Sentiment Analysis

Ruchin Patel

Introduction

Proliferation of platforms that enable us to connect to Internet is facilitating an ever increasing number of ways in which we use the Internet. As a result, every day we are creating data at an unprecedented rate — so much so that according to estimates, nearly 90% of all of the digital data that exists today has been created in the last two years. A large portion of this data is text data — people updating their status on Facebook, millions of people tweeting about trending topics, people expressing their thoughts and opinions on blogs, news outlets publishing news.

At the turn of the century, as the Internet was being leveraged to conduct e-commerce, thousands of websites started coming up that would host reviews of these products from people who would buy these products online. E-Commerce companies realized that these reviews were critical to the user's decision as to whether or not they would purchase the product listed on their website. Companies such as Amazon started integrating these reviews in the products pages itself. This helped eliminate the need for user to branch off to other websites, in turn ensuring that the user spent more time on their website, thus leading to increased revenue. From the customer's perspective though, these reviews serve one purpose which to help in the decision making process of whether to buy the product or not. Often, however, these reviews can range from two lines about to entire paragraphs.

Ultimately, all the various degrees of information obtained from the review are used by the potential customer, in one way or the other, to classify the subject matter of the review in a single binary domain. This is at the core of Sentiment Classification which is the process of determining the emotional tone behind a series of words; gaining a very succinct understanding of the opinions expressed within a vast pool of information.

The goal of this project is to apply Sentiment Classification to reviews of various books available on Amazon.com. The motivation behind this application is that with thousands of reviews of each book, it might not be feasible for a potential customer to sift through all this information. Moreover, there are times when the customer isn't necessarily concerned with the details in the review, but rather needs actionable information on which they can base their decision to whether buy the book, or not.

Data

Our data set was provided by Center for Machine Learning and Intelligent Systems at UC Irvine.

There are reviews of eight books in our data set. For each book, the total number of reviews available range from 15,000 to 20,000. Of the total reviews available for each book, we have used our classifier on a sequence of subsets, wherein for each subset, 80% of all the reviews are used towards training the classifier and the rest 20% of the reviews are used to test the predictions of the classifier.

From here, we will go through each step of our analysis and explain why we made the decisions we did, then discuss the results of the classification problem.

Initial Thoughts

Before beginning to talk about the features of the data set and the algorithm used to design out text classifier, there are some general aspects of our algorithm that deserve mention.

Generally, when we talk of sentiment analysis, it follows that we will have to consider the grammar used in the sentence to be able to understand the message being conveyed by the sentence. If we consider the occurrence of the words and not the grammar, the perceived meaning of the sentence would be entirely differently. For example, consider the following two reviews for the same book:

1. This book is not bad and not at all not readable.
2. This book is good.

The meaning of both the sentences is same. However, a naive person would think that since the word 'not' is repeated thrice in the first sentence, the review for the book must be very bad.

Just like the naïve person in the example above, Naive Bayes algorithm for text classification does not consider the grammar of the sentence. Instead it considers as if all the words in the sentence are independent of each other. In other words, all permutation of the words in a sentence essentially mean the same sentence. It is as if the sentence is a bag and words are placed haphazardly in it. Since this algorithm considers such an assumption, that is why it is literally called "Naive".

Although the assumption that the features used in classification are independent of each other is usually false, analysis of Bayesian classification problems reveal theoretical reasons as to why Naïve Bayes classifiers are unreasonably effective. This is because of the following reason — while the probabilities estimated by Naïve Bayes are usually more than the theoretical values, this does not matter since we are only using the said probabilities to make decisions and not to precisely predict the theoretical probabilities. It is because of these 'Naive' assumptions that Naïve Bayes text classifiers are less expensive from the perspective of time required in execution when compared to other superior techniques such as boosted trees.

It is for the above reason that we are using Naive Bayes algorithm to design our classifier since despite the naïve assumptions, practically, it works just fine. The intuition for this becomes clear in the further sections where we talk about the algorithmic steps used to design our classifier.

