Predicting Congressional Bill Outcomes

Barry Chen, Vivian Hu, David Wu

March 15, 2015

1 Introduction

How can we predict the outcome of legislation in Congress? Will it pass the committees, subcommittees, and be voted upon by both the House and Senate? The large variation in bill content makes prediction based on legislative text a challenge. We propose to use the Latent Dirichlet Allocation (LDA) topic model to extract features from legislative text and a Random Forest binary classifier to label each bill as "pass" or "fail."

Our general approach uses a collection of bills to generate the LDA topic model. With the topic distributions generated by our LDA model as features in a Random Forest, we can classify new bills into pass and fail groups.

We perform 5-fold cross validation on three hyperparameters governing our LDA model as well as our classifier in section 6. In addition, we compare our classification results with three baselines, and discuss variations in our data set composition in section 7.

2 Previous Work

Previous work in the area of congressional prediction has looked at legislator's voting habits or has used bill meta-data in predicting its committee outcome. Gerrish and Blei [1] uses LDA to extract topics from congressional bills, and applies the topics in the Ideal Point Model to predict individual legislator's voting habits. Yano, Smith, and Wilkerson [2] focus on a bill's survival through the congressional subcommittee process, based on bill meta-data and a data set of bills and their labeled categories.

Our work uses the LDA topic model, like Gerrish and Blei, but looks at bill survival throughout the congressional process, independent of subcommittees and legislators.

3 Data Sets

For this project, we use the comprehensive data complied at `govtrack.us`. Complete congressional bill text and voting results can be found for congresses 103 through 113 [3].

3.1 Pre-processing

After complete congressional information (meta-data and text) has been downloaded via `rsync`, we use Govtrack's API to retrieve congressional actions for every bill. These actions are parsed to determine if a bill has passed or failed. Finally, we partition the data into two classes: pass and fail.

For each document, we find the latest version of its text, removing trivial stop-words and non-alphanumeric characters. From there, we built the vocabulary by lemmatizing the words and assigning each word a unique ID, resulting in a "bag-of-words" whose size is the number of word IDs.

For this project, we consider a bill "passed" if it is explicitly passed in both the House and the Senate and is enrolled (passed to the President for a final signature). We say a bill has "failed" if it has been explicitly voted down, or it has not made it to the voting floor (by either not passing through the committee step or being ignored). Definitions differing from the above will be explored in section 7.

Total number of bills:	16,810
Number of bills in training:	10,669
Number of bills in testing:	6,141
Pass to Fail Ratio:	34% Pass : 66% Fail

3.2 Statistics

The composition of our data set can be seen below.

4 Latent Dirichlet Allocation

Latent Dirichlet Allocation (LDA) is a topic model that we use to extract topics from our collection of bills. LDA represents documents (in our case, bills) as a mix of topics, with each topic being a distribution of word probabilities for that topic [4]. Using Collapsed Gibbs sampling, we randomly assign words to topics and then iteratively improve those assignments, as explained in the next subsection [5].

Our LDA model takes as input the number of topics we want to find, the number of iterations of Gibbs sampling, and the set of bills with its associated vocabulary. We receive as output an assignment for each word in a document to a topic with some probability.

From this, we can generate a word distribution for each topic, which represents the probability that each word appears in a certain topic. We can also generate a topic distribution for all bills, which represents the probability, for each bill, that the words in the bill can be categorized into each topic. Using the word distributions for each topic, we can also intuit names for each topic based on the top words represented in that topic. It should be noted, however, that topics which may not make intuitive sense to humans may still capture some latent trends in the set of bills.

We show in Table 1 the top eight words for 5 topics generated from 20 iterations of Gibbs Sampling on the data set specified in section 3.2.

Topic 1	Topic 2	Topic 3	Topic 4	Topic 5
deleted	***united***	united	year	year
program	national	person	***health***	amended
fund	government	***commission***	***program***	***amount***
public	committee	***amended***	***child***	***inserting***
federal	***defense***	term	***individual***	***tax***
project	member	***federal***	care	***plan***
fiscal	title	action	title	***code***
grant	***force***	information	fiscal	credit

Table 1: Top eight words for each topic. Intuitive words italicized.

4.1 Gibbs Sampling

Starting from a random assignment of words to topics, Collapsed Gibbs Sampling will repeatedly improve this assignment for a fixed number of iterations. A word is assigned a topic by assuming that all the other words have been assigned correctly. More precisely, a word is assigned to a topic based on a multinomial distribution given by equation 7 in [5]. The equation shows that the selected topic for a given word in a document is based on the prevalence of that topic in the document and the topics assigned to other occurrences of the word.

4.2 Features

The features of each bill are given by the proportion of words in the bill corresponding to each topic. More specifically, a bill consisting of words $(w_1, w_2, ..., w_n)$ is represented as a feature vector, X, of length K, where:

$$X_k = P(topic = k|w_1, w_2, ..., w_n) = \frac{(\prod_{i=1}^{n} P(w_i|topic = k))P(topic = k)}{P(w_1, w_2, ..., w_n)}$$

Since the denominator is a constant for a given bill, we can compute the numerator and normalize the features to sum to 1. The terms in the numerator can be calculated in the following way:

$$P(w_i|topic = k) = \frac{\text{frequency of word} w_i}{\text{frequency of words assigned to topic k}}$$

$$P(topic = k) = \frac{\text{frequency of words assigned to topic k}}{\text{total number of words}}$$

If a test document contains a word that is not assigned a topic by LDA (i.e. the word is not contained in any of the training documents), then the word is skipped.

4.3 LDA Results

Perplexity is a measurement of how well the probability model assigns words to topics. The perplexity is given by equation 11 in [5]. After each iteration of Sampling, the assignment of words to topics should improve, and the perplexity should decrease, as shown below for 20 topics and 20 iterations of Gibbs Sampling on the data set specified in section 3.2.

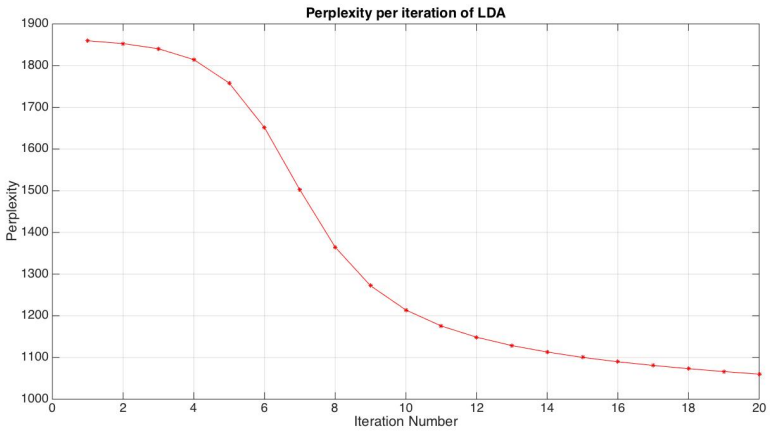


Figure 1: Perplexity for 20 iterations of Gibbs Sampling

The following figures show the word distributions for five topics generated from our data set. We only show the word distribution for the top 50 words overall over all topics. We also show the overall topic distribution for 50 randomly chosen bills in figure 7. We can see that topic 2 (in light blue) is a very general topic that is found with a high probability in many documents.

