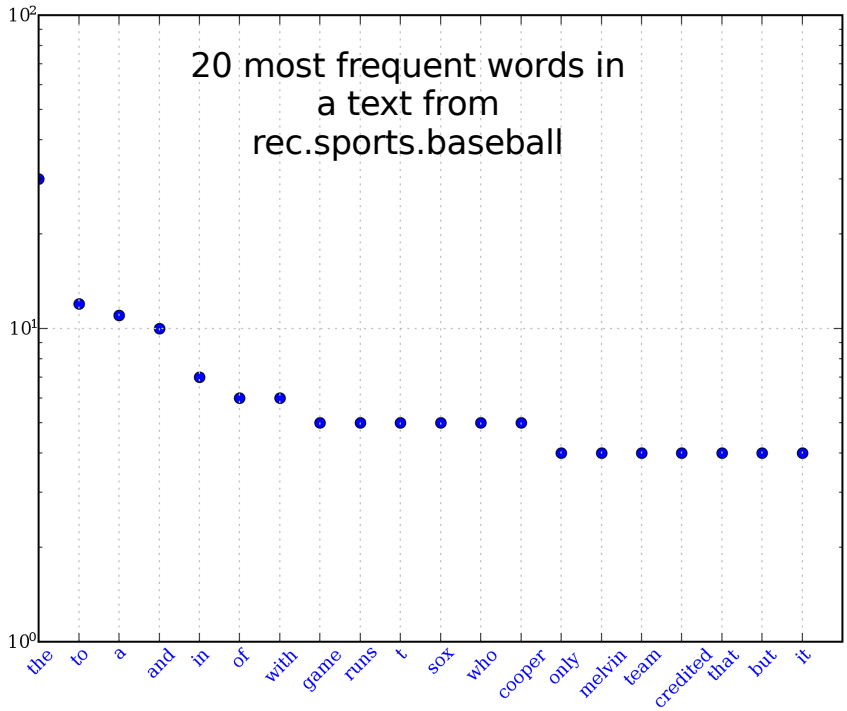

Figure 2: The frequency of the top twenty most frequent words in the rec.sports.baseball genre from the 20 Newsgroups database, plotted with PyLab.

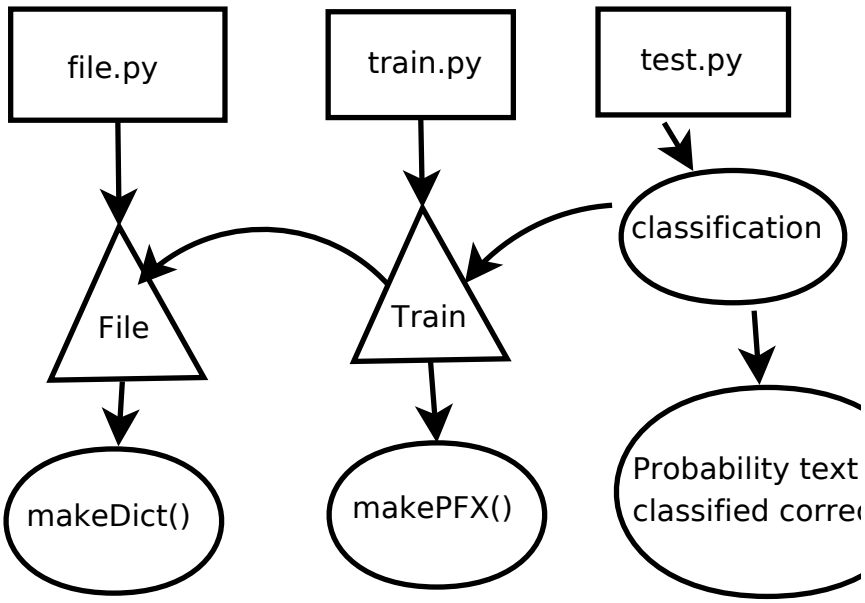

Figure 3: A model of the files and basic operations in the Naive Bayes classifier.

Laplace smoothing, and it accounts for words which are not found in the training cases but may be found in the test cases. Thus, for each individual text, the probability vector is binary, with a 1 if the word is in the text and a 0 if it is not.

Then with this probability vector for each class, by Bayes' theorum which states that each vector is independent, you can calculate the probability that a text is of a specific class by generating the probability vector for that specific text and comparing it to the PFX for each class, thereby classifying the the texts. For this, each variable (i.e. the occurance or non-occurance of each word) has to be calculated in order to form the probability that it is in a particular class. Then, the individual probabilities would usually be multiplied in order to determine what the probability would be. However, the multiplication can sometimes have problems, and so it is better to take the sum of the logarithms of the probabilities that the variable is an element of the class.

