Stat 154 Final Project: Team 11 (R^4)

Steven Yang Hu (23770857), Keun Gang Kyen (23983641), Vi Le (25109634), Yueyilin Qi (23514031)

December 7, 2015

1. Description

The data for this project comes in 22,308 text files from Project Gutenberg which contains text of books from one of four categories: Child, History, Religion, or Science. One book may be in multiple text files and there are a total of 7,164 text files classified as Children, 5352 text files classified as History, 2361 text files classified as Religion, and 7431 text files classified as Science. Thus, Science has the most files and religion has the least, the range on the number of files per category is about 5000 files. In the Childen category, texts tend to have simple vocabulary and pictures (which in the text files is made using punctuation) so that they can understand and enjoy the reading. On the other hand, books in other categories use more complicated vocabulary. Books in the History category mention a lot of events or famous people from the past, while books in Religion category contain the terms related to religious figures or the different types of religions a lot. Science books use a lot of scientific terms in all types of categories in cluding Math, Chemistry, and Biology. Overall, it is fairly easy for humans to classify the genre of the text files (this is possibly based on vocabulary level, our understanding of each of these genres, etc.) This report will investigate how well statistical classification methods can classify text files into these genres.

2. Word Feature Matrix Creation

A word feature matrix was created by using Python. Preliminary, there was a total of around 74,000 words among the 21,705 text files after test files that only contained Project Gutenberg license information were removed. All words that appeared a total of less than 10 times were not considered since they are too rare compared with the maximum number of appearances which is 788,100 times. Words that rare are unlikely to have much power to differentiate between classes, as they will likely only appear in one specific book, and are not a common trait of the class. Figure 1 below displays the distribution of total times a word appears throughout all the texts against how frequent that total appears before anything was done to decrease the total number of words. A log-based scale was used to better present the data. From the figure, it is evident that on a log scale the words that appear in less than 10 documents is much more common than those that appear in more than 10 documents. Thus, to ease computation this threshold was selected as the minimum for words to be kept before moving on to unsupervised feature filtering. The most common English words were removed (total of 119 words) to improve our prediction since those words appear in all genres. A word stemmer package (nltk.stem.wordnet) was used to capture all words from the same root word into one category so that we can minimize our number of features and focus on the meaning of each root word. Lastly, numbers and roman numerals were removed from consideration since this step purely creates a word matrix, not numbers. This decreased the number of words to 53,810. Since the matrix is sparse, packages handling sparse matrix operations were used in both Python and R to aid in computability.Thus, the final matrix dimensions after this step was 21,705 by 53,810.

3. Unsupervised Feature Filtering

From the originally generated word matrix, unsupervised learning was used to cut the number of word features down to around 2,000 features. For many models (excluding tree-based models) the features (in this case words) that only have a single unique value may cause the model to crash or the fit to be unstable. In addition, a predictor may only have a handful of unique values that occur with very low frequencies. These predictors may become zero-variance predictors when the data are split into cross-validation or bootstrap

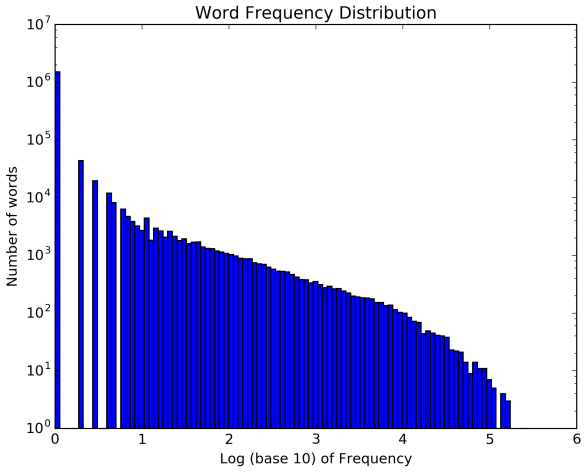

Figure 1: Word Frequency Distribution (log base)

sub-samples and a few samples may have an undue influence on the model. These "near-zero-variance" predictors may need to be identified and eliminated prior to modeling.

To account for this issue, three steps were performed in unsupervised feature filtering. First, the top 100 most frequent words (by number of documents they appear in) and all words apprearing in less than 10 documents were cut. The most frequent words were cut through ranking all the frequencies and looking at which words appeared the most. The cut-off occured when the words stopped looking like common words. After this step, the number of columns in the word matrix was cut down from 53,810 to 40,622.

Next, we sought to remove features with low variance, because low variance could indicate that a feature doesn't change much between classes. However, since explicitly calcuating the variances of every feature was computationally prohibitive, we used another method to remove low-variance features. First, the "percent of unique values" was calculated for each feature and features with a value less than a threshold were eliminated. The "percent of unique values" is the number of unique values divided by the total number of samples that approaches zero as the granularity of the data increases.When the unique value percentage is less than some threshold, we might consider a predictor to be near zero-variance. The range of "percent of unique values" was 0.0092 to 1.027. Thus, a threshold of 0.095 was chosen in order to cut the number of word features down to around 2,000 features.The word matrix then had 2,043 columns.

