Machine Problem 1 - Multinomial Naive Bayes and Subsequent Optimizations

Eddie Zhang, ete@ucsb.edu, 9423757

May 2020

1 Architecture

I think to be successful in creating a classifier one needs to prioritize development time over computer time. In other words, the faster the developer can try out new ideas, the more successful he will likely be, even if each iteration takes an extra 30 or 60 seconds. That's why I chose python. In addition, I separated out my training phase and testing phase, so that I could experiment with one and not affect the other. Several iterations of the model were created, and each major breaking change I created a new branch for. In the most recent model, you'll see that there are plenty of features such as log IDF, a tuned smoothing parameter, optimized stop word list, bigram/trigram, and something very curious: a split bayes classifier. I'll go into more detail about this last feature later.

2 Preprocessing

Actually I do very little in the preprocessing phase, other than feeding the documents through a stop word list. I tried both TF-IDF and Stemming, but it seemed that both of these only hurt performance rather than improve it. However, I do add on the features of bigram and trigram, along with the length of the document. I'll get into more of this in the next section. To find which stop words to use, I took every word with occurrences greater than 1000 (approx. 10% of the total training set vocabulary), and ran an experiment to see what accuracy could be achieved by using just that one word as the stop word filter. I found that simply adding all the words that improved the baseline accuracy scaled the performance logarithmically to the amount of words added. To run the experiment, I created a multithreaded wrapper function that called my classifier with each different stop word - I then modified the classifier to send the accuracy to a database in the cloud, with the key as the word and the value as the associated accuracy. Figure 1 illustrates how performance changed with respect to each stop word. I did the same with the hyperparameter for smoothing, as illustrated in Figure 2