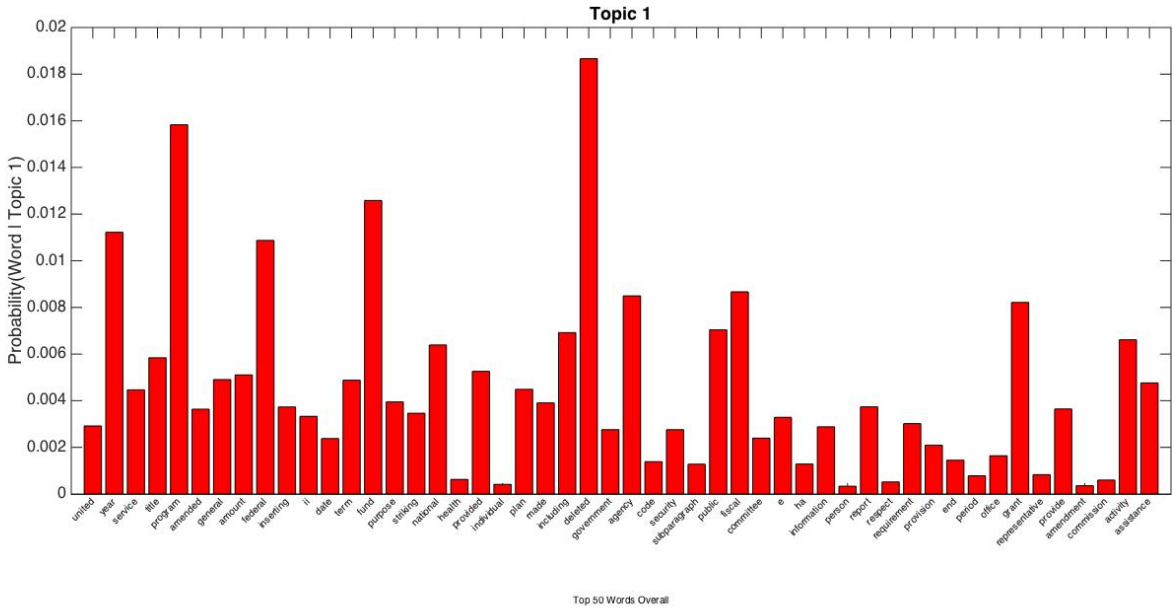


Figure 2: Word probability distribution for Topic 1

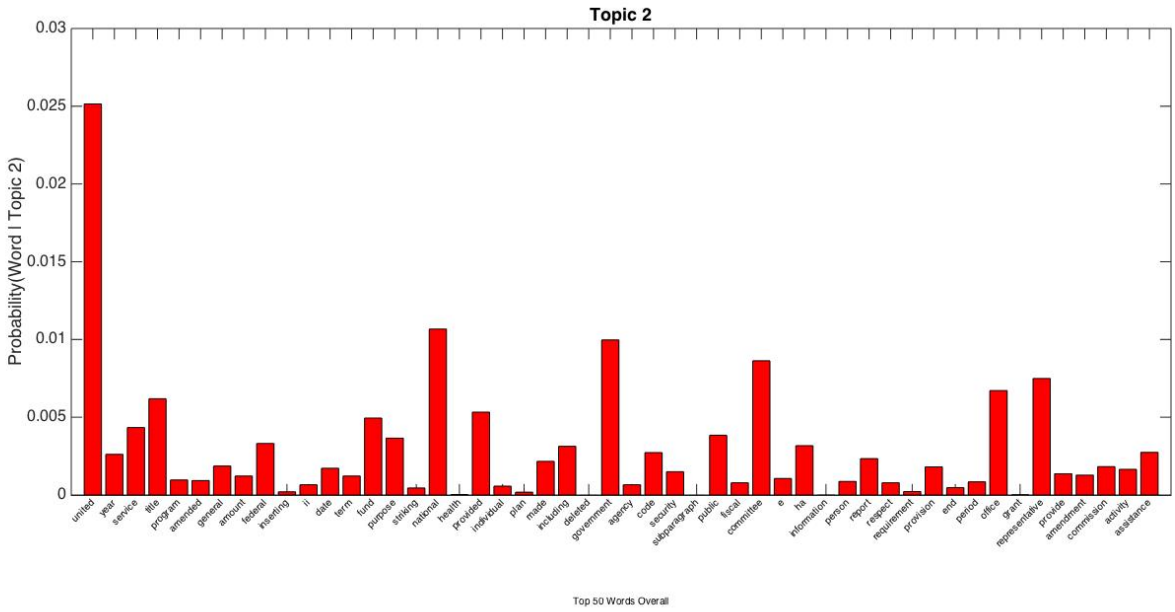


Figure 3: Word probability distribution for Topic 2

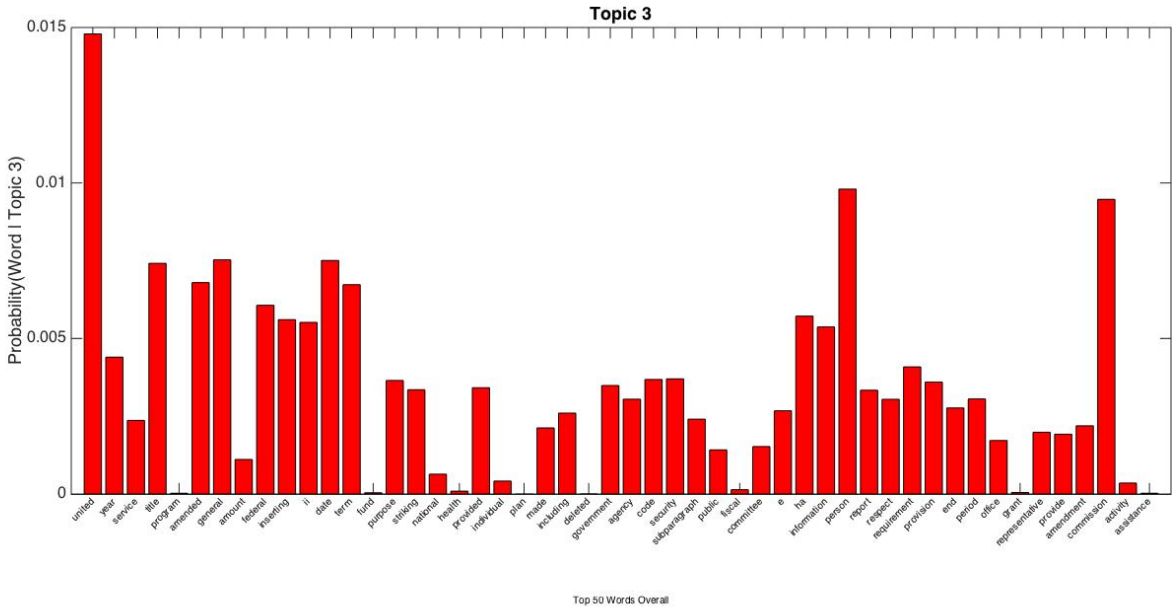


Figure 4: Word probability distribution for Topic 3

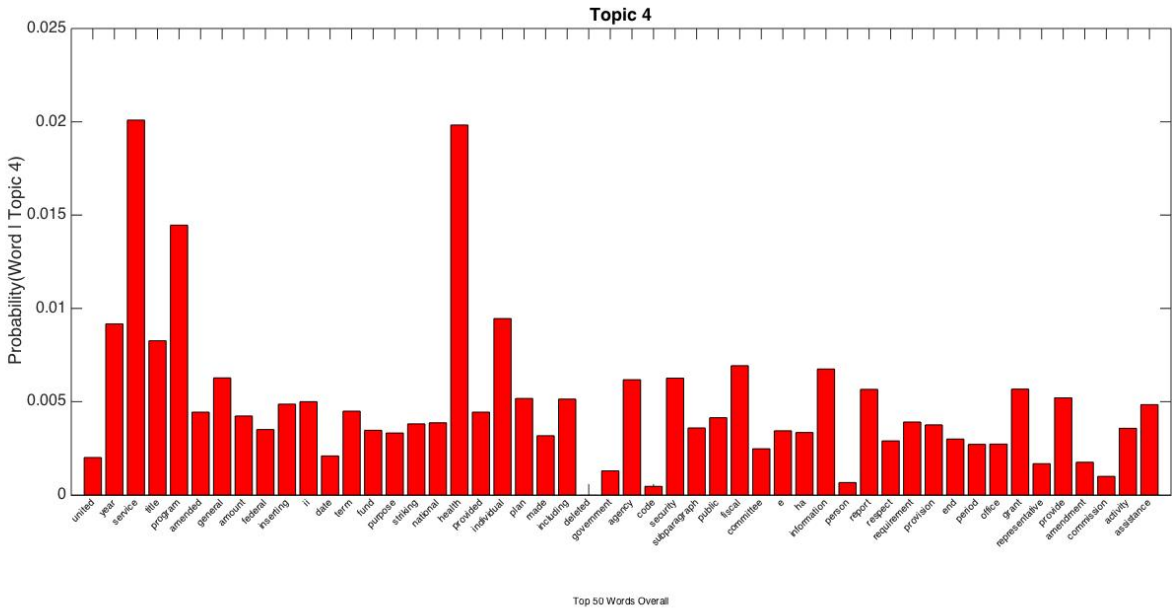


Figure 5: Word probability distribution for Topic 4

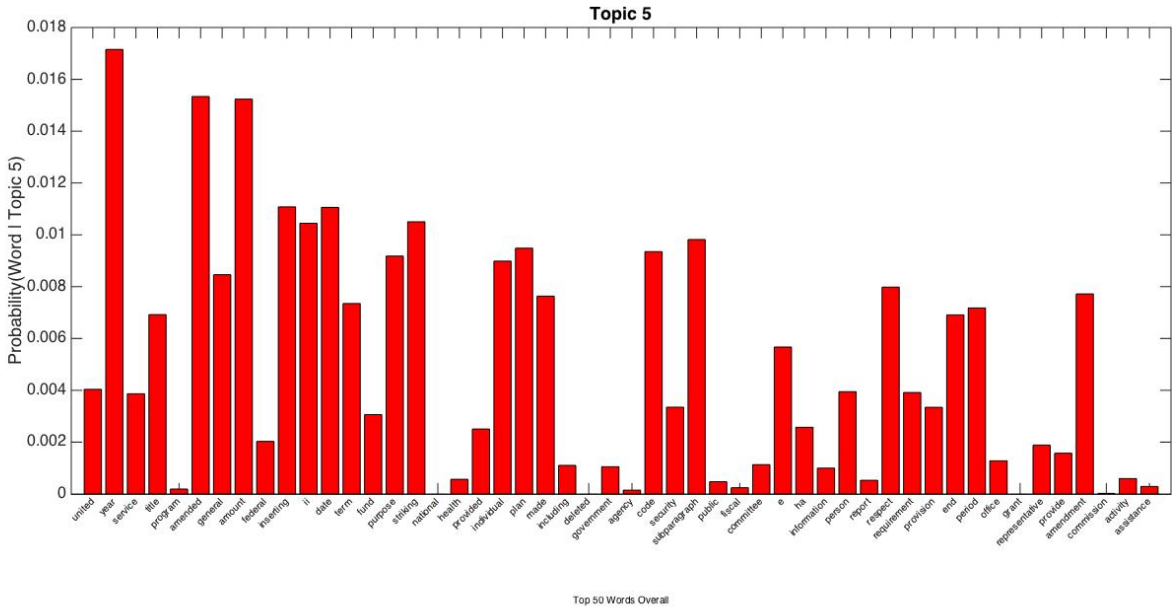


Figure 6: Word probability distribution for Topic 5

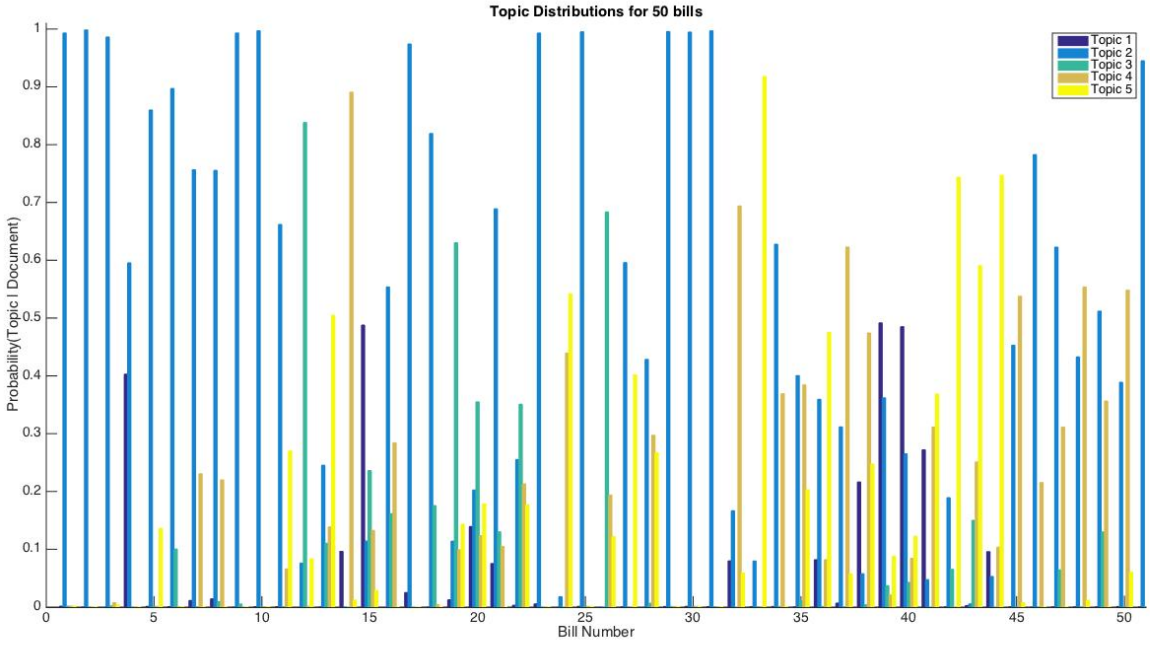


Figure 7: Topic distributions for 50 randomly chosen bills

5 Random Forest Classifier

After using LDA to model legislative texts as distributions of topics, we decided to take the ensemble approach and implement the random forest algorithm [6] as a binary classifier to make predictions about bill outcomes. In generating our random forest, we use bootstrap aggregating, or bagging, to grow individual decision trees.

For each tree, we randomly sample m examples from an original training set of size m without replacement and use this bootstrap sample to train the tree. The thresholds that ultimately

become the decision rules for that tree are chosen at each non-leaf node by considering multiple splits for each feature in a random subset. The samples are sorted by their value for that feature, and we take as splits the midpoints