The third step was to decrase the correlation between predictors. Pairwise correlation between predictors were computed. In each pair of features with correlation above some threshold, the predictor that is more correlated with all other variables was eliminated. Highly correlated predictors can make choosing the correct predictors to include more difficult and interfere with determining the precise effect of each predictor, yet having correlated predictors doesn't affect the overall fit of the model or produce bad predictions. However, because of the difficulty in choosing an optimal model when high correlation is present, it is worth exploring. A threshold of 0.7 was chosen because 0.6 would eliminate too many predictors and 0.8 is considerably too high to remain in the word feature matrix. After all features with a correlation of 0.7 or higher were elimiated, the dimensions of the final word feature matrix was 21,705 by 1,965.

4. Power Feature Extraction

Another feature matrix was created to represent power features. Features were chosen to compliment the word features. The matrix has each text file as a row and the different power features as columns. The sets of columns are:

1. *Word Length*: The total number of words in each document are classified into short (1-5 letters), medium(6-10 letters), and long words (11 or more letters). The frequency of each type is recorded in the matrix. The word-length cut-offs for each category were chosen. This metric may be used to distinguish any difference in word length distribution between genres.
2. *Frequency of 4-digit Numbers*: The total number of 4-digit numbers that appear in each document. This frequency was primarily created to capture years, although there will be some other numbers that may be included as well. The presence of years may be different between, say, children's texts and history texts.
3. *Frequency of Quotation Marks*: The number of quotation marks that appear in a text are recorded in this column. This frequency is used to detect the presence of dialogue within a text. It is suspected that the amount of dialogue present in Children's text is different from the others.
4. *Frequency of Question Marks*: The number of question marks that appear in a file. This feature was made to detect the presence of questions in a text. It can be suspected that some genres have more questions than others. For example, Science texts may ask more questions than Children's books. 5.*Bigrams*: two words that appear together (at least 200 times overall in our case) that have a meaning different from if they appear separately. The list of all two-word pairs was initially generated by ranking all two-word pairs by point-mutual information. A list of the top 200 word pairs was produced. This measure takes two words (X and Y) and computes a probability based on the following formula:

   $$logp(x, y)/p(x)p(y)$$

   where x and y represent the two words in the pair and $p(x, y)$ and $p(x), p(y)$ represent their joint and individual distributions, respectively. Thus this metric calculates a ratio of how many times the two words appear together over how many times they appear separately. The list was then manually filtered to remove undescriptive pairs (which are defined as pairs we know are common to Project Gutenberg, proper nouns such as characters in a book, non-word pairs, etc.).

The final dimensions of the power feature matrix was 21,705 by 60.

5. Word and Power Feature Combination

The word and power feature matrices were combined by concatenating the columns together. The final dimensions are 21,705 by 2,025.

6. Classification on the Filtered Word Feature Matrix

Random Forest and Support Vector Machine (SVM) algorithms were used to create classification models via validation set and 10-fold cross-validation on the 21,705 by 1,965 word matrix.

Random Forest on Word Feature Matrix For Random Forest, three different numbers of trees were ran with 10-fold cross validation, namely 100, 500, and 748. (748 was chosen to aid in parallel processing, as it is divisible by 4.) With each number of trees, five values of m (features to consider at each split) were used. These values were 10, 45, 60, 80 and 100. Figure 2 below shows the average error rate results.

From the figure, it is evident that going from 500 to 748 trees didn't decrease the error as much compared to the decrease in error from 100 to 500 trees for all values of m. Thus, for ease of computation time 500 trees was chosen as the optimal number of trees to grow for Random Forest on the Word Matrix. An m value of 80 was chosen (also to ease computing time) because further increases in M seem to have diminishing returns, and will increase computational time required. Table 1 gives the overall and by-class error rate results for 500 trees.

