Homework 3: Naive Bayes

Shamya Karumbaiah, Collaborated with - Rafael Lizarralde

October 7, 2016

1

1.1

252165

1.2

Top 10 positive words - the, and, a, of, to, is, in, i, it, that
Top 10 negative words - the, a, and, of, to, is, in, i, this, that
In general, these top 10 words would look similar to these in most english texts and has very little to add to the discrimination based on the label.

2

2.1

Nothing to report

2.2

$P(fantastic|+) = 1.54458162793e-04$
$P(fantastic|-) = 3.77720191813e-05$
$P(boring|+) = 6.18508616873e-05$
$P(boring|-) = 2.87275265149e-04$
Looking at the relative probabilities, it seems intuitive that fantastic is getting a higher probability for positive label and boring is getting a higher probability for negative label.

2.3

When we try to compute the probability of a word present only in one class, its probability for the other class would become zero if we take only the raw counts. This makes it look like that word could never occur in the other class. The words "not known" in that class may be just so because our training data is not exhaustive. Having a zero probability for such words would be incorrect. Any document we try to test in a new dataset which contains this "unknown" word would make that sequence's probability 0. Also, as we see later, we would use log to prevent underflow. Having a 0 probability would make the log undefined.

3

3.1

$$P(w_{d1}, ..., w_{dn}|y_d) = \prod_{i=1}^{n} P(w_{di}|y_d)$$

Taking log on both sides, the log likelihood is given by -

$$logP(w_{d1}, ..., w_{dn}|y_d) = \sum_{i=1}^{n} logP(w_{di}|y_d)$$

4

4.1

$$P(\boldsymbol{w_d}) = P(w_{d1}, w_{d2}, ...w_{dn})$$

By chain rule,

$$P(\boldsymbol{w_d}) = P(w_{dn}|w_{d1}, ...w_{dn-1})P(w_{dn-1}|w_{d1}, ...w_{dn-2})....P(w_{d1})$$

Since, we don't worry about the position of the words in the document as per our naive assumption, we could use conditional independence to simplify this,

$$P(\boldsymbol{w_d}) = P(w_{dn})P(w_{dn-1})....P(w_{d1})$$

$$= \prod_{i=1}^{n} P(w_{di})$$

4.2

We have the posterior probability,

$$P(y_d|\boldsymbol{w_d}) = \frac{P(y_d)P(\boldsymbol{w_d}|y_d)}{P(\boldsymbol{w_d})}$$

Taking log, we get the log posterior probability as,

$$logP(y_d|\boldsymbol{w_d}) = logP(y_d) + logP(\boldsymbol{w_d}|y_d) - logP(\boldsymbol{w_d})$$

Substituting the simplified equations, we get -

$$logP(y_d|\boldsymbol{w_d}) = logP(y_d) + \sum_{i=1}^{n} logP(w_{di}|y_d) - \sum_{i=1}^{n} logP(w_{di})$$

4.3

The normalizer is going to be the same for all the classes (y_d) given the word sequence ($\boldsymbol{w_d}$) as $P(\boldsymbol{w_d})$ doesn't change for different labels. So, having this extra term doesn't make a difference to the posterior probabilities.

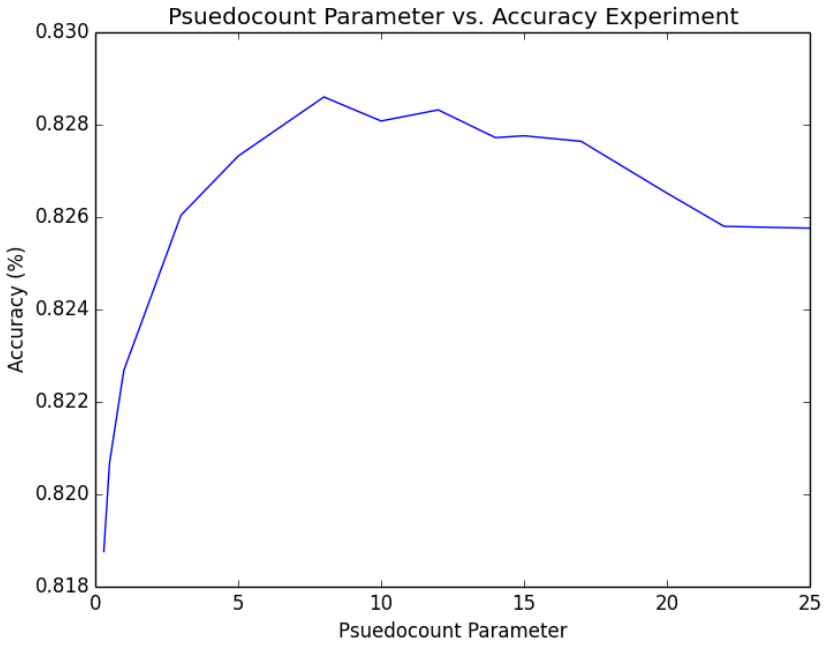
5

5.1

Classifier accuracy with alpha = 1.0 is 0.82268

5.2

Best alpha value was 7.5. See plot below -



5.3

10009_10.txt is an example of misclassification. This is a positive test case being classified as negative. One of the reasons this happened could be the part of the review where the reviewer is talking about the channel he viewed the movie in and not the movie itself. For example, "..*I usually hate having satellite but this was a perk of having satellite*...". The word "hate" would have a very low probability for "positive" class contributing to a low likelihood. The naive assumption of each word being conditionally independent of others adds to this. Emphasizing on the sequence of the word occurrence could have helped here. Also, having a long term dependency to recognize the object of the sentence could have helped here.