Features

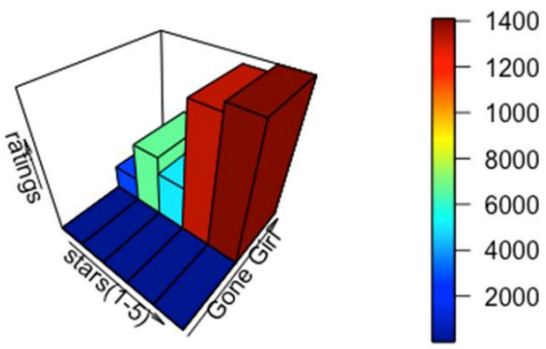


Figure 1 Graphical representation of the number of reviews of the book 'Gone Girl' that received ratings from 1-5 stars

For each of the eight books in our data set, the data set has the review along with the corresponding rating of the book given by a particular user for the book. These ratings are on a scale of 1-5, where 1 is the lowest rating and 5 is the highest rating. For example, the bar graph to the right graphically displays how many reviews from the total reviews for the book 'Gone Girl' were given 1 star, 2 star, 3 stars and so on.

To be able to classify all the reviews of a given book as 'good' or 'bad' with the help of a Naive Bayes classifier, we first needed to decide a threshold for the ratings from the original reviews that would deem a given review, and by extension the associated book, either as 'good' or 'bad'. After due deliberation, we decided that any review which has an associated star rating of 4 or 5 on a 5-point scale would be deemed as a good review. Any review with ratings below 4 would be considered to be a bad review. As a consequence of this consideration, our classifier will classify each review into two classes, namely, 'good review' and 'bad review'.

Feature Processing

Any review for each of the eight books in our data set is alphanumeric in nature. Thus, all the words/tokens in a given review do not give us any information about the review per se and thereby do not help us in classifying a particular review as 'good' or 'bad'. For example, stop words, verbs like is, conjunctions like and, or; and punctuations, numbers and special characters. Hence we remove all such works/characters from the review. The modified reviews which consider only the words which contribute meaning towards a review are referred to as the filtered review. We then find unique words from each filtered review and add them to the vocabulary.

Thus, the vocabulary for a particular book consists of unique words from each review for that book. Since on average we have 20,000 reviews for each book, it can be clearly seen that as a result of this the total number of unique words in all of the reviews for a particular book will also be very large. Moreover, in this bag of words(vocabulary) a majority of these words would occur only a small fraction across all the reviews. These words, which are not common to most reviews, will statistically not help us classify a review to the class 'good review' or 'bad review' given the word came

from that review. Since these words with less frequency do not contribute towards classification process, words occurring with frequencies less than 1% in the vocabulary are also removed.

Algorithm

For text classification, we use the unique words, that is, the tokens in a review to classify the review to either of the classes discussed above, namely, 'good review' and 'bad review'. If a 'Maximum a Posteriori' or MAP rule is used, the classifier can be described by the following formula:

$$c_{map} = \arg_{c \in C} max\,(P(c \mid r)) = \arg_{c \in C} max\,(P(c) \prod_{1 \le k \le n_r} P(t_k \mid c))$$

$$P(t_k \mid c) = \frac{T_{ct_k}}{\sum_{t' \in V} T_{ct'}}$$

Here, t_k refers to individual tokens, c refers to a specific class in the set of classes of classification C, $P(c \mid r)$ refers to the conditional probability that given a specific review r, the probability that it belong to specific class c, $P(c)$ refers to the probability of selecting the specific class c, $P(t_k \mid c)$ represents the conditional probability that a specific class c, probability that specific token t_k belongs to that class, n_r represents all the unique words that belong to a specific review, T_{ct_k} represent the total number of times token t_k appears in specific class c, and V represents the vocabulary described above.

To prevent floating point underflow in memory, logarithms are used in the above formulae. Moreover, to prevent the computer from computing values such as log(0) Laplace Smoothing is used. Hence, the modified formulae are given as follows:

$$c_{map} = \arg_{c \in C} max\,(\log P(c) + \sum_{1 \le k \le n_r} \log P(t_k \mid c))$$

$$P(t_k \mid c) = \frac{T_{ct_k} + 1}{\sum_{t' \in V} (T_{ct'} + 1)}$$

Once the vocabulary has been extracted from the reviews for a particular book, we can proceed to train the classifier described. For this, we first count the total number of reviews for a particular book. We then proceed to classify these reviews to either of the two classes mentioned above (feature processing). We now calculate the ratio of number of reviews that belong to a class $c \in C$ to the total number of reviews. This gives us the probability $P(c)\ \forall\ c \in C$. We then calculate the conditional probability $P(t_k \mid c)$. We first extract all the words that belong to reviews of a particular class. For the subset of these words that also happen to belong to the vocabulary, we calculate T_{ct_k}. From the formula above, the ratio for of $T_{ct_k} + 1$ to $\sum_{t' \in V}(T_{ct'} + 1)$ gives the said conditional probability. This is the end of the training phase. These computed values are stores and used during the testing phase.