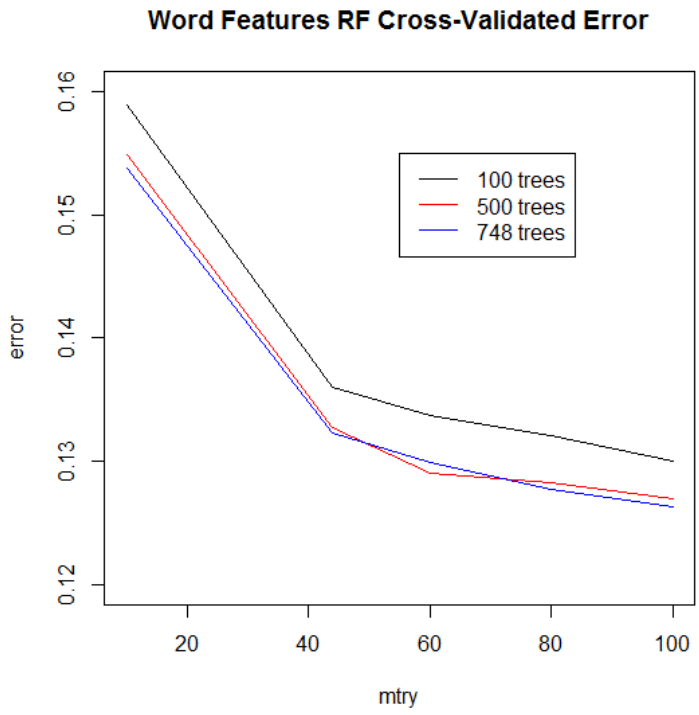

Figure 2: Random Forest CV Results on Word Feature Matrix

	mtry=10	45	60	80	100
Overall Error	0.1548495	0.1327811	0.1290492	0.1283122	0.1269297
Class 0 Error	0.1287497	0.1320676	0.1296540	0.1326577	0.1346818
Class 1 Error	0.2385739	0.1762993	0.1713612	0.1630772	0.1580223
Class 2 Error	0.2293407	0.1867453	0.1825585	0.1831792	0.1753074
Class 3 Error	0.0929032	0.0823764	0.0782344	0.0791766	0.0790551

Table 1: Average Error per mtry for 500 trees on Word Feature Matrix

It is evident from Table 1 that there are some discrepancies between the classes, but none of them perform exceptionally poorly. The best class classification is for class 3 and the worst is for class 2. The range in errors by class is around 0.1. The final Random Forest model on the word matrix was built using 500 trees with $m = 80$, in order to achieve good results while reducing complexity.

Linear SVM on Word Feature Matrix The word feature matrix was then used to create a linear SVM model. A linear model was used to avoid possible unneccesary complexity since the number of features is large. Furthermore, linear kernels generally perform well on text classification tasks. Before running the SVM, we scaled the data in each column by centering on zero and dividing by the standard deviation.

In order to tune the cost parameter, first, a validation set approach was used for linear SVM to find a target range for cost and then 10-fold CV was done on values in a smaller range correspond to those costs in the validation set with the lowest error. Using a validation set to narrow the range of costs allowed us to save time when cross-validating to tune the cost parameter more finely. Initially, the values 0.0001, 0.001, 0.01, 0.1, and 1 were used and errors were calculated on a test set.

Figure 3 displays the results.

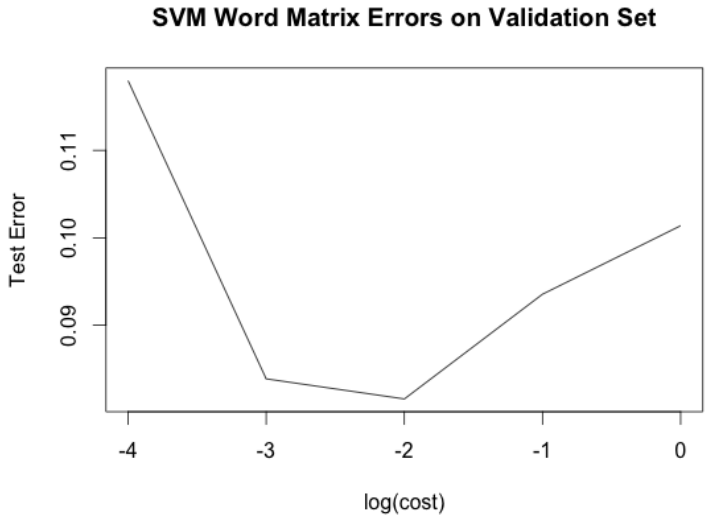

Figure 3: SVM Validation Set Results on Word Feature Matrix

It is evident from Figure 3 that the best cost parameter is 0.01 because it has the lowest test error. The graph at first decreases and then increases, indicating a trend that values higher than 1 and lower than 0.0001 would give higher errors. Since these values were only done on a validation set, the values 0.001, 0.005, 0.01, 0.05, and 0.1 were used in the next step (10-fold CV) to ensure a good range was captured.

Table 2 gives the overall and by-class error rates for each cost parameter value.

The 10-fold CV on the stated cost parameter values produces Figure 4 below.