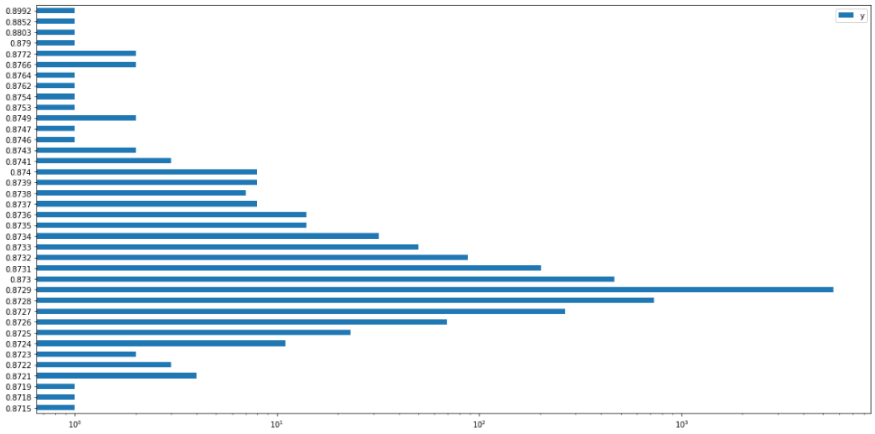


Figure 1: Frequency of accuracy, given a candidate stopword

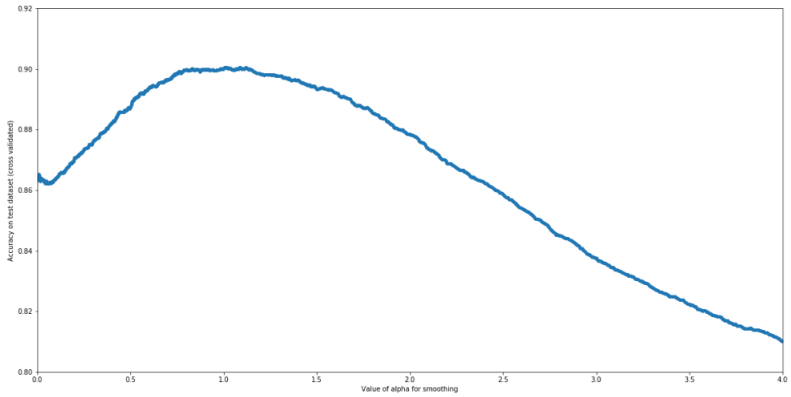
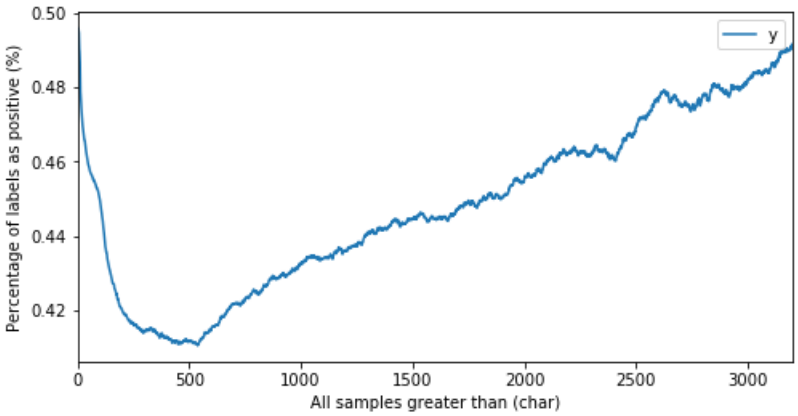
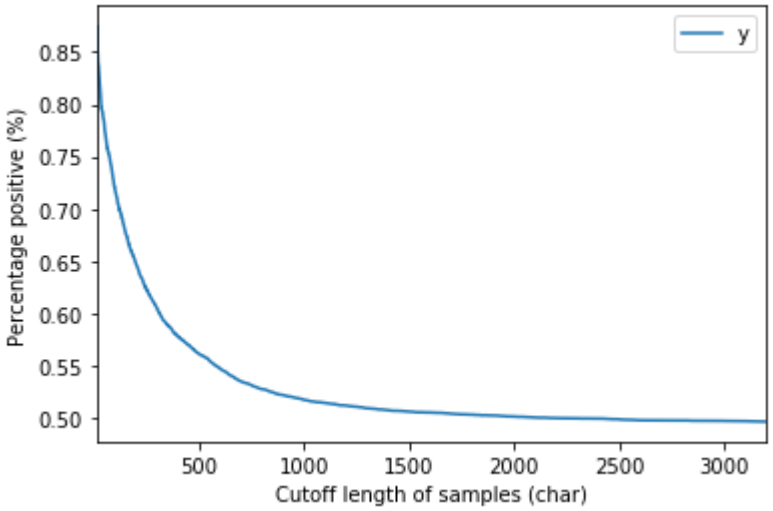


Figure 2: Accuracy with respect to smoothing parameter

3 Model Building

The only thing I would like to mention here is that I found there was a very nice pattern of percentage of positive reviews with respect to length of review, and I was able to exploit that by splitting the dataset into two parts: short length reviews and long length reviews. I then effectively trained two models, one for short and one for long. Figure 3 explains why I chose to split at length 500 Figure 4 illustrates the pattern described above.





4 Results

As my accuracy got progressively better and better, my speed got progressively slower and slower. I think this is unavoidable in the long term, and unless much time is spent on optimization this can only be expected. I achieve around 99.5% accuracy on the training dataset, and 90.4% on the testing dataset. On my last iteration, I take around 39 seconds to train and 69 seconds for labelling. Top 10 most biasing features for positive (not necessarily in order): 1. Excellent 2. Awesome 3. Amazing 4. Perfect 5. Favorite 6. Highly 7. Great 8. Love 9. ds (?) 10. Best Top 10 most biasing features for negative (not necessarily in order): 1. Boring 2. EA 3. Worst 4. Terrible 5. Horrible 6. Waste 7. Stupid 8. Poor 9. Worse 10. Wouldn

5 Challenges

Sometimes it got very frustrating when the model inexplicably dropped in accuracy, or when a change that I had spend several hours on didn't end up improving accuracy. Additionally, waiting for the model to test and train over each iteration became very painful, as I would more or less have to sit around twiddling my thumbs. Overall though, I think this project has made a much more resilient developer in the sense that I won't give up quite as easily when encountering a bug, and I have definitely thought about more ways to attack this problem than I ever have before. This was a very good experience, and I'm ready for machine problem 2!

6 Weaknesses

From the training accuracy to testing accuracy you can definitely see some overfitting happening, and although I explored using some advanced techniques to try to understand Naive Bayes as a classifier in space and use that to try and maybe smooth the data, it never panned out. Additionally, the model is fairly slow so it wouldn't win any competitions in speed.

7 Conclusion

"I always thought something was fundamentally wrong with the universe" [1]

References

[1] D. Adams. *The Hitchhiker's Guide to the Galaxy.* San Val, 1995.