6

6.1

The range of LR function is $[0, \infty]$ assuming there is no laplacian smoothing in which the probabilities could be 0. In case smoothed function, the range would be $(0, \infty)$.

6.2

Nothing to report

6.3

Here are the likelihood ratios (with alpha = 1.0) -
fantastic LR: 4.06898088596
boring LR: 0.216646954101
the LR: 1.03611983814
to LR: 0.94529239345

As expected, ”fantastic” has an LR of around 4 making it 4 times more likely to appear in a positive review than in negative. Similarly, ”boring” is very unlikely to appear in a positive review. The neutral words which are very common to occur in both review get a score around 1 like ”the” and ”to”.

6.4

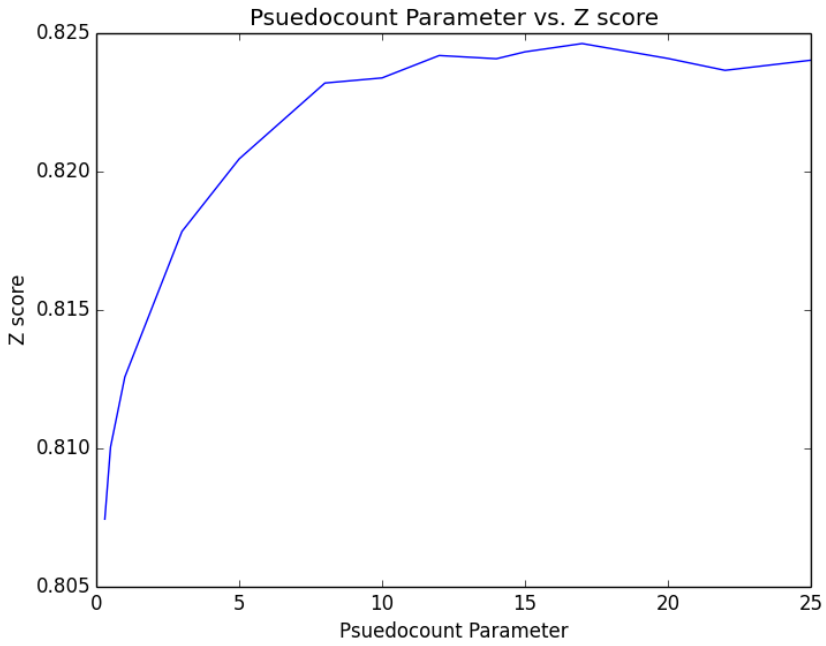
The LR value of a word denotes the factor to which that word is contributing to the predictive power of the classifier. If we look at the words as the features, a high or low LR valued words are the more discriminative features than the ones with value around 1 which contributes equally to both the classes. For example, higher the number of words with say LR=100, the more confident the classifier is about its prediction of positive label as there is a significant difference between the probabilities of the two labels. A higher LR value indicates strong support to the positive label and a lower LR value indicates a strong support to the negative label as we saw in case of words like ’fantastic’ and ’boring’.

7

7.1

1. Validation - Doing a train-validation-test split resulted in the same best pseudocount.

2. Z score - Though the classes aren’t skewed, evaluating the model using z-score may be important. For example, IMDB might be more sensitive to getting a wrong negative predictions as it could tamper a movie’s reputation. In such case, precision becomes a better measure than accuracy. Here is the plot of Z score on the pseudo counts sweep. The best pseudocount changes to 17 instead of 7.5 in this case.



3. NLTK word tokenizer (nb_7.py) - Replacing the tokenizer in the code to NLTK tokenizer didn't improve the performance. The best test accuracy was 0.8133 and best Z score was 0.8026
A dump of the output to measure the difference in tokenization -
NUMBER OF TOKENS IN POSITIVE CLASS: 3555096.0
NUMBER OF TOKENS IN NEGATIVE CLASS: 3499421.0
VOCABULARY SIZE: NUMBER OF UNIQUE WORDTYPES IN TRAINING CORPUS: 134725

7.2

A good set of features should be able to capture the generalizable opinion of the reviewer specific to his experience with the movie being reviewed. Here are a couple -
1. (movie element, [adjective list], class)
For example, a feature (*story, [impressive, gripping, beautiful, great,...], positive'*) would be set to 1 for the positive class if the review contains a line (needs appropriate tokenization) with word *story* and one of the words *[impressive, gripping, beautiful, great,..]*..
2. (adjective, class)
A simpler feature could just be a pair like (great, positive) which would be set to 1 for positive class if the review contains the word great.

7.3

For a class, we could consider the data cases from all the other classes as negative examples. The count statistics would only vary for total counts by now counting all classes instead of 2 classes.

7.4

The normalizer doesn't depend on the class. There shouldn't be any change.

7.5

The classify function would do a max of posterior over all the classes instead of two.