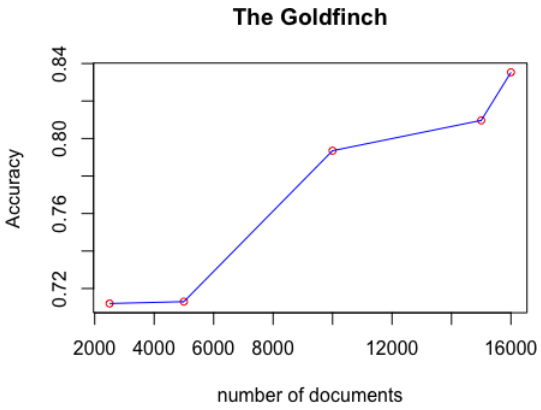
During the testing phase, for each review, we first extract a list of words from the review that also belong to the vocabulary. Then for each class, we compute $(\log P(c) + \sum_{1 \le k \le n_r} \log P(t_k \mid c))$ using the values determined in the previous steps. The review then belongs to the class for which the above expression yielded the maximum value.

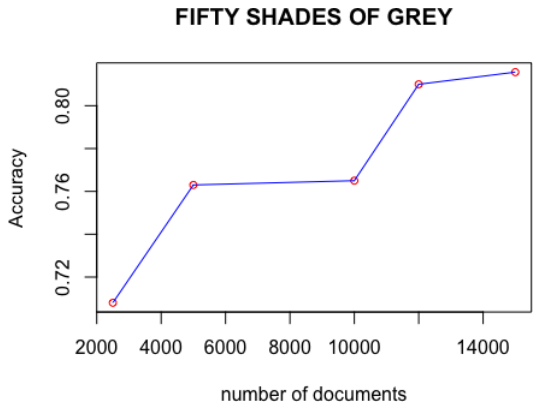
Results

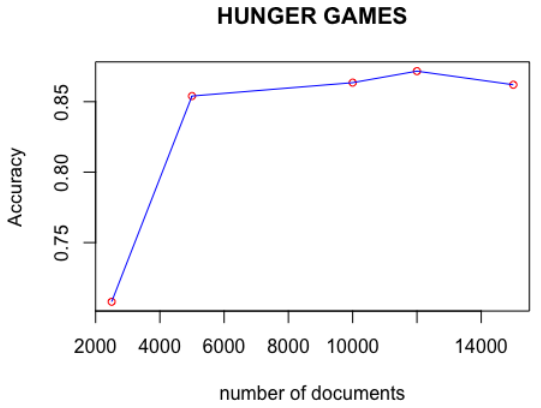
We successfully implemented and tested the Naïve Bayes Classifier on our data set of 8 books. Our Naive Bayes classifier classified each review of the test data set to a particular class, either 'good review' or 'bad review'. In doing so,

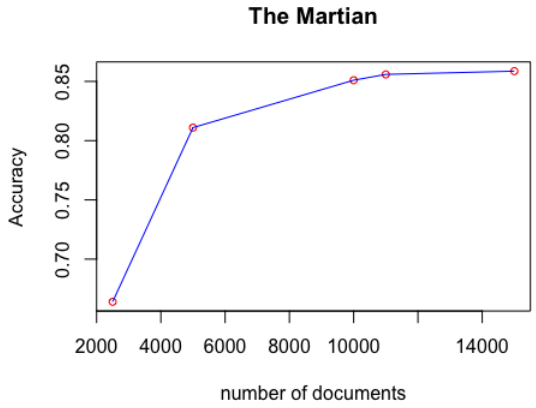
we estimated $P(c)$, which is inherently a marginal PMF and also estimated $P(t_k \mid c)$, which is inherently the ratio of joint PMF of (t_k, c) to the marginal PMF of c.

