Learning US Senate voting behavior from bill sponsorship profiles

Michael Allemann, Roman Bachmann, Liamarcia Bifano, Virginia Bordignon
École Polytechnique Fédérale de Lausanne, Switzerland

***Abstract*—This work aims to model the US senate community as an undirected graph based on the similarity of behavior apprehended from roll-call voting records. First, a historical analysis is carried out, where the evolution of the graph is studied throughout senates 105 to 115. In order to predict the senators' voting patterns, the sponsorship profile for active bills was built for each of these senates. The problem is approached here in two ways: the first one consists in employing a convolutional neural network for graphs and the second one translates the scenario into a graph interpolation problem considering a transductive learning framework. Different prediction accuracy metrics were calculated for each congress, achieving a maximum accuracy of 78% and 86% for the CNN and transductive learning respectively.**

I. Introduction

The United States Senate constitutes one of the two chambers of the United States Congress and is composed of 100 senators, two representing each US state. Each senator serves a 6-year term and elections are structured such that every 2 years approximately a third of the Senate seats is renewed. Therefore, it is assumed that the senate composition is changed every two years and forms a new Senate instance, e.g. the 115$^{\text{th}}$ Senate took place between 2016 and 2018.

As part of the legislative process, senators can propose bills, which will be voted on by the Senate itself and the House of Representatives before actually becoming a law. In general, the voting happens as a roll-call vote, where each senator votes *yea* or *nay* regarding a certain question. Each bill has necessarily a main sponsor, who introduced it in the first place, and optionally a set of cosponsors, which consists of other supporting senators [1]. The number (and diversity) of cosponsors of a bill is thus known to be a good indicator of how well accepted the proposition will be.

Indeed, the problem of predicting the evolution of a bill in any of the Congress houses has been studied in the literature for at least 50 years [2]. More recent developments include machine learning strategies applied to the voting prediction problem [3] based on different levels of prior information. Based on such prediction, the sponsor can use that information to postpone the bill submission until it is sure to pass or until it has enough cosponsors to support it.

Motivated by such, this work proposes a solution for modelling individual voting behavior for each Senate instance (from instances 105 to 115), by representing the Senate as a social network and senators as its nodes. The edges between nodes can be built from past voting behavior, and the graph structure can then be employed to provide learning algorithms with node-to-node similarity information.

First, in order to better understand the general data properties, a preliminary data analysis is performed, where the historical evolution of the respective adjacency matrices are inspected. Particularly the profile for the last Senate in office, the instance 115, is further detailed to motivate the examples from the following sections.

Finally, two learning methods are proposed in the scope of this document: graph interpolation applied to transductive learning and convolutional neural networks (CNN) for graphs. Both strategies use as input the graph structure and the sponsorship profile from the bill, whose result one wishes to anticipate. Their accuracy is compared for all Senate instances and meaningful conclusions on their performance are drawn in Section IV.

II. Preliminary analysis

In this section we will present the data analysis, scientific questions and voting behavior that were investigated and modeled in this project. All data used in this project was retrieved from the ProPublica Congress API [4].

A. Graph evolution

Over the two years that each senate holds, the interaction among senators can evolve and it might be interesting to investigate if there is a certain pattern around it and the period of year. In addition, this pattern can shift if there is an unexpected event or a certain power dispute is happening. The following analyses are based on the votes position of all senators collected over the last 20 years (1997 - 2008 or 105 senate - 115 senate). The figure 1 shows the averaged adjacency matrix of each senate, which is the average of all adjacency matrices, where each entry *(ij)* is one if senator *i* had the same vote position as senator *j* and zero otherwise. Besides that, the rows and columns are ordered by party and it starts with Democrats, then independents and finally Republicans. From figure 1 it is possible to notice that the degree of connections changes not only between Republicans and Democrats but also within parties as well. For instance, Republicans were more cohesive during Senate 109 than during Senate 110. In section II-B we present an analysis of the adjacency matrix during the Senate 115.

