Group U3 (Good fans of U2, the band)

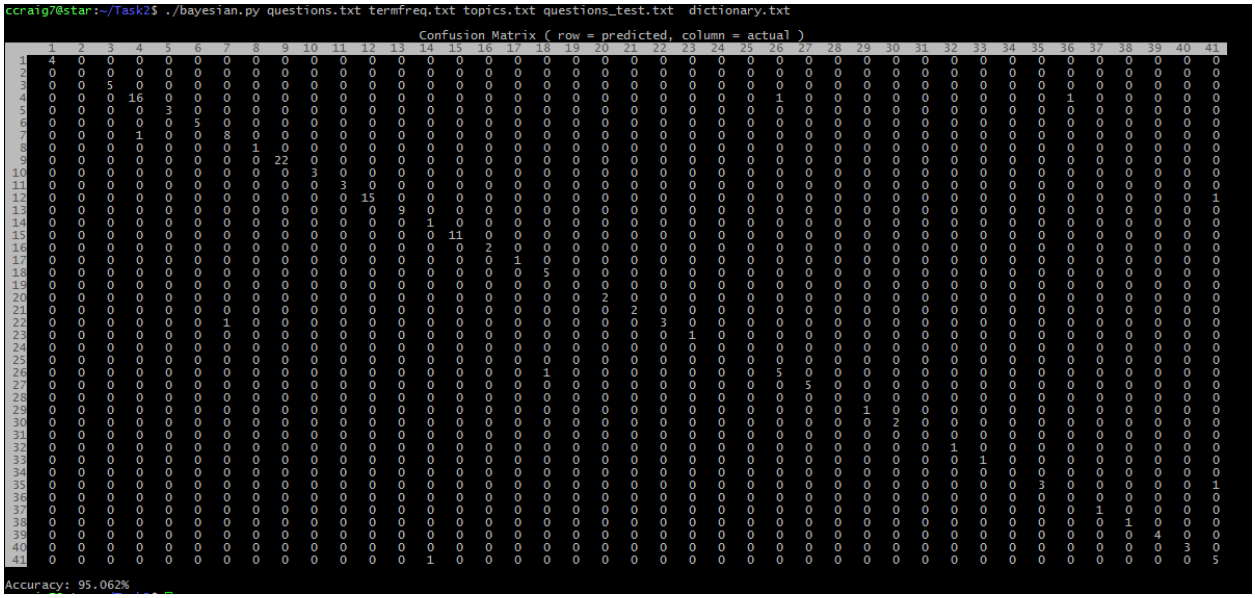
Joshua Strange, Chris Craig, Seth Elliot

Task 2 Documentation

How to run our program

Since our program is python-based no compiling is required. Running the command without any arguments returns our prompt which. Our API uses a preset order to read in data and the dictionary is **required**. The program returns a confusion matrix of the 41 classes and their proper identification and the overall accuracy of the individual test.

Prompt :
`./bayesian <path_to_questions.txt> <path_to_termfreq.dat> <path_to_topics.txt> <path_to_test_questions.txt> <path_to_dictionary.txt>`



Responsibilities

Joshua Strange

- Wrote parsers, after an hour long fight with python. Moved several of these parsers into a single library for ease-of-use.

Chris Craig

- Wrote the termfreq parser. Calculated the initial probabilities and determined the program's algorithm along with Seth Elliot. Ran results and wrote documentation.

Seth Elliot

- Broke apart the calculated data to generate the confusion matrix. He also built the majority of the interface for the Naïve Bayesian Classifier along with Chris Craig. Wrote the last-minute hack to account for a split in training data.

Group U3 (Good fans of U2, the band)

Joshua Strange, Chris Craig, Seth Elliot

Methods & Results

Naïve Bayesian Classification:

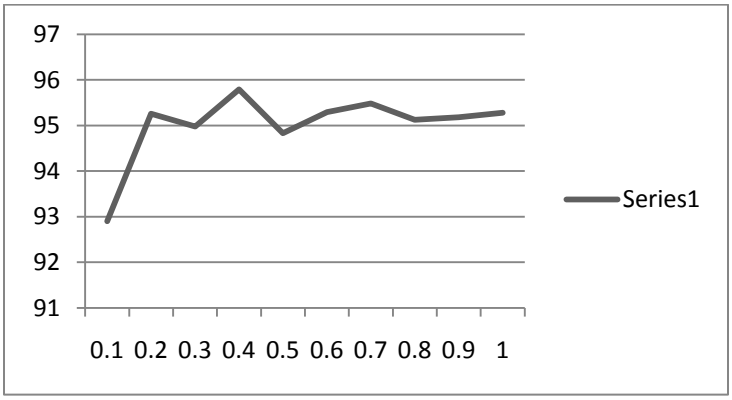
This was our primary method of classification and the process itself is rather straightforward. Using the following equation, we parsed the necessary data from the given text files, to get a classifier that has an average accuracy of 95%. The confusion matrix can be seen below.

$$P(Y|X) = \frac{P(Y)\prod_{i=1}^{d} P(Xi\,|Y)}{P(X)}$$

Figure 1

Brief translation: To get the probability of a topic given a question, we must multiply the probability of the topic in the training questions by the probability of the term in the topic. Since the probability of the individual question would have to be included in each factor we removed it in order to maximize the product and determine the best category for our evaluation.

To evaluate our classifier, we ran it through several thresholds of test data and took the average of 50 random collections of test-data using a bash script. The results are below.



Threshold	Accuracy
.1	92.900
.2	95.255
.3	94.980
.4	95.792
.5	94.830
.6	95.294
.7	95.484
.8	95.121
.9	95.178
1	95.280

But we encountered an error when we discovered how the classifier will be evaluated. Since we are receiving an incomplete set of training data when evaluating, certain terms will not be found in our term-frequency matrix. Initial handles for this term were zeroed out and ignored, but doing so distorts our probabilities. So, we agreed to ignore values but preserve the data we did find.

Conclusion

What does it mean? After classifying this data we can determine with 95% accuracy that key terms in a question can determine their class. This 95% accuracy was achieved by having a complete set of training data. When certain questions are parsed from the training set, our accuracy drops to 5-30%. Because certain values only contain one question and if it is removed from the training set, there is no way to properly classify the data. Another observation was the weight of certain words within the data. We determined that key words like, Fulmer or Tennessee tended to push the classification in the particular data.