between those samples which have different labels. Each split is considered by calculating the information gain, or expected decrease in entropy, at that threshold. Information gain and entropy, a measure of sample impurity, are defined as:

$$Entropy(S) = \sum_{i=1}^{c} -p_i \log_2 p_i$$

$$Gain(S, A) = Entropy(S) - \sum_{v \epsilon Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

Finding the threshold with the lowest entropy, i.e. the threshold that results in the most homogenous sample, increases the likelihood of us reaching a leaf node. Therefore, we choose the split with the highest gain, or lowest decrease in entropy, for each feature. Of those, the feature with the greatest maximum gain and its corresponding split value are chosen as the decision rule. Ultimately, each leaf node culminates in a posterior probability of "passing" and casts a vote for "passing" if the probability is more likely than the alternative, or "failing". The samples that aren't used to grow the tree, our out-of-bag samples, are used to produce the out-of-bag error, which we used as an early estimate of classification error.

At test time, for each bill in the test set we produce individualized posterior probabilities for each topic generated by LDA during training based on the words in that document. Then, using our grown random forest, we run each test sample down each tree in the forest, making decisions at every level based on decision thresholds for a specific feature (determined during training), and we choose the majority vote as the class label.

6 Cross Validation

The hyperparameters of a random forest consist of the maximum depth of each decision tree and the number of randomly selected features to consider at each split. From LDA, we also have a third hyperparameter for the number of topics used to represent each bill. To determine the optimal hyperparameters, we performed 5-fold cross validation on a set of 10,669 bills, including 2073 passed bills and 8596 failed bills. Below are two sets of graphs (one with and one without error bars), showing the relationship between the cross-validation error and the maximum decision tree depth.

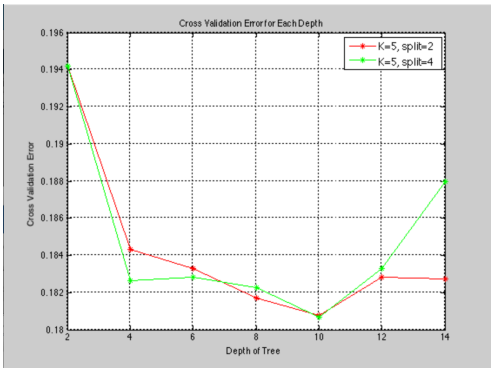


Figure 8: Cross Validation Results for k=5

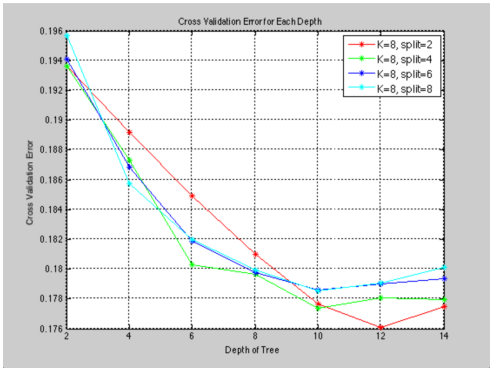


Figure 9: Cross Validation Results for k=8

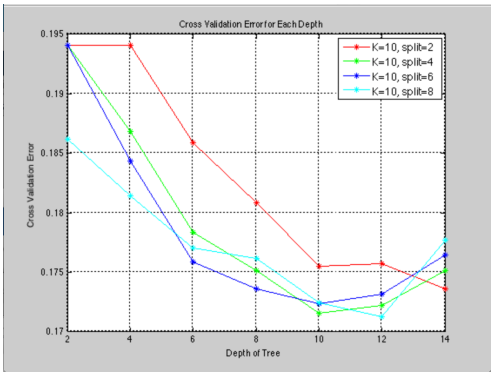


Figure 10: Cross Validation Results for k=10

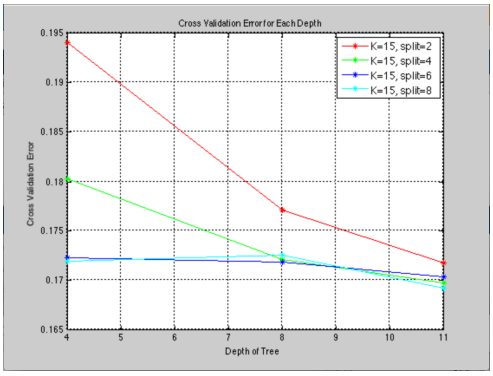


Figure 11: Cross Validation Results for k=15

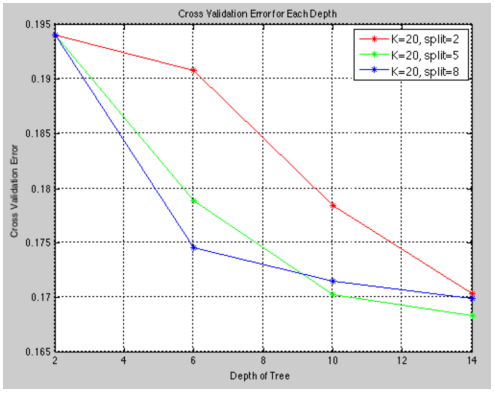


Figure 12: Cross Validation Results for k=20

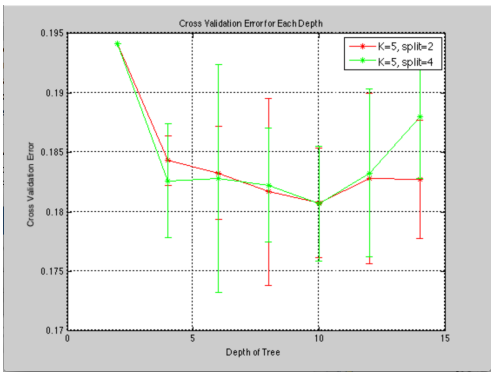


Figure 13: Cross Validation Results for k=5 with error bars.