	0.001	0.005	0.01	0.05	0.1
Overall Error	0.0840369	0.0797981	0.0822859	0.0880452	0.0912241
Class 0 Error	0.1055345	0.0966488	0.0970532	0.1001912	0.1061255
Class 1 Error	0.1224902	0.1191674	0.1240538	0.1296560	0.1301528
Class 2 Error	0.1070282	0.0910036	0.0941223	0.0938548	0.0966151
Class 3 Error	0.0251819	0.0289752	0.0317224	0.0425236	0.0451550

Table 2: Average Overall and by-Class Error for Each Cost Parameter from 10-fold CV on Word Matrix

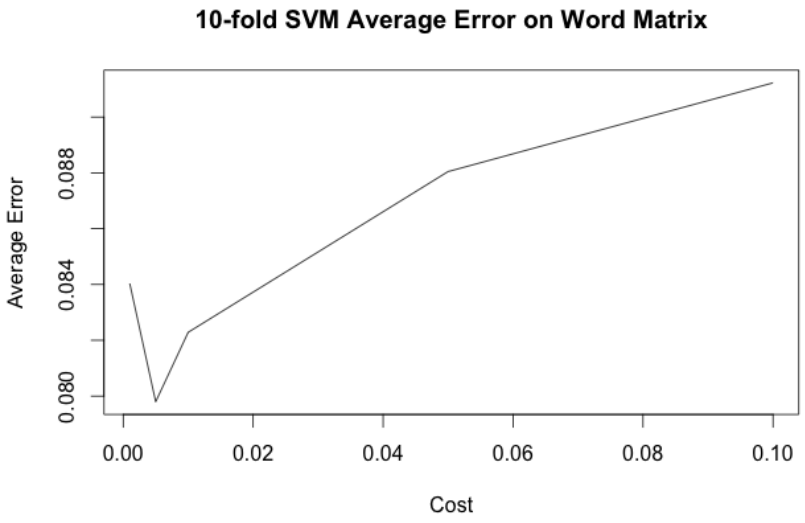

Figure 4: SVM CV Results on Word Feature Matrix

The average error across ten folds decreases from a cost of 0.001 to 0.005 and then increases afterwards. The by-class error rates from Table 2 support this conclusion since the average error decreases from a cost of 0.001 to 0.005 and then increases afterwards as well for every class. Thus, the cost parameter for the word matrix was chosen as 0.005. It is interesting to note that this classifier had the best classification rates for classifying texts of class 3 and the worst classification rates for calculating texts to class 1 over all cost parameter values. Lastly, a full linear SVM model on the word feature matrix was run with the cost of 0.005. The cross-validated error rates achieved by the SVM are substantially better than those achieved by the random forest model.

7. In-Class Competition

[This step was performed in class.]

8. Classification on the Power Feature Matrix

Random Forest and linear SVM models were also fit on the power feature matrix. 5-fold and 10-fold cross validations as well as validation set approach were used.

Random Forest on Power Feature Matrix To determine m and the number of trees for the power feature matrix, 5-fold CV (due to computation time) was used on 100,500 and 748 trees with $m = 4, 7, 10, 13, 16$. Figure 5 below shows the results.

Similar to the word feature matrix, the drop in error is small from 500 to 748 for all values of m compared to the drop in error from 10 to 500 trees. Thus, in the interest of computing time 500 trees was used on the final model. $m = 13$ was chosen as the best value for m since the error decreases for all numbers of trees before $m = 13$ and increases after it for 100 and 748 trees. Although the error at $m = 16$ is lower than at $m = 13$ for 500 trees, overall $m = 16$ increases in error so choosing it even for a Random Forest on 500 trees may not be safe. Table 3 below shows the overall and per-class error rates for all m values using 500 trees found using 10-fold CV.

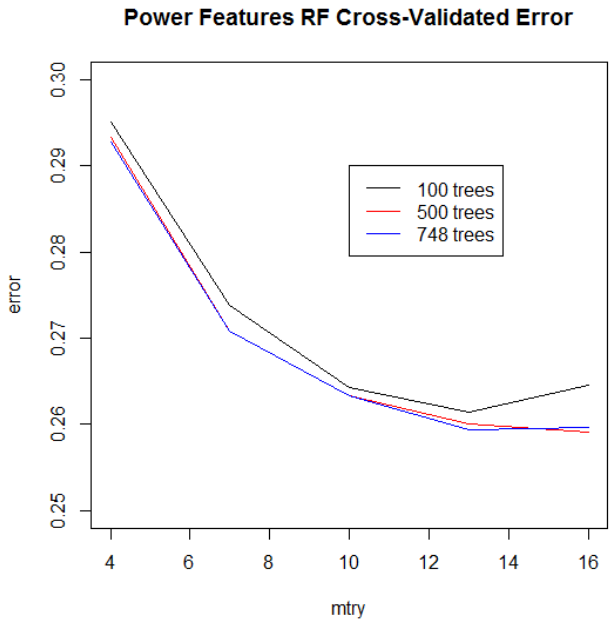

Figure 5: Random Forest CV Results on Power Feature Matrix

	mtry=4	7	10	13	16
Overall Error	0.2932965	0.2708592	0.2633495	0.2600323	0.2591569
Class 0 Error	0.2033786	0.2031020	0.2029353	0.2044597	0.2047143
Class 1 Error	0.4032308	0.3679414	0.3517976	0.3453873	0.3413896
Class 2 Error	0.4808988	0.3982312	0.3758532	0.3635130	0.3619191
Class 3 Error	0.2391329	0.2237621	0.2201900	0.2172712	0.2178926

Table 3: Average Error per mtry for 500 trees on Power Feature Matrix

From the table, the lowest per-class error rates belong to classifying class 0 and the highest per-class error rates belong to classifying class 2. This is different from the Random Forest Model on the word feature matrix since the lowest and highest by-class error rates were class 3 and class 2, resectively. A full model on all the training data was used to create the final Random Forest model for just the power feature matrix using these parameters.