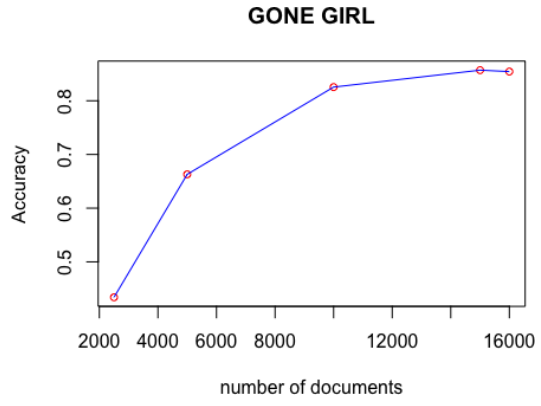
For every book's data set, we trained and tested our classifier on up to 5 subsets of varying sizes, where 80% of all the reviews in the subset were used towards training the classifier and the rest 20% of the reviews are used to test the predictions of the classifier. For each subset, we compared our predicted results with the theoretical results for the corresponding subset. For each of the 8 books, the variation of accuracy of prediction of our classifier as the size of the data set increases is illustrated as follows:

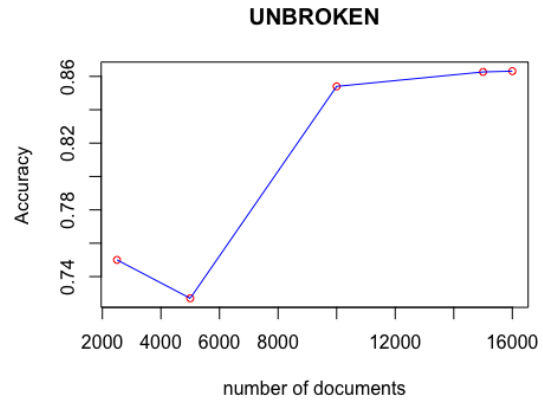


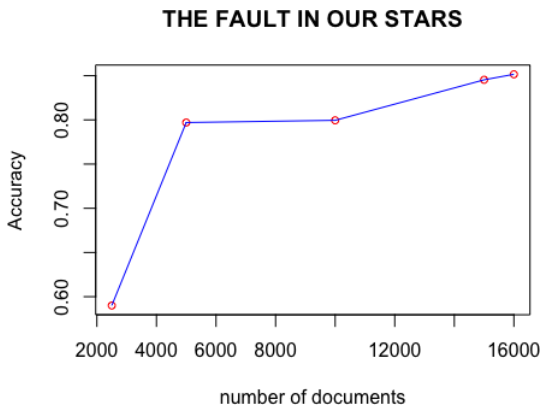


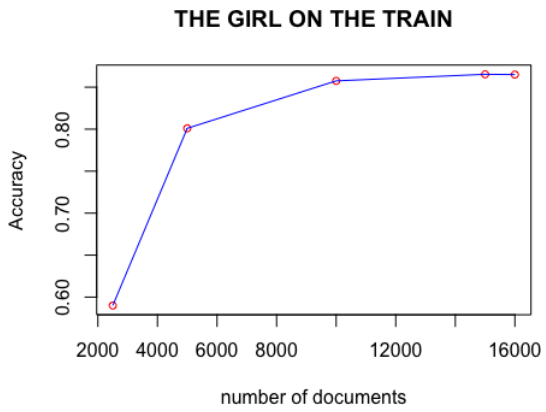












Conclusions

As can been seen from the graphs above, as the size of the dataset increases, the accuracy of our classifier also increases. Particularly, in reference to section ***Data*****,** in can be concluded that the accuracy of our classifier increases as the absolute size of the training data set increases.

Also, it may be noted that for the limiting case when the biggest subset of the total number of available reviews was used to train the data set, for each of the eight books/data set, at minimum, the **accuracy** of our classifier is greater than **80%.** Hence we may conclude that the **predictions** of our classifier **are good**.

Future Scope

The steps and algorithms described in this project are general in nature and can inherently be extended to all applications that may benefit from sentiment analysis with little to no modifications. These applications may include sentiment analysis of tweets on twitter pertaining to a domain like the United states presidential election, which could give us information on how well was a candidate's speech received, whether a recent revelation about the candidate affected the approval rating of the candidate or sentiment analysis of news article on a particular subject from different news website can help us determine about the gravity of the situation of that particular subject

Acknowledgments

We'd like to thank the Center for Machine Learning and Intelligent Systems at UC Irvine for providing useful data set for Machine Learning applications .

We'd also like to thank Mohammad Reza Rajati and the Teaching Assistants for an enriching semester, and for giving us the toolset to accomplish so much in such a little time.

Sources

Kevin P. Murphy, Machine Learning: A Probabilistic Perspective

Machine Learning Repository, Center for Machine Learning and Intelligent Systems at UC Irvine