Predicting Congressional Bill Outcomes

Over the course of this quarter, we built a predictor that identifies which US bills will succeed in both the House and in the Senate. The predictor is modeled on Congressional data from the 109th - 113th Congresses (spanning the years 1997 - 2010) from a wide range of sources. We evaluated our system based on whether or not we were able to predict the outcome of a particular bill correctly.

Data Sources & Infrastructure

Data from GovTrack & Thomas.gov

We downloaded metadata of all bills from the 109th - 113th Congresses from the @unitedstates project's bulk data downloads page. Documentation on the structure of these JSON files can be found on this page in the group's wiki.

MapLight's Congressional Bills Data

We relied most heavily upon MapLight's data about the contributions given to members of Congress from interest groups that support or oppose various bills. We pulled information on each session for each kind of bill with a Python script that scraped the Maplight data for the 109th - 114th congressional sessions, where each bill had an outcome (SUCCESS or FAIL) from GovTrack associated with it.We have made the structured data available here. For bills from previous congresses (i.e. > 2 years old), if the status is "Referred to committee" or "Introduced" or "Reported to committee" then itâs a failure.

Bill types ââ **HR:** House Bill, **S:** Senate Bill, **HJ Res:** House Joint Resolution, **SJ Res:** Senate Joint Resolution, **H Con Res:** House Concurrent Resolution, **S Con Res:** Senate Concurrent, **H Res:** House Simple Resolution, **S Res:** Senate Simple Resolution

Approach

We modeled the challenge of classifying bill outcomes as a classic machine learning task. Using the data on congressional bills from the sources listed above, we were able to experiment with several machine learning approaches for the classification task.

Baseline: Using a conjecture from a previous study from Michigan State on bill advancement (see the section on Prior Work), we established a baseline prediction of a billâs success using just one feature, the total sum of lobbying money in support for that bill.

Oracle: Our ground truth for bill outcomes is taken from our bill status data with GovTrack. In order to interpret bill status as a binary outcome (SUCCESS or FAIL), we mapped all possible statuses to their most likely outcomes. (See the section on GovTrack Data.)

Main Approach: Our first step after collecting and cleaning the data was representing it in an appropriate feature form. Because we wanted to include unique lobbyists, legislators, organizations, and industries in our features and their effect on the bill outcome, we took on a text-classification "bag of words" approach to representing these features. For our particular data, we associated lobbying amounts (either given or received for, and in support or against a bill) to each feature instead of binary indicators (feature is present vs. not present). A few of the different numerical representations we considered were absolute value of money for a bill, ratio of money for and against a bill, and the net sum of the money for a bill (i.e. money given/received in favor of the bill - money given/received in opposition to the bill). Some examples of resulting features are given below:

- If Senator Grinch received $14,000 total in support for **Bill #1919: World Peace** and $30,000 against, then the feature "Senator Grinch" would map to `-16,000`.
- If the sum for representatives in a certain voting district of California was $3,000 in support and $1,000 against, then the feature "CA-2" would map to `2000`.
- If the manufacturing industry contributed a total of $40,000 against and $0 in support, then the feature "manufacturing" would map to `-40,000`.

With our aggregated training examples, our resulting data included over 11,000 features for just under 3,000 bills.

Our second step was to take our newly formed feature representation and assess a second baseline accuracy from naive or linear supervised classification algorithms. We used **Naive Bayes** and **Logistic regression** respectively to accomplish this, as they are relatively quick and simple classification algorithms.

Our final step was to move to more sophisticated approaches that were more suitable for dealing with high-dimensional feature spaces. Similar to methods in classc text classification, we decided to use models like **Support Vector Machines** (SVMs) that are robust but also capable of modeling non-linear boundaries. We also experimented with Perceptron to see if this single-layer network could more effectively capture relationships betwee our various features. In order to perform both classifications, however, we had to first reduce our feature space using **Principal Component Analysis** (PCA). In addition to reducing our feature space from ~11,000 to ~500, PCA also helped us discover that these 500 features capture the most variation in the data.

Literature Review & Prior Work

Statistics from previous Congresses : GovTrack provided an important overview of the bill success rates of previous congresses, which you can find here. According to GovTrack's data, nearly 0% of bills are considered "failed legislation". The vast majority of bills that don't become law simply die in committee or some other intermediate step.

Lobbying & Congressional Bill Advancement : A Michigan State University paper titled *Lobbying & Congressional Bill Advancement* found that "the amount of interest group lobbying is associated with majority party sponsorship, wide cosponsorship, and high-profile issues" and that "lobbying helps predict whether bills advance through committee and each chamber, independent of congressional factors typically associated with bill advancement".

Predicting Congressional Bill Outcomes: In 2012, Stanford students took on a very similar question in *Predicting Congressional Bill Outcomes* where the team aimed to answer the question "Can we predict the voting behavior of a Representative on a given bill using only the frequency of words in the bills text?", and they relied on text classification.

Lobbying on Effective Legislation: Revolving Door Lobbyistsâ Targeted Legislative Influence: A University of Texas at Austin paper titled *Lobbying on Effective Legislation: Revolving Door Lobbyistsâ Targeted Legislative Influence* studied the varying levels of influence exerted by lobbyists on each step of the legislative process.