Linear SVM on Power Feature Matrix Before running the SVM, we scaled the data in each column by centering on zero and dividing by the standard deviation. In linear SVM, a validation set approach on costs of 0.01, 0.05, 0.1, 0.5, 1, 5, and 10 were used to find a range with low test errors. The results are in Figure 6 below.

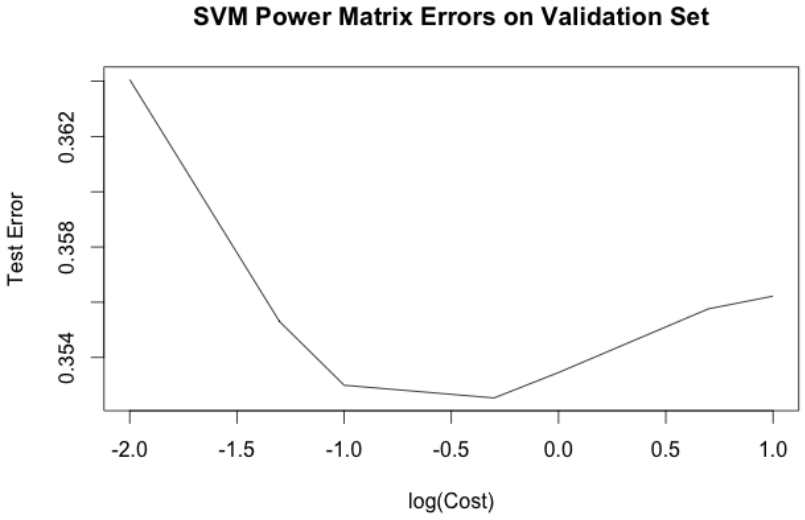

Figure 6: SVM Validation Set Results on Power Feature Matrix

Similar to the previous validation set graph, the test errors first decrease then increase as the cost parameter goes up. The minimum test error occured at a cost of 0.5, so a range around 0.5 was chosen for cross validation.

Next, 10-fold CV was run on the values 0.05, 0.1, 0.5, 1, and 5 and the average errors over 10 folds for each cost parameter are depicted in Figure 7. Table 4 below details the error rates overall and by-class.

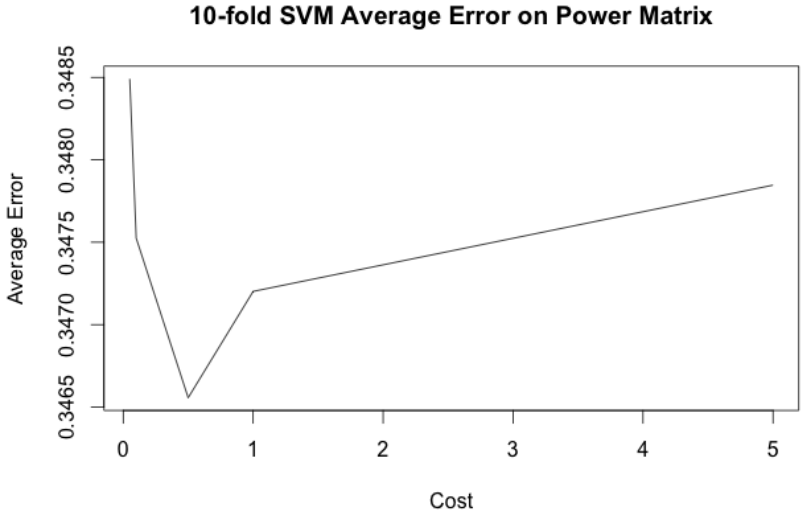

Figure 7: SVM CV Results on Power Feature Matrix

	0.05	0.1	0.5	1	5
Overall Error	0.3484911	0.3475240	0.3465568	0.3472018	0.3478467
Class 0 Error	0.2252988	0.2276413	0.2283003	0.2283591	0.2283915
Class 1 Error	0.5217967	0.5202636	0.5194109	0.5210707	0.5260437
Class 2 Error	0.6169457	0.6145029	0.6086376	0.6082307	0.6114578
Class 3 Error	0.2526819	0.2493445	0.2484821	0.2491456	0.2462620

Table 4: Average Overall and by-Class Error for Each Cost Parameter from 10-fold CV on Power Matrix

The cost value in Figure 6 with the lowest average test error was 0.5 (the same as the validation set approach conclusion.) The graph also behaves in the signature decrease-then-increase fashion. From the table, the best classification error is for class 0 and the worst is for class 1. Unlike the SVM for the word matrix where the difference in error between the best and worst classes was around 0.1, the difference this time is around 0.3. In addition, for the classes 0 and 3 the best error was not for a cost of 0.5. The final linear SVM model on the power matrix was run on the full training set with a cost parameter of 0.5. Unlike the models run on word features, out of the two power matrix models, the random forest has lower cross-validated error.

