Learning to Classify Documents
TJHSST Senior Research Project
Computer Systems Lab 2009-2010

Edwin Zhang

June 3, 2010

Abstract

This project uses a Bayesian method to classify documents into certain categories. A set of training data will be used to to derive a formula for probability. A set of features (words) specific to a certain topic and the conditional probability of the appearance of these features, using a formula, will be are used to determine the classification of documents of unknown categories. After some minor trouble and erros early in testing, my program worked for both 2 and 5 categories on a variety of documents with extremely high accuracy.

Keywords: Bayesian probability, document classification, conditional probability, features

1 Introduction

In this project, I will be using a method similar to the Nave Bayes Classifier to classify documents based on content. The Nave Bayes Classifier computes the conditional probability p(T—D) for a given document D for every topic T and assigns the document D to the topic with the largest conditional probability. This method, and my program, has two main parts: a Prediction and a Learning part. There are two major differences between my method and the Naive Bayesian Classifier. First, when I am calculating the scores and counts in the prediction part of my program, I am counting the number of times a term appears totally in all the training documents, whereas in the

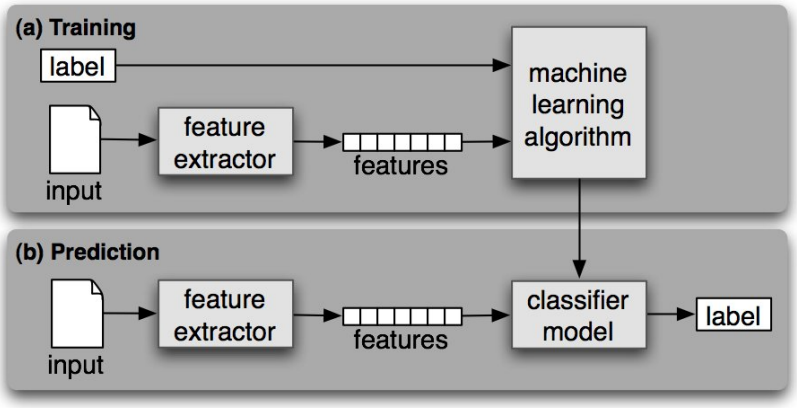

Figure 1: A visual summary of the algorithm for my project.

Naive Bayesian Classifier method, the program just counts the number of training document a term appears in. Second, my score calculation method during the Prediction section is slightly different, a lot of which is because of the way I did the counts in the Learning part.

I expected that initially, the program may have trouble classifying documents into the correct category due to errors and unoptimal formulas but as I test more and fix the errors and use a more optimal formula, the program will get better at classifying documents into the correct categories.

This topic is something that we use almost everyday, though we might not know it. When our email receives mail, it classifies the material as spam or not spam based on the content of the material. When we search using search engines, the search engine returns a list of materials related to what we searched for by looking at the content of the material.

2 Development

The program consists of two major steps: Learning and Prediction. The Learning part makes use of training documents to develop a formula for conditional probability, to be based on the probability that certain features appear in documents of similar topic. We will go through the training documents and look at how often a certain feature appears in a document that is about a certain topic. For example, if our topic is "tennis" and our feature if "winner" we would go through all the documents and see how often "winner"

occurs in documents about tennis and other documents. The Prediction part uses the results from the Learning portion to predict and classify the topic of an unknown document.

Right now, I am starting with only two categories: tennis and other. Once my program predicts correctly for two categories, then I will add more categories and keep testing. So far in my program, I have obtained my training documents for both categories and I have read them in. I created 3 new classes: Category, Documents, and Terms. My Category class deals with the different categories and holds all the training documents for specific categories. My Documents class deals with documents and all the terms that are in every document. My Terms class deals with terms, as well as the number of times each term appears in documents of different categories and assigns each term a score based on the term's counts in each category.

I have also created an array of categories. Each category has an array of documents, which each has an array of terms. In addition, I have an array of Terms containing every term that appears in all of my training documents and have the correct counts for each term. I have given each term a score for each category by dividing the number of times it appears in that category plus one over the number of times it appears in all the other categories combined plus one to avoid dividing by 0. I have also sorted my array of terms by the score for each term.