Error Analysis

Naive Bayes with Principal Component Analysis (PCA)

Summary

We began by using Naive Bayes as our initial predictor, which performed poorly in the case of a PCA-transformed space. The predictor correctly classified points in the test set just over 26% of the time, far worse than the 50% that one would expect with chance. Our initial reaction was one of shock, but after some reflection we realized that the source of the problem was likely the approach's "naive" assumption of conditional independence between each of the features. This joint probability model can be expressed as:

$$\begin{aligned} p(C_k|x_1,\dots,x_n) &\propto p(C_k,x_1,\dots,x_n) \\ &\propto p(C_k)\ p(x_1|C_k)\ p(x_2|C_k)\ p(x_3|C_k)\ \cdots \\ &\propto p(C_k)\prod_{i=1}^{n} p(x_i|C_k)\,. \end{aligned}$$

It's intuitively obvious that many of the features are closely correlated. In particular, we included a wide variety of features representing many different aspects of the lobbying processes, and as a result one would expect there to be many features that communicate overlapping information about a given bill.

In comparison to the classification in untransformed space, we can see that Naive Bayes suffers after pre-processing with PCA. This can be explained by the fact that PCA extracts features that maximize variation within the data, and is heavily dependent on covariations between features. Naive Bayes, on the other hand, applies a fundamentally different assumption with the data when creating its models and could even derive distortions from the PCA-selected features.

Correctly Classified Instances	705	26.1208 %
Incorrectly Classified Instances	1994	73.8792 %

Detailed Accuracy By Class

	Precision	Recall	F-Measure	Class
	0.067	0.806	0.124	SUCCESS
	0.943	0.223	0.361	FAILURE
Weighted Avg.	0.886	0.261	0.346	

Confusion Matrix

a	b	<-- classified as
141	34	a = SUCCESS
1960	564	b = FAIL

Naive Bayes without PCA

Summary

Correctly Classified Instances	2060	75.7075 %
Incorrectly Classified Instances	661	24.2925 %

In comparison to the classification in untransformed space, we can see that Naive Bayes suffers after pre-processing with PCA. This can be explained by the fact that PCA extracts features that maximize variation within the data, and is heavily dependent on covariations between features. Naive Bayes, on the other hand, applies a fundamentally different assumption with the data when creating its models and could even derive distortions from the PCA-selected features.

Detailed Accuracy By Class

	Precision	Recall	F-Measure	Class
	0.131	0.486	0.206	SUCCESS
	0.956	0.776	0.857	FAIL
Weighted Avg.	0.902	0.757	0.814	

Confusion Matrix

a	b	<-- classified as
86	91	a = SUCCESS
570	1974	b = FAIL

Logistic Regression after Principal Component Analysis (PCA)

Summary

Logistic Regression was quite effective at predicting bill outcomes, coming in second place with an 83.7% success rate. The feature space was too large to perform logistic regression with reasonable time and computation power, so we had to first transform the data with PCA before creating the models.

Intuitively, logistic regression in conjunction to PCA should perform relatively better than Naive Bayes; logistic regression is dependent on covariances and does not make the same probabilistic assumptions about independence as Naive Bayes. In fact, Aguilera et al. proposes that PCA actually enhances logistic regression of high-dimensional, multi-collinear data. Given this, it is no surprise that logistic regression results in a significant improvement from the baseline performance.

Correctly Classified Instances	2259	83.6977 %
Incorrectly Classified Instances	440	16.3023 %

Detailed Accuracy By Class

	Precision	Recall	F-Measure	Class
	0.125	0.251	0.167	SUCCESS
	0.944	0.878	0.91	FAILURE
Weighted Avg.	0.891	0.837	0.861	

Confusion Matrix

a	b	<– classified as
44	131	a = SUCCESS
309	2215	b = FAIL

Support Vector Machine after Principal Component Analysis (PCA)

Summary

Support Vector Machine performed the best out of the four methods we used, correctly predicting the outcome of almost 94% in 10-fold cross validation. This is not surprising, as SVM is generally a highly effective algorithm for large feature spaces, robust due to regularizations, and fast due to kernel use. In addition, SVM is capable of finding non-linear decision boundaries, which is ideal for features like ours with complex or dynamic relationships. Due to the performance and speed of this algorithm, we would most likely favor SVM over other approaches in this analysis.

Correctly Classified Instances	2530	93.7384 %
Incorrectly Classified Instances	169	6.2616 %

Detailed Accuracy By Class

	Precision	Recall	F-Measure	Class
	0.074	0.003	0.65	SUCCESS
	0.997	0.926	0.94	FAILURE
Weighted Avg.	0.937	0.866	0.921	

Confusion Matrix

a	b	<– classified as
13	162	a = SUCCESS
7	2517	b = FAIL

Voted Perceptron after Principal Component Analysis (PCA)

Summary

Voted Perceptron achieved an accuracy of 65% on our 10-fold cross validations. One explanation for the sub-par performance is that this algorithm is a "shallow" single layer variation of the far more powerful neural networks. In order to achieve higher accuracy in our classifications, a better approach would be applying full neural networks instead of the voted perceptron. One disadvantage to this however is that Perceptron was by far the slowest of all our algorithms (and neural nets even slower), which also presents a significant challenge in computational power and time.

Number of perceptrons = 1133

Correctly Classified Instances	1755	65.0241%
Incorrectly Classified Instances	944	34.9759%

Detailed Accuracy By Class

	Precision	Recall	F-Measure	Class
	0.079	0.411	0.132	SUCCESS
	0.942	0.667	0.781	FAIL
Weighted Avg.	0.886	0.65	0.739	

Confusion Matrix

a	b	<– classified as
72	103	a = SUCCESS
841	1683	b = FAIL

Contributions: This project was a joint project between Devon Zuegel and Emma Marriott from CS221. Roles played:

Arushi Jain - Data scraping, cleaning and formatting
Devon Zuegel - Project writeup and static website setup
Emma Marriott - Data formatting and preliminary analysis

All three of us worked on applying different models and algorithms to the data for analysis.