A set of metrics can be extracted from the adjacency matrices and analyzed over time such as number of edges, density and degree of centrality. The figure 2 presents a box

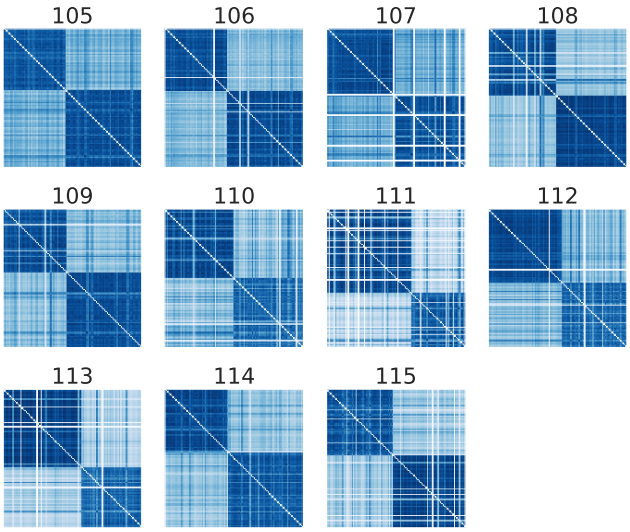

Figure 1. Adjacency matrices evolution over 20 years.

plot with the three metrics cited above for each Senate. We can see that the pattern over time is well behaved, except for the Senate 111 which has an expressively lower number of connections and is considerably sparser. This phenomenon is due to the fact that it is the first Senate in which Democrats were in charge of decisions, meaning that most bills' results were the outcome of a cohesive majority of Democrats. In this Senate, Republicans were disagreeing among themselves and could barely stand united with a position, so most of the bills were the result of Democrats cohesion.

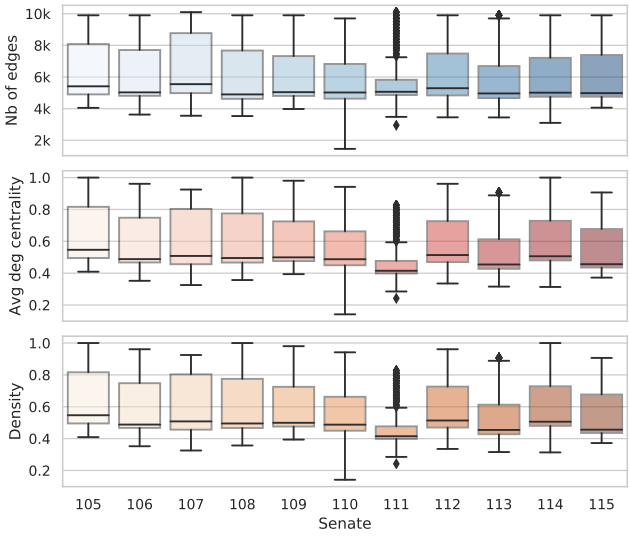

Figure 2. Adjacency matrices general metrics.

Another way to analyze the level of unity between the parties is to first get the result of a vote for each party, i.e. the number of *yea* and *nay* for Republicans and Democrats individually. Then, repeat it for all votes relative to that Senate, giving us an intuition on how *yea* and *nay* votes differ within the same party throughout the term. Finally, we contrast these results so as to evaluate the proportion of this votes for which both parties are concordant and discordant. To achieve that, we used the Kendall correlation coefficient, which is defined as

$$\tau = \frac{(\#\text{ concordant pairs}) - (\#\text{ discordant pairs})}{n(n-1)/2}, \qquad (1)$$

where n is the number of samples, i.e. the number of votes considered. The table I shows the inter-party Kendall correlation for each congress.

Senate	105	106	107	108	109	110
Kendall	-0.18	-0.24	-0.23	-0.40	-0.33	-0.28
Senate	111	112	113	114	115	
Kendall	-0.58	-0.23	-0.41	-0.41	-0.30	

Table I
KENDALL CORRELATION COEFFICIENTS FOR EACH SENATE.

The correlation between the two parties is usually negative, since both parties tend to oppose each other. This trend is more accentuated for Senate 111, where Democrats were the majority in the Congress and had more decision power. Another interesting feature can be observed from analyzing the proportion of senators within the same party that have the same position, i.e. intra-party concordance. The figure 3 presents the intra-party conformity or concordance as a distribution for all votes in the same Senate.