9. Classification on the Combined Feature Matrix

Random Forest and linear SVM models were run for the combined word and feature matrices. This section details how the parameters were chosen for each model through cross validation.

Random Forest on Combined Feature Matrix For the Random Forest model, 5-fold cross validation was used to determine the optimal m value and the number of trees to build. The values $m = 20, 45, 70, 95, 120$ were cross-validated for 100, 500, and 748 trees each. The results of the CV's is given in Figure 8.

Once again, the drop in error over all m values from 100 to 500 trees is significantly greater than from 500 to 748 trees. In light of this, 500 trees was used for the final Random Forest model to aid in computation time. The decrease in error rate as m increases potentially indicates that the optimal number of trees was not yet reached. Still, the improvements to error as m increases are clearly diminishing. Thus, the final Random Forest model was built using the lowest error from CV, namely 120 trees, to provide good results while maintaining computational speed. Table 5 provides overall and class-specific error for all m values with 500 trees.

	mtry=20	45	70	95	120
Overall Error	0.1466022	0.1356830	0.1276664	0.1250864	0.1231513
Class 0 Error	0.2033786	0.2031020	0.2029353	0.2044597	0.2047143
Class 1 Error	0.2122810	0.1805458	0.1625652	0.1601256	0.1536058
Class 2 Error	0.2162537	0.1973167	0.1746869	0.1722539	0.1657110
Class 3 Error	0.0889684	0.0833839	0.0802107	0.0770796	0.0770849

Table 5: Average Error per mtry for 500 trees on Combined Feature Matrix

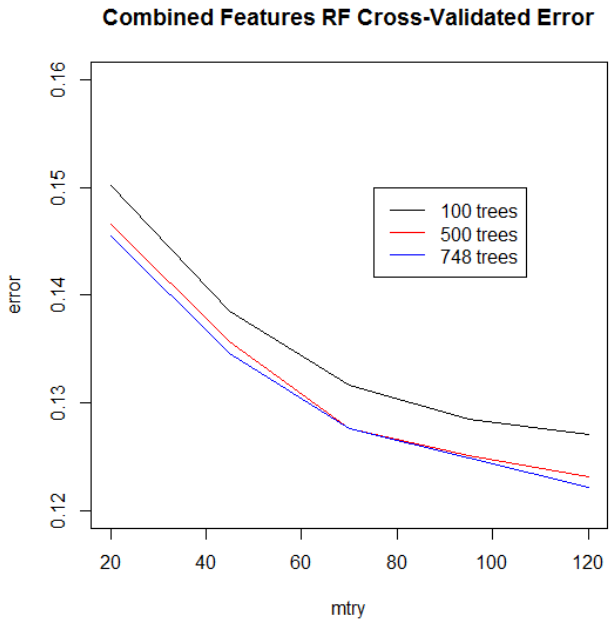

Figure 8: Random Forest CV Results on Combined Feature Matrix

The best by-class error rate was for class 3 and the worst was for class 0. Interestingly, class 0 and class 3 error rates for the best m value found are not the lowest out of all the possible m values tried. The difference in error between the best and worst class classifications was 0.13, which is lower than that of the error rate difference for the power matrix.

The final combined power feature matrix was trained on 500 trees with $m = 120$. Its performance in terms of cross-validated error is slightly better than the word feature random forest.

Analysis of Random Forest Predictions for all Matrices The improvement in cross-validated error from the selected word feature random forest model to combined feature random forest model is rather small. However,from the performance of the power feature random forest, it is evident that the selected power features must have some predictive value on this dataset, as we do achieve accuracy in the mid-70% range. It is possible that combining the two adds little new information. This is described in the following table.

	Word	Power	Combined
Overall Error	0.1283122	0.2600323	0.1231513
Class 0 Error	0.1326577	0.2044597	0.1316938
Class 1 Error	0.1630772	0.3453873	0.1536058
Class 2 Error	0.1831792	0.3635130	0.1657110
Class 3 Error	0.0791766	0.2172712	0.0770849

ROC Curves for Random Forest Models ROC curves were produced for Random Forest models trained on each type of matrix. The predicted values came from predicting the entire set of training data on the final models obtained from training the word, power, and combined matrices on the CV-validated best parameters. Figure 9 gives the resulting ROCs of one-vs-all for each class.

