COMP 472 Project 2 Report

1 Task 1 Analysis and Results

In task one we train the Naïve Bayes classifier, by using a training set of docs.

1.1 Output of classifier

The classifier essentially has two outputs which it uses to help it classify the documents:

1. A table with the probability of each label among all the documents. This is *P(label)*
2. A probability table that tells us the probability of each vocabulary word given the category. This is *P(word | label)*

In my case the classifier is a class which stores these values as attributes of the object instance.

1.2 Choosing A Smoothing Factor

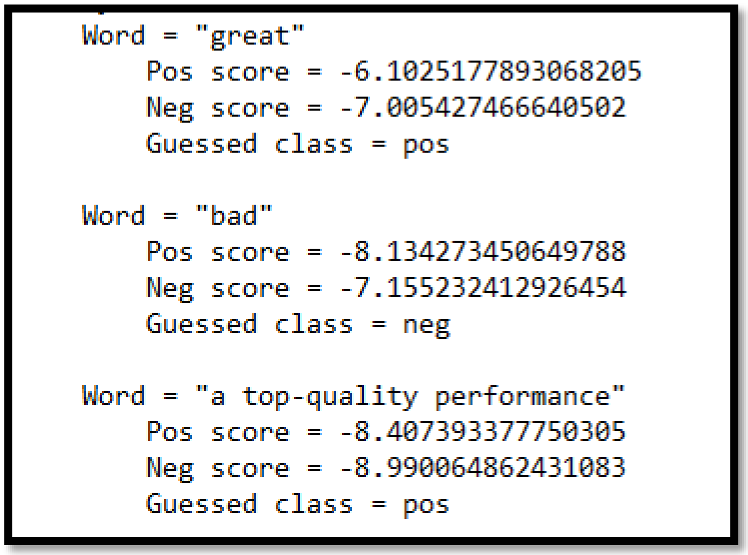
The purpose of the smoothing factor is to first ensure the classifier actually runs. If a word in the vocabulary is not found for a certain class the probability would calculated as *log(P(word_not_in_label | label))* → *log(0)*. This causes an error since there is no such result

It's main purpose though is to allow words that never show up in the doc to not have a strong influence in telling the classifier that the lack of the word indicates the label is not the correct one. Since there are many words in the vocabulary with low frequencies. The lack of these words should not cause the classifier to choose a different label.

Upon testing different smoothing factors, I found I got the best accuracy when I used a smoothing factor of about 2.6.

1.2 Sample Results

Here is an example of the scores given by the classifier:



```
Word = "great"
    Pos score = -6.1025177893068205
    Neg score = -7.00542746664050
    Guessed class = pos

Word = "bad"
    Pos score = -8.134273450649788
    Neg score = -7.155232412926454
    Guessed class = neg

Word = "a top-quality performance"
    Pos score = -8.407393377750305
    Neg score = -8.990064862431083
    Guessed class = pos
```

1.3 Analysis

The classifier tends to generally give a negative score. This is because there are a lot of words in the vocabulary (around 50,000 words) and a lot of them do not show often in the whole sample of docs. Even though this is the case, we can see that when the classifier sees that the word is a positive word, the positive label has a greater score.

2 Task 2 Analysis and Results

2.1 Results

	Test = pos	Test = neg
Guess = pos	920	188
Guess = neg	233	1024

Here are the results for one of the runs when classifying the test set using the trained classifier. The overall accuracy of this run is 82.33%. The accuracy of the positive results is 79.80% and the accuracy for negative review results is 84.49%. This was received with an 80-20 split between the training and test set with a 2.6 smoothing factor

2.2 Analysis

The overall accuracy of the classifier was pretty good but not perfect at around 82%. When looking at the incorrectly classified documents, some of them had cases where it just so happened that the doc had words that were generally associated with positive reviews even if they were negative and vice versa. For example, there are cases where the reviewer talks about a negative experience they had, but then claim the product was useful in helping them get out of that negative experience.

In, general the main issue is that a Naïve Bayes algorithm has no concept of association between words, since the **main assumption** of the Naïve Bayes algorithm is that **all features are independent**. When it comes to the words in a paragraph or sentence, this is generally not true though. We have structure to the sentences we write and in general when we talk about a certain topic, certain words are more or less likely to show up. This shows that even with a good Naïve Bayes algorithm, there will always be a bottleneck as it also has this assumption and the reality is that words are not independent of each other in a sentence.

Another issue is that these docs seemingly come from one source given the similarity of some of the reviews and topics. This means that our trained classifier was bound to get overfit to the documents it was given rather than have a broad idea of what a positive and negative review is.

3 Difficulties

When creating my Naïve Bayes Classifier, the main issues I came across were performance issues and trying my best to improve the accuracy.

3.1 Performance

The first issue I kept coming across was the classifier taking way to long to classify its results. After discussing with other students, I found that the issue was that I was using a Python List.count() function to get the frequency of vocabulary words for each doc. To fix this I had to replace that count with the use of the Counter class which was suggested in the project instructions.

The second performance issue I came across was when I was classifying the documents in my test set. When I first created my classifier, I decided to store only the frequency table of the vocabulary words for each label and calculate the probabilities when I classify. This did not work well though since the test set itself was quite large too. In this case I simple added and also stored the probability table. Now I only needed to access the stored probabilities resulting in much better performance.

3.2 Accuracy Improvements

When I first created my classifier, it had an accuracy of around 78%. I wanted to improve this and thought that maybe some filtering of the vocabulary would help improve the accuracy. There were two things I did for this.

The first thing I tried was to remove words that contained the characters in a-z only one time. This filtered out all very short words and non-textual words. For example, commas (,) were one of the characters being added to the vocabulary, even though they have no influence on whether a person is writing a positive or negative review. Upon doing this my classifier would now get about 81% accuracy.

The second thing I did was add a package in my project called “stop-words”[1]. This package simply allowed me to get a list of common English words, for example “the” and “it”. These words are common in the documents, but tend not to have any meaning for whether the review was positive or negative. In general, this made my accuracy stick to around 81 to 82% and as a bonus it also increased performance since the classifier was able to ignore any of these words as they weren’t added as a part of the vocabulary.

4 Future Prospects and Questions

While I did look into certain ideas that could help improve the accuracy and performance of the algorithm, it would be interesting to see and learn about other methods of cleaning up the vocabulary to better fit a broader set of reviews.

Another interesting idea would be to see what happens if we train our classifiers with a different set of reviews and see how they classify docs in comparison.

In my classifier I also made the labels generic, so I would be interested in how my classifier would work given more labels to categorize. Would it do better or worse. I have a feeling it could do worse given that more categories means less documents for each category.

Finally, I wonder how other forms of Ai learning would apply to this same task. Would deep learning have better accuracy for such a task, or how about decision trees.

References

[1] - https://pypi.org/project/stop-words/