Next, I chose features for each category based on my array of Terms and the corresponding scores. I sorted the array for each category based on the score and took the first twenty-five terms that appeared and wrote them to separate files. That finished the Learning portion of the program.

After that, my program transitioned to the prediction part. I took the files containing the terms and the corresponding counts and scores and read them in and put the features into an array of Terms. I then had my program read in a document that I do not know the topic of. I go through the whole document and look for terms that are also features, for any category. I then divide the total number of times that features appears in my training documents over the total number of times it appears in all the training documents combined. I then take the log and do some other manipulations to make sure that the number I receive is positive and greater than 1. For each category I have a variable associated with that category that contains the "score" of that category with respect to the prediction document. I take the number I obtain and for each category, I multiply the number that I calculated with respect to that category to my variable. At the end, whichever category's

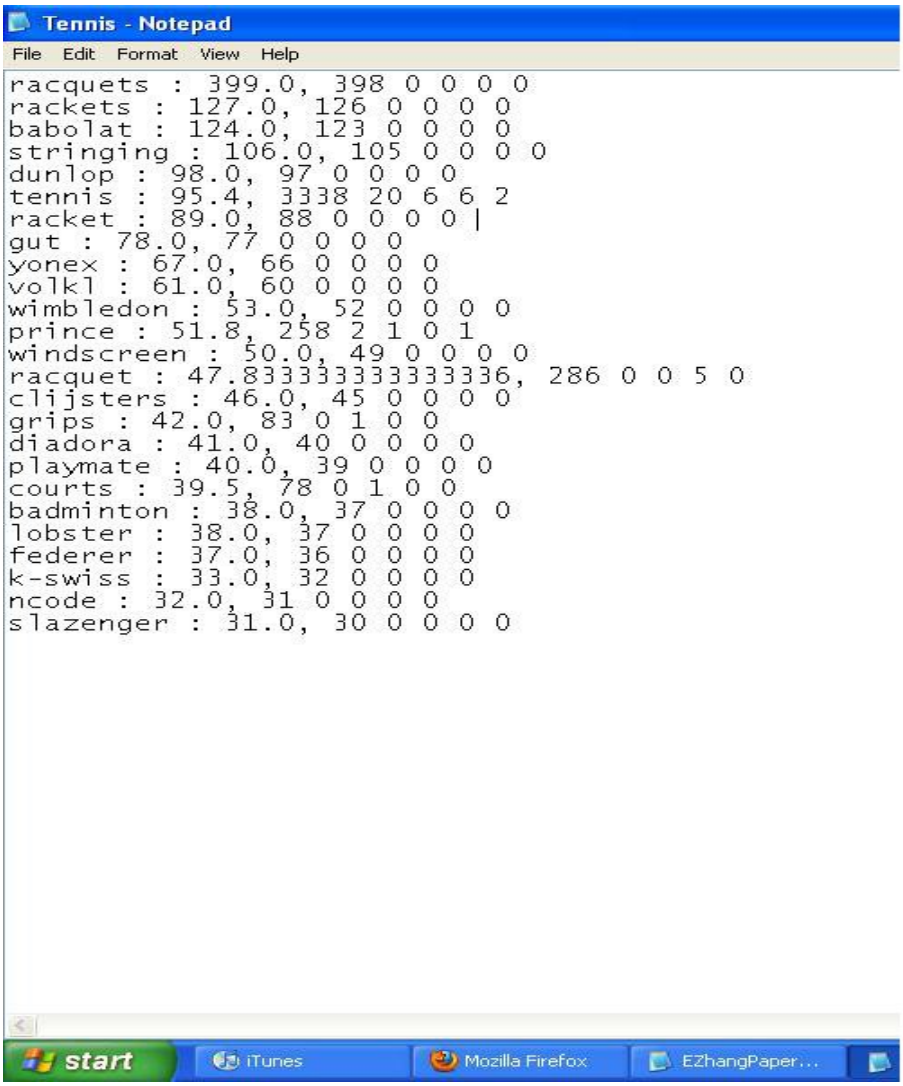

Figure 2: The features for tennis and the score and counts for each term.