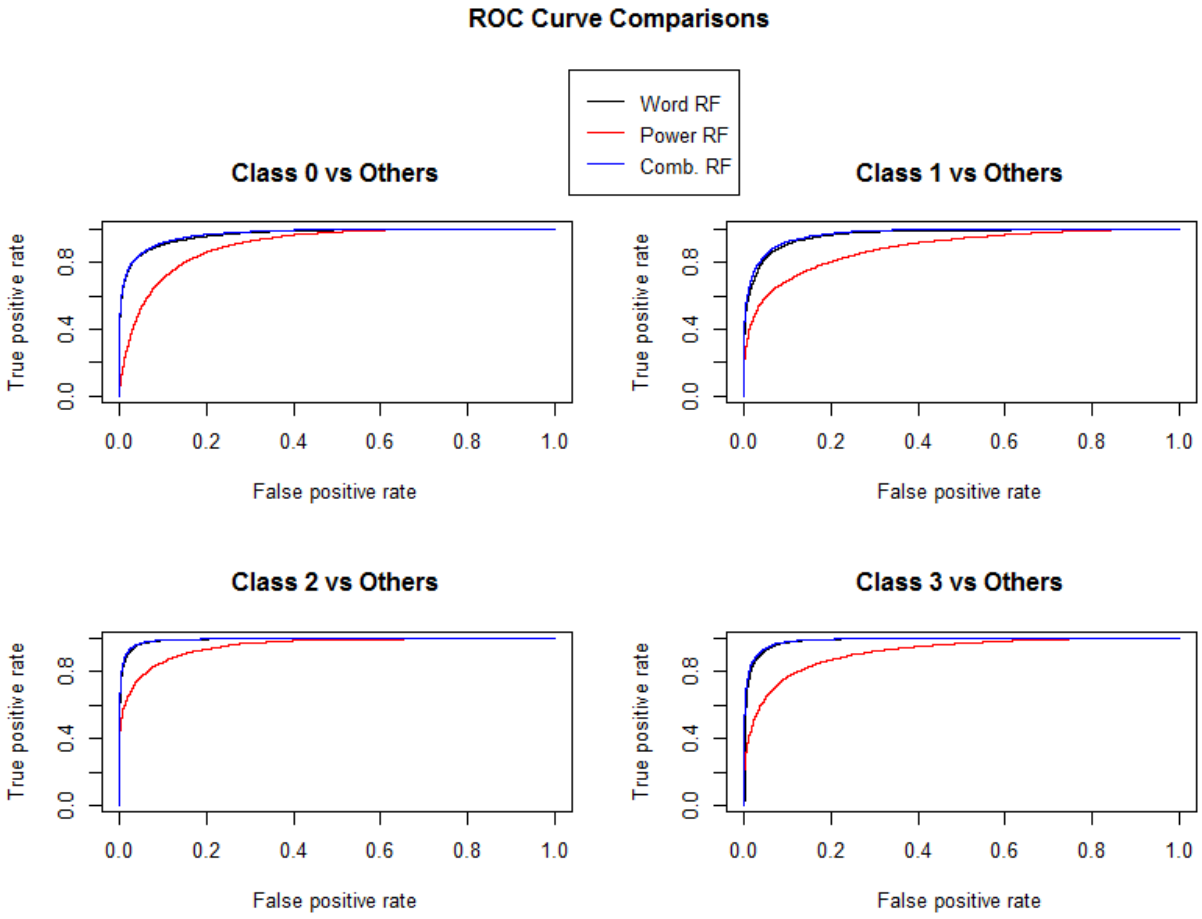

Figure 9: ROC Curves For Random Forest Models by Class

From this figure, it is evident that the area under the curve (AUC) differs by class as well as model. For all single class versus all other classes the power matrix Random Forest model performed the worst and the word and combined matrices had mostly the same performance and was always better than the power matrix model. Overall, the AUC for classes 2 and 3 was greater than that for classes 0 and 1. This implies all Random Forest models were generally better at correctly predicting texts of classes 2 and 3 than 0 and 1. The fact that the word and combined models are so close seems to indicate that combining the power features with the word features in fact does not add much new information.

Linear SVM on Combined Feature Matrix Before running the SVM, we scaled the data in each column by centering on zero and dividing by the standard deviation. For tuning the cost parameter of the linear SVM model, cost values of 0.0001, 0.001, 0.01, 0.1, 1, 10, and 100 were initially chosen and 10-fold cross validation was conducted on each cost parameter. The results of the initial CV are given in Figure 10 below. Table 7 below details the errors for each cost parameter.