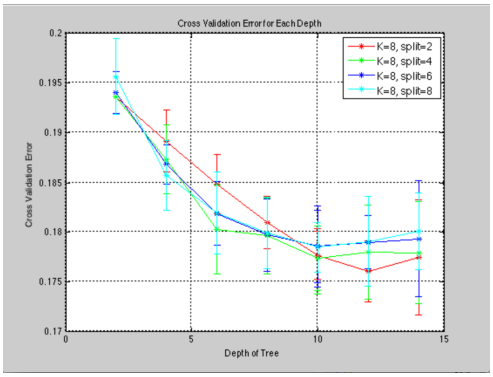


Figure 14: Cross Validation Results for k=8 with error bars.

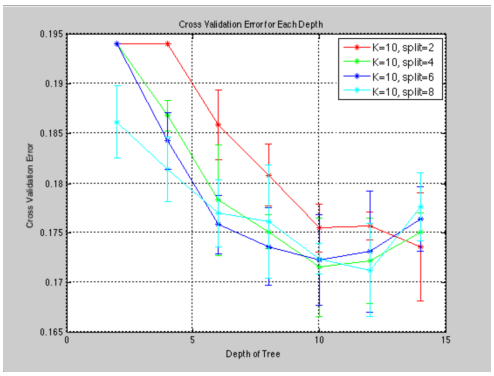


Figure 15: Cross Validation Results for k=10 with error bars.

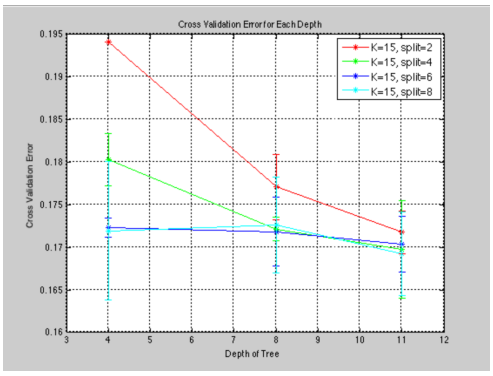


Figure 16: Cross Validation Results for k=15 with error bars.

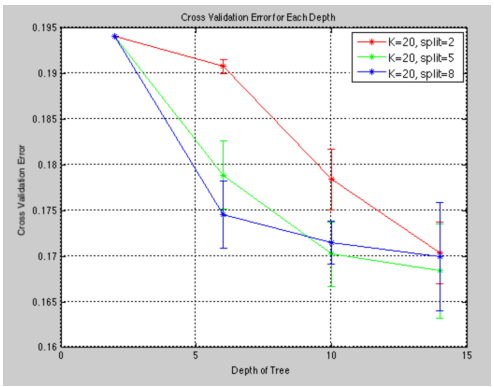


Figure 17: Cross Validation Results for k=20 with error bars.

As we increase the depth, the cross validation error decreases until a minimum is reached, after which the error increases. Since the number of failed bills greatly exceeds the number of passed bills, a larger max depth is appropriate; a lot of branching is necessary before reaching a node with a majority of passed bills. Our decision tree is restricted to be a binary tree, so for larger numbers of topics, the optimal depth must also increase. This is because each node split is based on exactly one feature; in order to describe the set of feature values that represent passed bills when there are many features, more branching is necessary. In fact, for K=15, we are unable to determine the optimal depth based on the set of hyperparameter values we considered, as the optimal depth seems to exceed 14.

Below are two sets of graphs (one with and one without error bars), showing the relationship between the cross-validation error and the number of randomly selected features considered at each split.

