CS165A
MP1
Ning Kang
8676173
ningkang@ucsb.edu

**Architecture**

The Bayes program consists of following parts:

1. Data input: Read the data and represent each article as a list of words. With a label attaching in the end. With input as the file, output as list of [[doc,label]]
2. Bayes training: train the Bayes network based on the training data with adjustable parameter. With input as training data, and adjustable parameter, output as the dictionary of score for each word, and a list of author's frequencies. Score is calculated from the prior probability and multiplication of corresponding IDF score.
3. Model testing: make prediction based on the Bayes network and calculate the corresponding accuracy score. With input as the word score dictionary, test data, and author frequency, output as the accuracy score, predictions and actual test labels.

**Preprocessing**

I represent document as a bag of word. Preprocessing is simple and easy, with input as the file, output as list of [[doc,label]]. Doc is a bag of words in a list, doc =[word1, word2,word3]. I treat each word as an individual feature.

**Model Building**

Training a Naïve Bayes classifier is the process of calculating probability of each author, the prior probability and the corresponding IDF weight of each word.

The probability of each author is defined as:

$$P(C_j) = \frac{n_j}{doc_{tot}}$$

With $n_j$ represents the frequency of the author j, $doc_{tot}$ represents the total amount of document in the training set.

Built upon the idea from lecture 6, prior probability is defined as:

$$P(x_k|C_j) = \frac{n_k + \propto}{n + \propto * len(Vocab)}$$

with $P(x_k|c_j)$ as the prior probability, $x_k$ represents the word k, $c_j$ represents author j, $n_k$ represent the frequency of word k in all author j's article, n represent the frequency of word k in all documents, $\alpha$ as the smoothing parameter which is defined as 0.0000001 after series of testing, len(vocab) as the total amount of distinct word. All above information could be calculated by going through the training data for one time.

IDF weight of each word is defined as:

$$IDF(x_k) = \frac{n_{tot}}{n}$$

With n represents n represent the frequency of word k in all documents, $n_{tot}$ represents the total amount of words in all documents. With the lower frequency of the word, the higher IDF score of it.

Considering the underflow prevention and the multiplication with IDF score, prior probability and probability of each author, must be represented positively in the log form. Modified prior probability is defined as:

$$P(x_k|C_j)_{mod} = \frac{-1}{Ln(P(x_k|C_j))}$$

Modified author frequency is defined as:

$$P(C_j)_{mod} = \frac{-1}{Ln(P(C_j))}$$

I also find that, prior probability defined in this way gives too much credit to those authors with more samples in the training data set. For example, author 6 has the greatest amount samples in the training data set and due to the design of prior probability (2nd equation), he will have a lot of unfair advantage in the $n_k$ than other authors. We can't make prediction purely based upon the number of samples in the training data set. There should have a punishment for the length of sample to counteract. After considering this prospect, the final score of a word is defined as:

$$S(x_k|n_j) = (1-\beta) * IDF(x_k) * P(x_k|C_j)_{mod} + \frac{\beta * IDF(x_k) * P(x_k|C_j)_{mod}}{n_j}$$

With $\beta$ defined as an adjustable parameter on how much we should use this punishment. After series of testing $\beta$ is chosen to be 0.9938.

The calculation of prior probability and final score is much smaller than iterate through all the training data. The computation time is equal to:

$$O(doc_{tot} * L_d)$$

With $dod_{tot}$ defined as the total amount of document in the training data, and Ld as the average length of the document.

Dictionaries containing $S(x_k|n_j)$ and $P(C_j)_{mod}$ will be stored and pass on to the next test stage.

**Results**

Using equation below to make prediction, final accuracy on the testing data is equal to 0.778, which is fairly high considering the simplicity of the method. Accuracy on the training data is equal to 0.985. The final training time is equal to 2 seconds and final testing time is equal to 8 seconds which are blazing fast.

$$C_{NB} = argmax_{c_j \in C}[P(C_j)_{mod} + \sum S(x_k|n_j)\,]$$

10 most important feature for each class

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|
| nasty | dismissal | perverse | saluted | offending | appalled | compassionate | advantageous |
| bullets | shaven | wakes | creaking | appealing | inquisitive | identify | persistent |
| clutching | lazily | pas | splash | gasp | lunatic | accompaniment | deposit |
| foolishness | oddly | dismissal | awkwardness | bony | warlike | mournfully | firmer |
| reappeared | contrivance | identify | chronicle | detection | hap | speck | incumbent |
| grinning | creaking | inarticulate | chalk | officials | rambling | incumbent | blunder |
| developing | innate | twain | shaven | madly | reproached | wakes | truer |
| sneered | lunatic | foretold | spin | lunatic | dispositions | truer | stifling |
| twinkle | compassionate | bruised | wills | elated | foretold | merciless | commended |
| knotted | glimmer | shaven | elated | poorly | governing | rippling | dominant |

| 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|
| menacing | monsters | cowardice | smelt | delightfully | rebellious | cheerless |
| click | awkwardness | bolted | splash | brilliantly | weakened | afore |
| dismayed | offending | dazed | awkwardness | sweeps | twinkle | compassionate |
| groaning | sallow | renewal | volunteered | mourn | vl | chalk |
| tasks | reappeared | restoring | tapping | click | scratching | identify |
| cursing | chalk | blunder | expectant | bewildering | suggests | tormented |
| awkwardly | stealthily | clutch | truthful | sap | wills | mane |
| hurts | noises | inarticulate | wins | colossal | offending | amen |
| creaking | elated | completeness | bruised | smells | poisonous | elated |
| nasty | innate | dominant | knotted | poised | lunatic | foretold |

**Challenges**

Main challenges I faced are underflow problem and over trust on sample amount. I used the log version of the prior probability to solve the underflow problem and added a punishment parameter for the author with too many articles in the training dataset.

**Weakness**

The weakness of my method is that it needs large amount of sample to fully understand an author's writing style. On one side, for class 6 which has large number of samples, it could recognize 58 out of 62 in the testing data. On the other side, for class 4, due to lack of training sample, although it could recognize 94/94 in the training data set, only 3 out of 11 was recognized in the testing data size. The overfitting problem hurts a lot when test samples are small. Addition of test samples will definitively help the problem. Also, the addition of more representative feature could help, such as including the position of the word.