COMP 472 Project 2 Report

Alexia Soucy[1]

[1] 40014822 a_oucy@live.concordia.ca

1 Introduction

This report concerns the development of a basic Naive-Bayes classifier based on the algorithm outlined in the lecture slides [1] and the technical considerations therein. The application uses a text file containing collected and formatted data for its training and evaluation. There is no built-in assumption regarding the number or character of labels found in the file.

1.1 Technical Notes

The classifier makes use of true division imported from the __future__ module to compute probabilities for its classification as well as the modified open function from the codecs module to ensure proper file encoding.

Additionally, it uses the built-in Counter data structure to count the frequency of word occurrences in the data it examines. The classifier makes use of log probabilities to avoid rounding errors and as such requires the NumPy library's log function.

Finally, the sys and os modules are imported to ensure the training file is read from the script's folder rather than Python's working folder, allowing for added portability across Python installations, as well as to allow for additional functionality in terms of custom training and evaluation files supplied by the user at runtime in the form of arguments (argv[1] and argv[2] respectively). If only a training file is provided, it will be split with 80% used for training and 20% used for evaluation per the project specs, but if two files are provided, the first will entirely be used for training and the second entirely for evaluation. If the training file is not a text file or if the file does not exist, the standard file is used for both training and evaluation. For the evaluation file, if either of these errors occur, the provided training file is used for both training and evaluation. The proper formatting of these files is left up to the user. These files must share the same folder as the script.

2 Parameters

The primary parameter considered for optimization in this project was the smoothing factor. Since the algorithm utilizes log probabilities, there is a necessity for a smoothing factor of some kind to avoid division-by-zero errors. [2] To implement the smoothing

factor, standard additive smoothing was used as shown in (1), where V is the entire vocabulary and δ is the smoothing factor.

$$P(w_i|c_j) = \frac{P(w_i,c_j) + \delta}{P(c_j) + \delta|V|} \tag{1}$$

2.1 Smoothing Optimization

A simple loop was executed to find the optimal smoothing factor, iterating through possible values of δ to finally determine that the optimal smoothing factor for this dataset was approximately 4.1 (see Fig. 1).

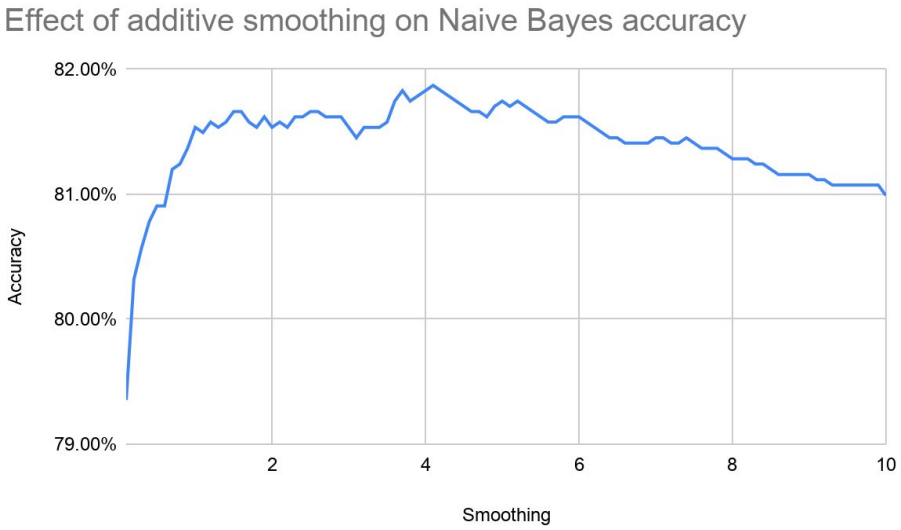

Fig. 1. Effect of additive smoothing on Naive Bayes accuracy

Working on the assumption that the dataset is a sufficiently large population of product reviews to serve as an accurate statistical sample, it is relatively safe to assert that this optimized smoothing factor will work for any set of product reviews. This is not verified, however, and it may result in overfitted data.

3 Results

With the smoothing factor obtained in section 2.1, the overall accuracy of the classifier when evaluated on the provided documents is of approximately 81.87%, as opposed to 81.54% with add-1 smoothing. Furthermore, the classifier appears to have more ease classifying positive reviews than negative ones (see Table 1). Whether this is due to

quirks in the English language, coincidental qualities of the sample, or the nature of product reviews as a cultural artifact remains unknown.