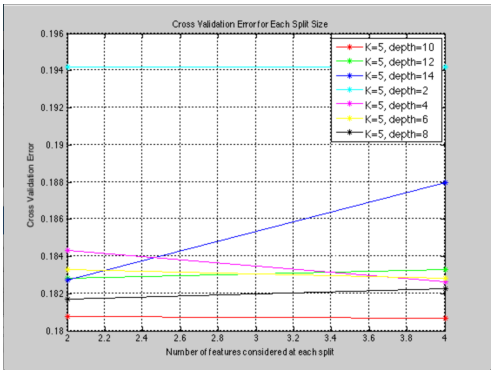


Figure 18: Cross Validation error vs. Split size, k=5

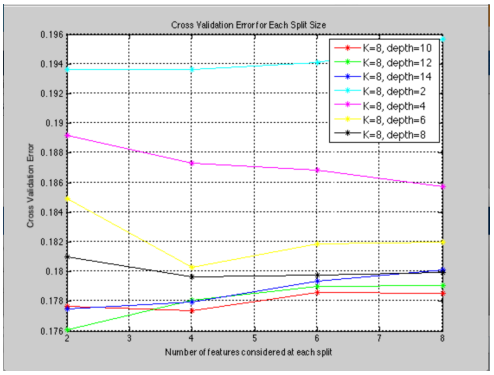


Figure 19: Cross Validation error vs. Split size, k=8

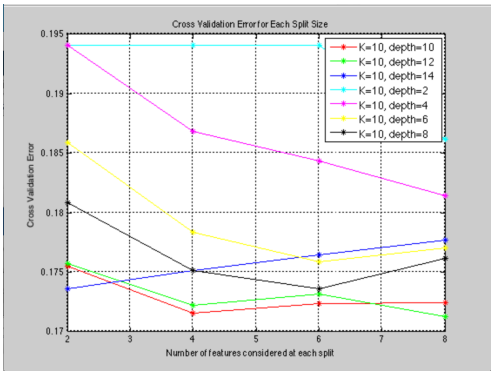


Figure 20: Cross Validation error vs. Split size, k=10

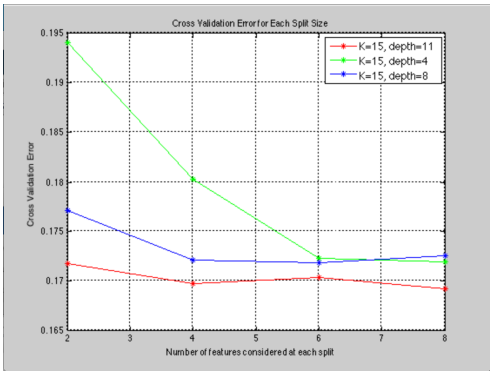


Figure 21: Cross Validation error vs. Split size, k=15

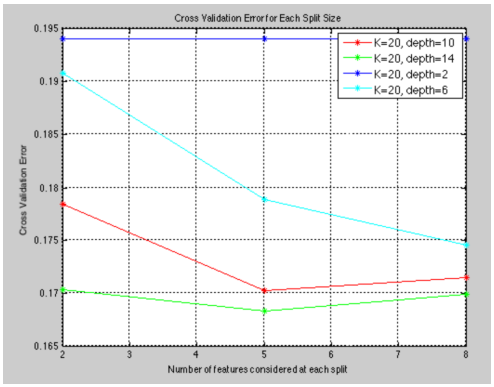


Figure 22: Cross Validation error vs. Split size, k=20

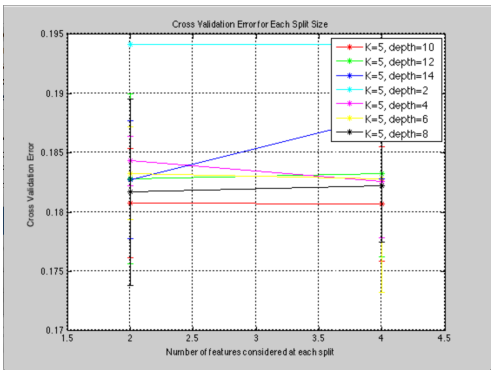


Figure 23: Cross Validation error vs. Split size, k=5, with error bars

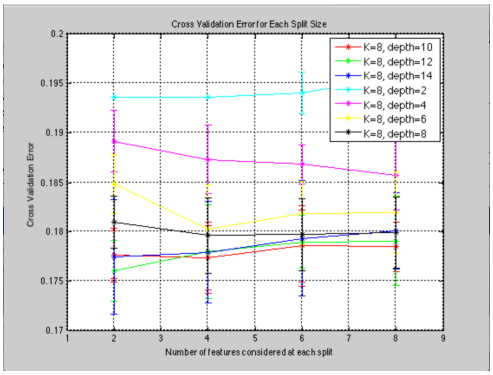


Figure 24: Cross Validation error vs. Split size, k=8, with error bars

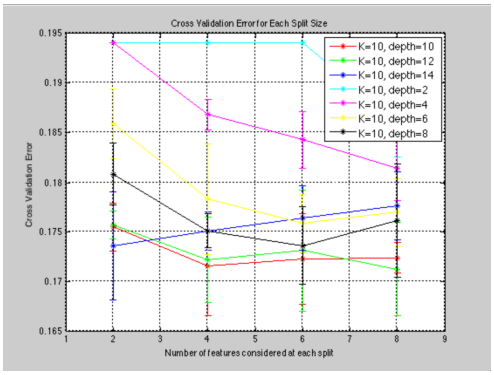


Figure 25: Cross Validation error vs. Split size, k=10, with error bars

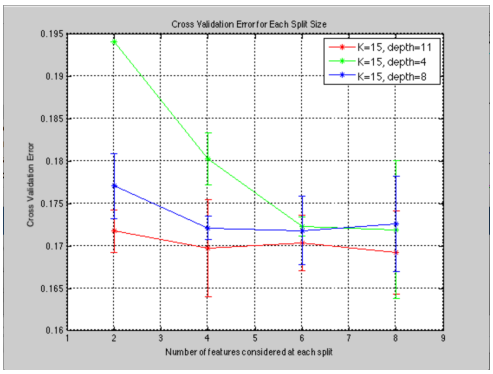


Figure 26: Cross Validation error vs. Split size, k=15, with error bars

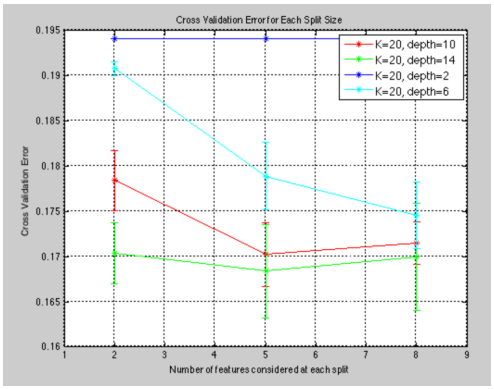
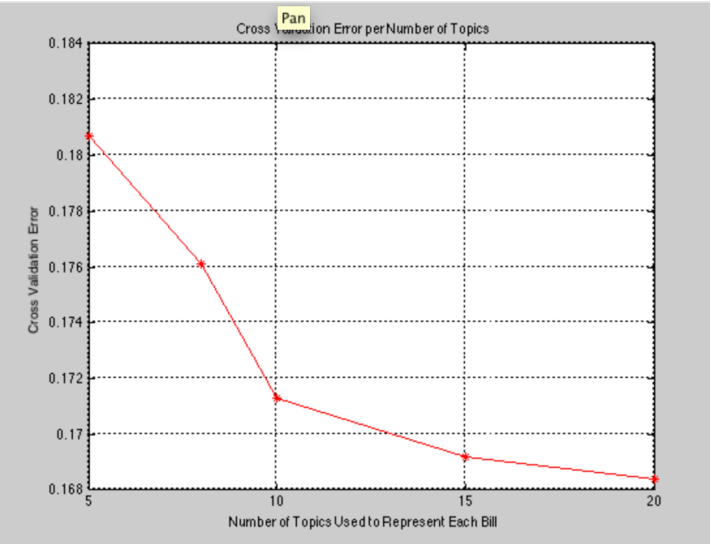


Figure 27: Cross Validation error vs. Split size, k=20, with error bars

The motivation for restricting the number of features considered at each split (called feature bagging) is to reduce the correlation between trees when one or a few features are very strong predictors of whether a bill will pass. A split size of 2 gives high error, but the remaining higher split size values seem to consistently give the same error. Therefore, not restricting the number of features considered at each node split seems to be optimal. Since splitting barely affects the overall classification error, it is possible that no feature strongly determines the label of the bill.

Below is a graph showing the relationship between the cross-validation error and the number of topics used to represent each bill, K. For each value of K, the error is the classification

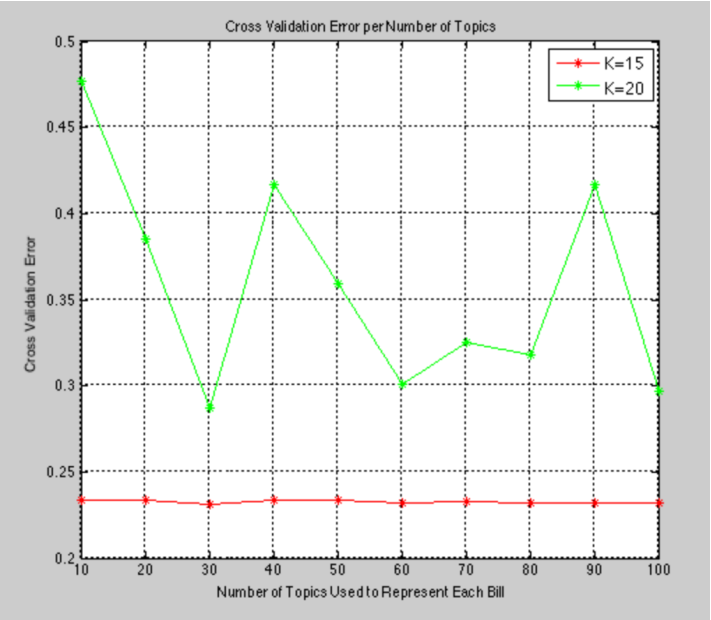
error given by the best architecture of the random forest for that given value of K.