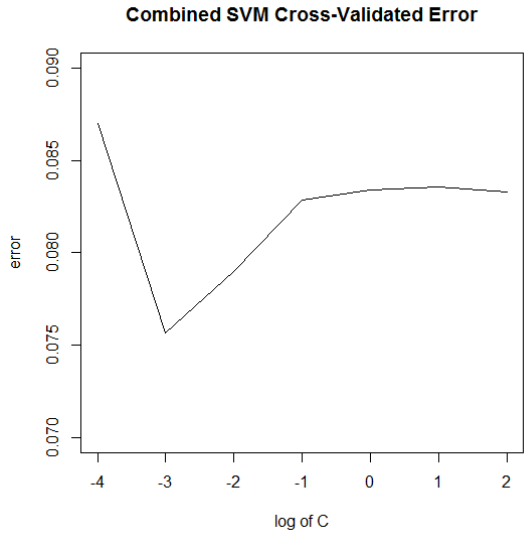

Figure 10: SVM CV Results First Round on Combined Feature Matrix

	cost=0.0001	0.001	0.01	0.1	1	10	100
Overall Error	0.0869848	0.0756507	0.0789675	0.0828377	0.0833907	0.0835750	0.0832986
Class 0 Error	0.1072789	0.0921645	0.0893846	0.0939996	0.0952516	0.0936073	0.0935740
Class 1 Error	0.1297604	0.1100392	0.1154596	0.1201358	0.1216122	0.1219527	0.1217962
Class 2 Error	0.1116501	0.0930523	0.0893296	0.0883140	0.0854702	0.0883504	0.0867132
Class 3 Error	0.0256945	0.0268608	0.0370573	0.0411151	0.0413794	0.0424228	0.0422638

Table 7: Average Error per Cost Parameter on Combined Feature Matrix

From Figure 8, the error curve first decreases, then increases, and eventually levels off. Still, the lowest average error was below the error in the level-error range. The lowest average test error occured at 0.001. From table 7, it is evident that the best classification was for class 3 and worst was for class 1, just like that of the SVM for the word feature matrix. And also similar to the word feature matrix, the errors at a cost of 0.001 is not the lowest for classes 1 and 3. Another round of 10-fold CV was conducted on values around 0.001, namely 0.00025, 0.0005, 0.001, 0.002, 0.004, and 0.008. Figure 11 shows the results of these CV's.

The lowest average error was from a cost of 0.002. This graph also follows a decreasing-then-increasing pattern. The final SVM model on the combined matrix was created from all the training data with a cost of 0.002. This model had the lowest overall cross-validated error of all models we tried.

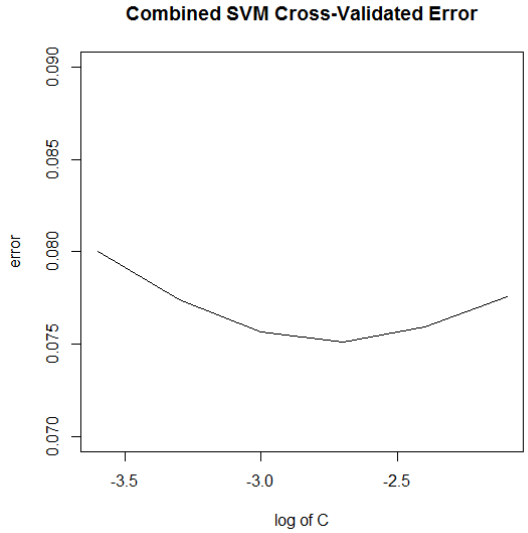

Figure 11: SVM CV Results Second Round on Combined Feature Matrix

	Word	Power	Combined
Overall Error	0.0797981	0.3465568	0.0756507
Class 0 Error	0.0966488	0.2283003	0.0921645
Class 1 Error	0.1191674	0.5194109	0.1100392
Class 2 Error	0.0910036	0.6086376	0.0930523
Class 3 Error	0.0289752	0.2484821	0.0268608

Analysis of SVM Predictions for all Matrices Please refer to the table above. Like our random forest models, our combined SVM model has the lowest cross-validated error, but only slightly lower than the word feature SVM. This may indicate a problem with our choice of power features, as they seem to provide little new information that improves predictive accuracy.

10. Validation Set

Predictions were submitted from a total of four models to Kaggle: the word and combined random forests, and the word and combined SVMs. What is immediately evident is that the combined models performed significantly worse than the word models for both random forests and SVMs. Out of the four attempts, the word SVM performed the best, with an accuracy of .9317.

Word SVM	Combined SVM	Word Random Forest	Combined Random Forest
0.9317	0.91712	0.87963	0.85256

Table 9: Kaggle scores for Subset of Final Models

This problem may be explained by flawed power features. Although due to time limitations, it was not possible try different sets of power features and further investigate the problems, it is hypothesized that our choice of power features, particularly in the bigrams (as they are selected based on frequency in the training set, which may not be representative of an unseen validation set), may have been too specifically tuned to the training dataset. As a result, it is possible that the combined models achieved their slightly better cross-validated error due to overfitting, and therefore performed poorly on the final validation set when a completely new corpus came into play.

In general, it seems that SVMs perform better for this text classification task than random forests. Therefore our word feature SVM was selected as the final model, achieving a Kaggle score of 0.9317. From cross-validation, the lowest error in was in class 3 predictions, at 0.045, while the highest error was in class 1, at 0.130. This does seem reasonable, as the vocabulary of science books is probably the most distinct out of all the classes, with words that rarely appear in any other genre/context. The other three classes, particularly history and religion, likely share more of their vocabulary.