Table 1. Classification results

	TRUE	FALSE
POS	83.10%	16.90%
NEG	80.83%	19.17%

Using the demo file provided, the trained algorithm manages an accuracy of 66.67%. This does not change with a lower smoothing factor, but the entries for which errors occur vary slightly, indicating that a larger data set would possibly display a discrepancy in accuracy. It is worth noting that the sample used is quite small (featuring only 12 documents), meaning each error has a high impact on the overall perceived accuracy of the algorithm. In this case, only 4 errors account for 33.33% of the sample.

3.1 Analysis

A few critical points regarding the Naive Bayes classification methodology can be drawn from the results obtained with this sample's evaluation testing and demo sample evaluation.

First, since Naive Bayes is fundamentally based on strong independence assumption between the statistical units of analysis, compound phrases that are majoritarily composed of words whose polarity is counter to the overall expression's will run statistically counter to their intended polarity. For example, the phrase "not very good" will have a tendency to be labeled as positive since each word is analyzed only in isolation rather than in relation to its neighbours. This shows how a reviewer's writing style can easily influence the classifier's functioning.

Second, entries that use many words that don't match the overall polarity of the entry can fool the classifier. For example, a positive product review that lists negative points that the user would like to see improved will be mislabeled as negative. This problem runs deeper than simple statistical analysis and would require an understanding of the structure of a product review embedded in the system.

Third, reviews based on expectations that go counter to normal attitudes will be mislabeled by a statistical model such as a Naive Bayes classifier. For example, a prayer book review in which the user cites the writing's lyrical and intellectual quality is classified positively by the system even if the user is citing these qualities as negatives.

Fourth, the size of a review is statistically meaningful; individual words in each review will have a much greater impact on the classification in a shorter review.

Lastly, the context of the document analyzed matters as if it does not fit in the same context as the training sample (such as if its source has a different level of expertise), which was the general public in this case, the vocabulary used will be harder to categorize.

4 Challenges

The greatest challenge of this project's execution has been the at times unintuitive nature of statistical models; the implementation of the algorithm itself is relatively simple once understood, but the underlying mathematics at times proved unwieldy to the untrained eye and result analysis therefore took longer than expected.

5 Further Development Considerations

Given more time, it would have been possible to explore modifications to this project to implement k-fold cross-validation and look for an optimal training mechanism for the classifier. Furthermore, experimenting with different training datasets and using the current dataset to classify new data to test the working hypothesis on which the smoothing optimization detailed in section 2.1 is based would surely be insightful.

6 References

1. Houari, N.: Artificial Intelligence: Machine Learning - 1. Lecture slides, pp. 54 (2019)
2. Houari, N.: Artificial Intelligence: Machine Learning - 1. Lecture slides, pp. 57-58 (2019)

7 List of Figures

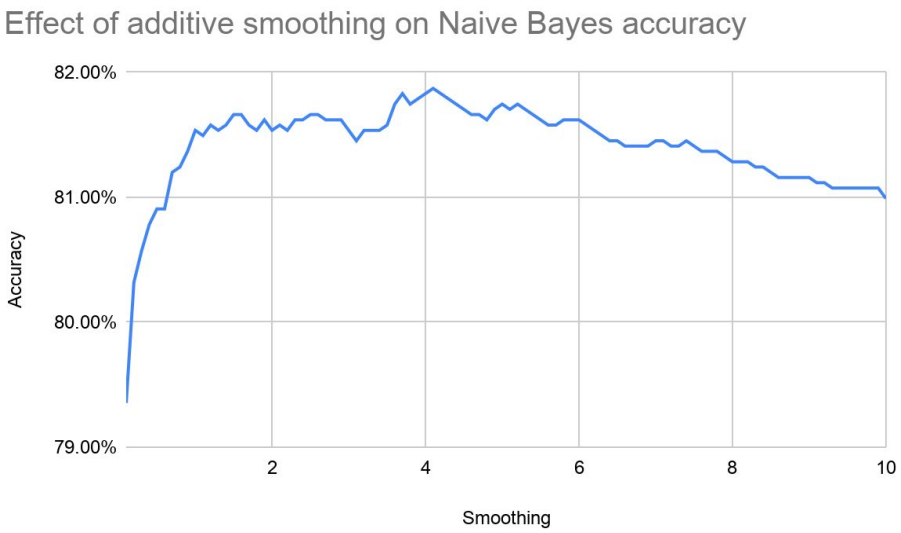

Fig. 1. Effect of additive smoothing on Naive Bayes accuracy