As the number of topics increases, the error decreases and seems to level off at K=15 or K=20. We can see underfitting for small numbers of topics presumably because they aren't sufficient to describe the wide variety of bills that exist.

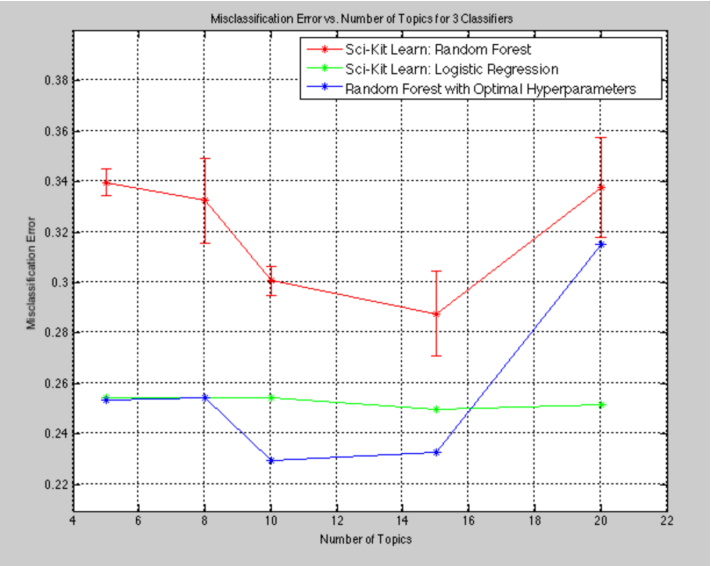
7 Results and Discussion

The following graph shows the classification error as a function of the number of trees in the forest:



Typically, a larger number of trees in the forest will increase the accuracy of the majority vote, decreasing the error. For K=20, there is indeed a slight downward trend, but many huge oscillations, particularly at 40 and 90 trees, seem to suggest that the random forest may be unstable and inconsistent. K=15, the performance is much more stable and consistent, and the error stays constant at around 23%.

Sci-Kit Learn provides libraries for logistic regression and random forest, which are used as baselines for our own random forest implementation. The following graph shows the error using the three classifiers on a training set of 10,669 bills (from cross-validation) and a test set of 6,143 bills (1566 passed bills and 4577 failed bills).

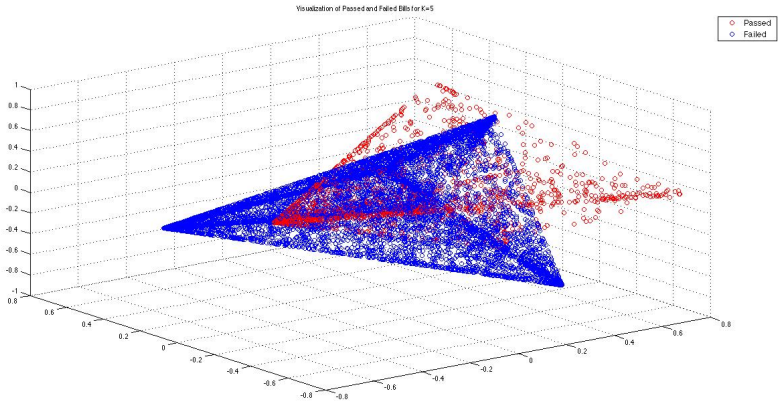


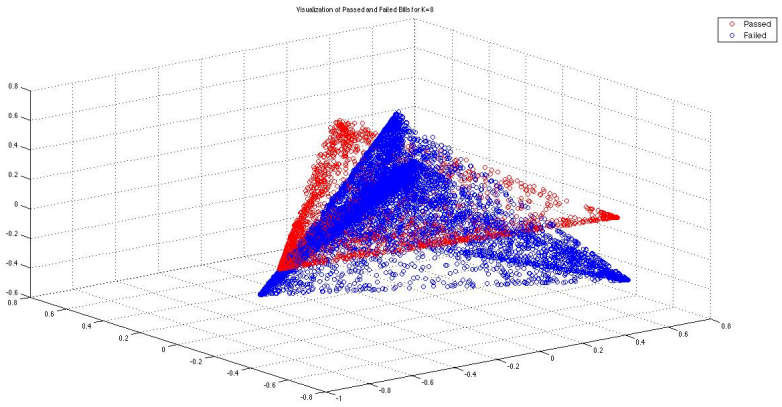
Both logistic regression and our random forest performs better than Sci-Kit Learn's random forest. For K=10 and K=15, our random forest performs better than Sci-Kit Learn's logistic regression. Our random forest produces an extremely high error for K=20, further demonstrating that K=20 is unstable.

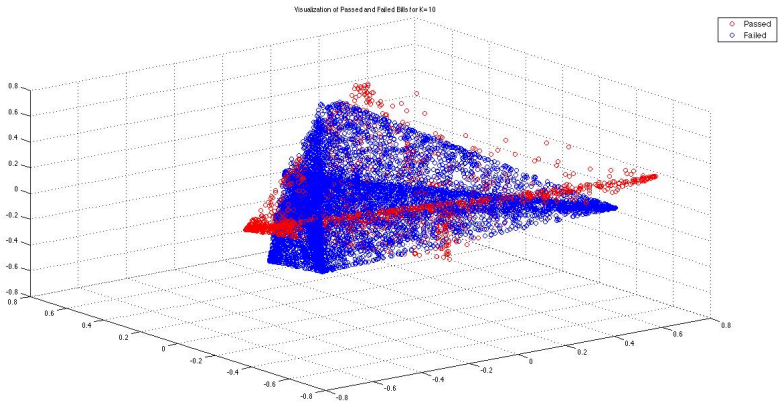
We note that the error is much larger overall compared to cross-validation. One potential reason may be that the proportion of passed bills is much higher in the test set (25%) than in the validation sets from cross-validation (19%). When K is set to 20, the significantly worse performance on the testing set relative to cross-validation may be attributed to the different ratio of passed to failed bills.