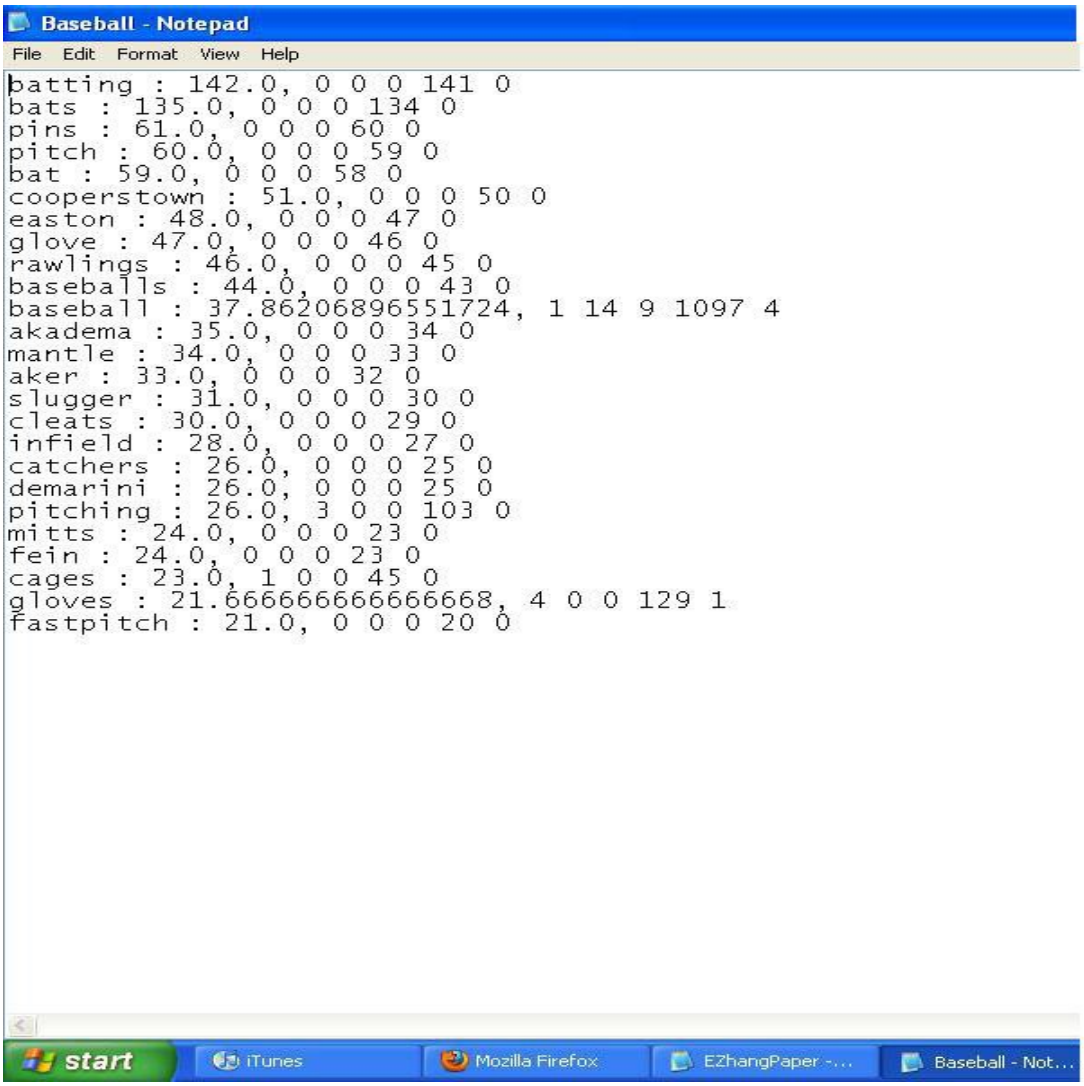

Figure 3: The features for baseball and the score and counts for each term.

variable had the highest number was the likely category of my document.