3.4 Future design

Once the second part of the project is completed and the program can convert from the PFX to the probability that a given text is in a class, the basic naive Bayes is complete. It can then be tested with different genres of news (see testing below). An extension is the multivariate Bernoulli event model, which essentially is the same thing except it takes into account the frequencies of the words within the texts themselves. I will also test further the effect of stemming the words (for example, changing "running" and "runs" to "run") and the impact on performance.

3.4.1 Bayes' Theory

For the most part I describe the algorithms above, but one thing which needs to be clarified is the assumption central to the naive Bayes classifier– Bayes' Theory. This theory states that all of the variables are independent in their determination of the class. In this case, the variables are individual words. The assumption is obviously false, because certain words usually occur together. For example, in a news article in the genre of news about space, if the word 'rover' occurs, the word 'Mars' is more likely to occur. However, even though the assumption is false the naive Bayes classifier is a rather effective tool for classification.

3.5 Testing

As far as testing goes, the 20 Newsgroup database from which the news stories come is separated into training and testing sets. The testing sets come in folders with the correct class in the name, and so with Python methods to check the folder from which the text came whether or not the class was correctly determined can be showed. However, the fact that the program produces the correct class does not mean that the algorithm is working correctly. So, the tests cannot solely be based on that.

One way to test the program is to test the components. You can see whether or not the probability vector is determined properly for a very small data set, and if it works on a small scale it should also work on a large scale. The naive Bayes itself can be tested with specific small data sets. One example for the multivariate model from http://nlp.stanford.edu/IR-book/html/htmledition/naive-bayes-text-classification-1.html contains only 4 training texts and one testing text for a total of 14 words, but it can classify certain words correctly.

I also tested the program by making sample data sets based on a programmed-in probability for each word. With this perfect data, I was able to check my PFX vector and found that it figured out the same probability as was programmed in. Thus, the probability calculation is correct. I was also able to check whether the testing part classified the perfect data correctly, which it did.

4 Results and End Material

With the simple Naive Bayes classifier, I tested the effect on the correct percentage of classification of stemming the words with a Porter stemmer. I expected that it would improve the classification, for it would allow words which are very similar to each other to be counted together. However, I found that it did not help classification to stem–it was actually approximately a percentage point less effective whenever the stemmer was used (Figure 4). I would expect that this is because when you stem the words, you are losing information about the words. Perhaps some genres use the past tense more, or something else like that, and so that information should not be lost. With the stemmer, 82.6 percent correctly classified vs 83.6 percent without in alt.atheism and rec.autos, 66.6 percent vs 67.7 percent in

comp.sys.ibm.pc.hardware and comp.sys.mac.hardware, and 69.3 percent vs 70.4 percent in sci.crypt and alt.atheism.

In the future, I can use Pylab to plot the percentage of test cases successful and failed for classification as a pie chart. I can also give the reader a view of what is happening with the algorithm by plotting the probability of selected words from each class, which is made with the training data. On this plot I can also include whether or not that word is present in a sample test case. The plot of the most probable words would also gain insight as to what words are key to each class, especially if that plot is adjusted as to the relative frequencies of the word in the global vocabulary.

Next quarter, I will also make a multivariate model and compare it with the binomial model, and I will try to account for the differences in lengths of text.

My prospective results could help other researchers know the relationship of the naive Bayes classifier to the multivariate Bernoulli event model and perhaps another classification system. They would be able to build upon this information by potentially doing another classification method (perhaps using neural networks) and using the same testing and training data and comparing the results.

References

[1] David J. Hand and Keming Yu, "Idiot's Bayes: Not So Stupid after All?', *International Statistical Review / Revue Internationale de Statistique 68*, pp. 385-398, December 2001.

[2] John Nerbonne, "Linguistic variation and computation", *EACL '03: Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics*, pp. 3–10, 2003.

[3] Yirong Shen and Jing Jiang, "Improving the Performance of Naive Bayes for Text Classification," CS224N Spring 2003

[4] K. Korutchev and E. Korutcheva, "Text as Statistical Mechanics Object," October 2008, arxiv.com

[5] David Lewis, "Naive (Bayes) at Forty: The Independence Assumption in Information Retrieval," AT@T research

[6] http://nltk.sourceforge.net/index.php/Book

[7] http://matplotlib.sourceforge.net/

[8] http://nlp.stanford.edu/IR-book/html/htmledition/naive-bayes-text-classification-1.html