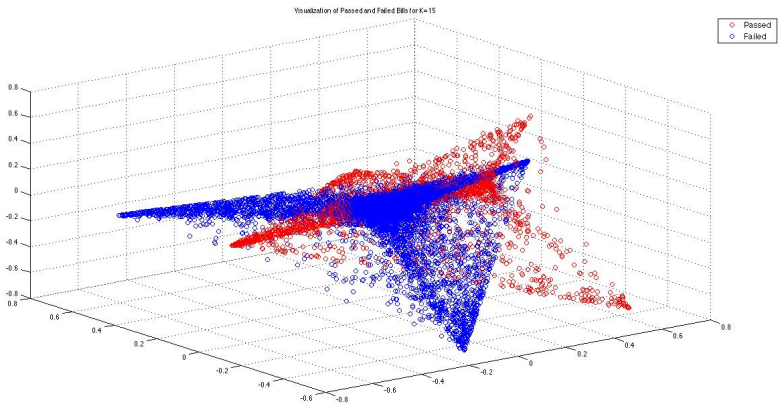
8 Data Visualization

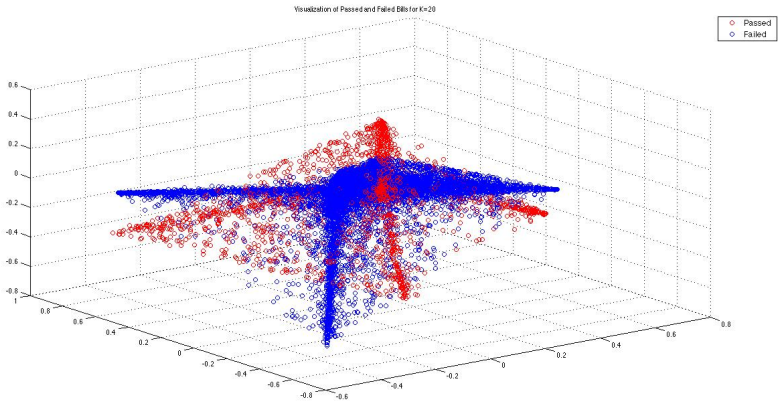
To visualize the data, we used Principal Component Analysis to reduce the dimensions of the feature vectors for each bill to three, and plotted the data points in the following graphs:


Visualization of Passed and Failed Bills for K=5
Passed
Failed


Visualization of Passed and Failed Bills for K=8
Passed
Failed


Visualization of Passed and Failed Bills for K=10
Passed
Failed


Visualization of Passed and Failed Bills for K=15
Passed
Failed



Visualizing the data helps to understand the level of difficulty of the classification problem. Although there is some separation between points from the two classes, the distinction is not obvious.

9 Data Sets Revisited

Because our congressional bill data set is quite large, we had the luxury of experimenting with a varied composition. However, given the time constraints, we were not able to fully test these specialty cases. Nevertheless, we found the results worth mentioning and perhaps worth investigating at a later time. Below we explore the results of varying pass to fail ratios and definitions.

9.1 Big Data

For congresses for which there is comprehensive bill text readily available (i.e. Congresses 103–113), there are some 123,012 bills in total with 86,032 unique words. Each bill has a maximum of 276,152 words. Unfortunately, this data set failed to run in `MATLAB` on machines with 16GB of memory, as the matrices being built for LDA exceeded the available memory on the machine. Here, algorithms that are optimized for large data sets such as Online LDA [8] might have been more appropriate.

9.2 Balanced Classes

The pass to fail ratio for our original data set (detailed in section 3.2) was 1:3. We created a new data set that is about the same size, but with a pass to fail ratio of 1:1 (details below). We run our classifier for 20 topics (this was after we found that k=20 produced sub-optimal results, however there was not enough time to re-run LDA for a different value of k as 20 iterations of Gibbs Sampling takes 10+ hours). Our classification error on this data set is 49.9%, compared to scikit-learn's random forest classification error of 40.7%.

Total number of bills:	18,889
Number of bills in training:	13,550
Number of bills in testing:	5,339
Pass to Fail Ratio in total:	48% Pass : 52% Fail

9.3 Unambiguous Bill Outcomes

Our original problem defined the `fail` class as containing bills that either were explicitly voted upon and failed the vote, or did not pass the subcommittee stage, or were un-acted upon. If we instead define `fail` as the class only containing bills that are explicitly voted upon and fail, and define the `pass` class as before (containing bills that are explicitly voted upon and pass), then we have the following composition of our data set.

Note that out of our total data set of 123,012 bills, only a total of 4% (5026) fit these class definitions, including only 350 total bills that explicitly fail. This composition agrees with

Total number of bills:	5,026
Number of bills in training:	3,971
Number of bills in testing:	1,055
Pass to Fail Ratio in total:	93% Pass : 7% Fail

Govtrack's analysis in [9]. For this reason, we chose not to adopt such explicit labels for our overall classification. When we perform classification on this data set, our classification error is 8.6%, compared to scikit-learn's random forest classification error of 8.5%. Once again, this was done with 20 topics.

10 Implementation Details

Based on the tutorial from [10], we were able to draft the pseudocode for LDA, as well as adopt the preprocessing of stop words, lemmatization, and word stemming, which can be found in the following files: extract_words.py, lda.m, learn_distributions.m, random_topic_assignment.m, word_dist.m, topic_dist.m, perplexity.m. We ultiminately decided to implement the code in a different language (`MATLAB`).

Scikit learn provides libraries for running logistic regression and random forest, which we use as baselines against our random forest implementation. Scikit learn's library is used in test_sklearn_forest.py and in cross_validation.py (within comments). There is also a Scikit learn library for PCA, which we used in pca_visualization.m and pca_dimred.py (within comments). It can be found at [11] and instructions for environmental setup can be found in the `README`.

References

[1] Probabilistic Topic Models; David M. Blei, 2012 (http://www.cs.princeton.edu/~blei/papers/Blei2012.pdf)

[2] Predicting Legislative Roll Calls from Text; S.M. Gerrish, D.M. Blei, 2011 (http://www.cs.columbia.edu/~blei/papers/GerrishBlei2011.pdf)

[3] Govtrack.us - About (https://www.govtrack.us/about)

[4] Latent Dirichlet Allocation; D.M. Blei, A.Y. Ng, M.I. Jordan (http://www.jmlr.org/papers/volume3/blei03a/blei03a.pdf)

[5] Gibbs Sampling in the Generative Model of Latent Dirichlet Allocation; T. Griffiths (https://people.cs.umass.edu/~wallach/courses/s11/cmpsci791ss/readings/griffiths02gibbs.pdf)

[6] Random Forests; L. Breiman, A. Cutler, (https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm)

[7] Decision Trees - Tutorial (http://dms.irb.hr/tutorial/tut_dtrees.php)

[8] Online Learning for Latent Dirichlet Allocation; M.D. Hoffman, D.M. Blei, F. Bach (https://www.cs.princeton.edu/~blei/papers/HoffmanBleiBach2010b.pdf)

[9] Kill Bill: How many bills are there? How many are enacted?; J.Tauberer (https://www.govtrack.us/blog/2011/08/04/kill-bill-how-many-bills-are-there-how-many-are-enacted/)

[10] https://shuyo.wordpress.com/category/machine-learning/lda/

[11] http://scikit-learn.org/stable/