3 Results

I tested my program initially with two categories: tennis and other. After testing and revising my program several times, it finally worked for 2 categories, based on 10 documents. Once I went to 5 categories, however, my program started to malfunction again. After debugging my program, I realized my program was riddled with minor errors and after many small adjustments and corrections, my program worked for 5 categories, as well. I have only tested 25 to 30 documents with 5 categories, but I will get more prediction documents in the future to test so I can see how often my program works.

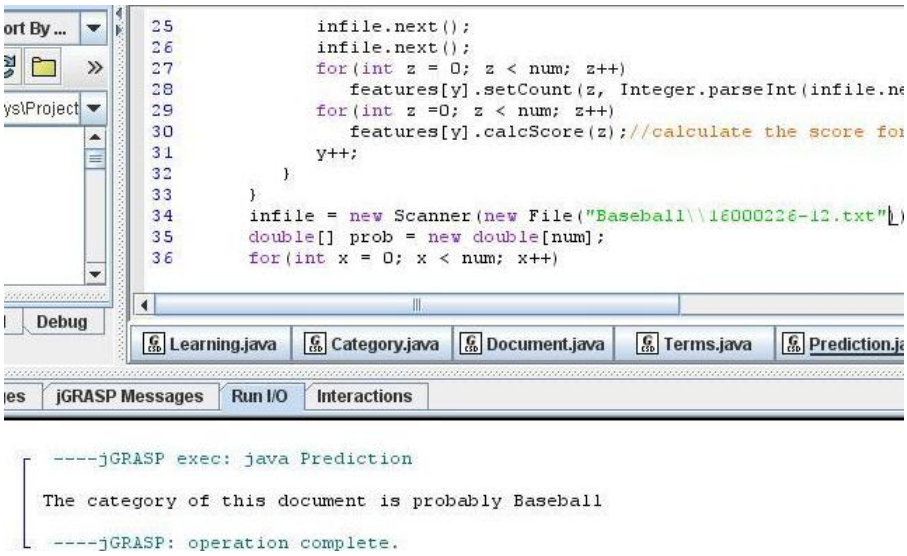

Figure 4: A sample output.

4 Discussion

I only tested my program on roughly 25-30 documents and it has worked for nearly every document, with just a couple exceptions. Unfortunately, it was not nearly as many documents as I had hoped to test. For a relatively small

sample size however, my program performed exceptionally well. However, there are several areas for improvement or further research. Adding more categories might affect my program. Five categories is not nearly as many as I would have liked. By adding more categories, terms that are currently features for a category may longer be features anymore, so that is one area that could have been explored further. I also expect the different methods of assigning scores will produce different results, so that is one area for further research. Another possible area to look into is when I am computing the conditional probability in the Prediction part of my program. Instead of dividing the count in one topic by the total count and taking the log, I could have done some other methods that would likely have produced different, though not necessarily better or worse, results. There are countless other possible areas that could be further examined later.

References

[1] Chai, Kian Ming Adam, Hai Leong Chieu, and Hwee Tou Ng. *ACM Portal.* Assocation of Computing Machinery, 2002. Web. 14 Jan. 2010. ¡http://portal.acm.org/citation.cfm?id=564376.564395coll=Portaldl=ACMCFID=70884224C

[2] De Pasquale, Jean-Frdric, and Jean-Guy Meunier. "Categorisation Techniques in Computer-Assisted Reading and Analysis of Texts (CARAT) in the Humanities." and the Humanities 37.1 (2003): 111-118. . Web. 25 Feb. 2010. ¡http://www.jstor.org/stable/30204883¿.

[3] Eyheramendy, Susana, and David Madigan. "A Flexible Bayesian Generalized Linear Model for Dichotomous Response Data with an Application to Text Categorization", *Lecture Notes-Monograph Series*, 54 (2007): 76-91. . Web. 25 Oct. 2009. ¡http://www.jstor.org/stable/20461460¿.

[4] Lavine, Michael, and Mike West. "A Bayesian Method for Classification and Discrimination." *Canadian Journal of Statistics* 20.4 (1992): 451-461. . Web. 14 Jan. 2010. ¡http://www.jstor.org/¿.