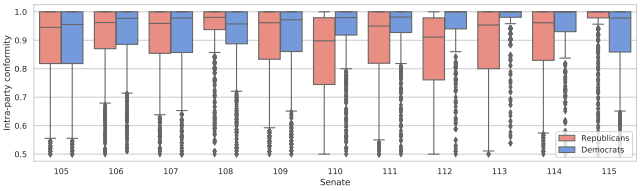

Figure 3. Concordance by party.

As also observed in the adjacency matrices, the Republican party started to lose its cohesion around senate 109 and got it back again to the same level on 113. It is remarkable that Republicans have dominated the Senate until Senate 109 (2005-2007), then Democrats took the power until Senate 113 (2013-2015) and then Republicans dominate until now. As Barack Obama (Democrat) assumed office in 2009 (two years before Democrats have dominated the congress) and Donald Trump (Republican) assumed it in 2018 as well, it meant that two years before a certain party was elected, the congress had already been dominated by them two years before. Finally, it is relevant to notice that, as a party becomes minority in the Senate, it seems to loose cohesion.

B. 115^{th} US senate profile

The 115th Senate took place from January 2017 until January 2019. We used this Senate to build our models, for this reason we present some in depth information about the voting that was done during this senate. There are a total of 105 Senators in the 115th senate. 48 Democrats, 55 Republicans and 2 who are party independent. This means that the Republicans have a majority. There are more than 100 senators, because during the time period 5 dropped out and were replaced. The 115th Senate voted 599 times in total, whereby 267 votes were related to bills. Of these bill related votes 79 were sponsored by Senators. These 79 bills are our dataset for this specific Senate. It is to note, that the rest of the bill related votes were sponsored by members of the House of Representatives and are thus incompatible for our models.

III. Vote prediction

In this section, two approaches for learning a vote model for a given senate are explored: a Convolutional Neural Network for Graphs and a Graph Interpolation strategy. In both cases, only votes associated with bills are taken into account in which there is one main sponsor and possibly multiple cosponsors.

The training dataset used in both cases is composed of a sparse matrix $X^i_{tr} \in \mathbb{R}^{N^i_{tr},N^i_S}$ in which the only non-zero elements are the actual votes for sponsors and cosponsors, and where N_{tr} and N_S represent the training set size and number of senators for Senate instance i. It is assumed here that since these particular senators are highly interested in the positive outcome of their sponsored bill, they should make their position known in advance to other senators. It is considered that there are three possible votes for each senator: *yea*, represented by the number 1, *nay*, represented by -1 and *absence*, taken as 0.

Once trained, the model is then used to predict the outcome of the remaining votes contained in the test set $X^i_{te} \in \mathbb{R}^{N^i_{te},N^i_S}$ for Senate i, so as to validate the learning methods. The absences in the actual voting outcomes are disregarded in the computation of the performance indexes, as the model as-is cannot anticipate nonattendance.

A. Convolutional Neural Networks on Graphs

Recently, an effort has been put into generalizing Convolutional Neural Networks for graph-structured data [5], [6], proposing efficient formulations for the filtering and pooling operations traditionally used for Euclidean data. Using the approach developed in [6], a regression CNN was designed, with two convolutional layers of depth $[32, 64]$, polynomial orders of 4 and no pooling, i.e. the graph was kept with constant dimension.

For each Senate instance, the number of bills sponsored by senators is considerably reduced. So the pooling operation was avoided in order to keep the most information possible within the network. Also, it is worth noting that the Congress API failed to tag votes as bills for the Senate 106, thus there are no training or prediction results for this particular instance.

Note that the parameterization of the convolutional neural network is not optimal, as it was considered that the model to be identified was sufficiently straightforward. Improved results could be obtained by optimizing the learning rate or the polynomial orders for the best prediction accuracy. But more importantly, a richer and larger dataset would have a positive impact on the overall CNN performance.

Example 1 (Senate 115): To illustrate the models developed in this work and to qualitatively evaluate their prediction power, a particular vote for Senate 115 was considered, drawn from the test set of votes from that Senate. First, a training set is selected among all bill-related votes recorded for both sessions of Senate 115. It is assumed here that each sample is time-independent, therefore the set of bills is shuffled to compose train and test sets.

Once the CNN is trained using this training set, the resulting model is used to label this particular vote-example, whose result can be seen in Figure 4. In the given example, the prediction accuracy amounts to 100%, because of the assumption that absences are unpredictable.

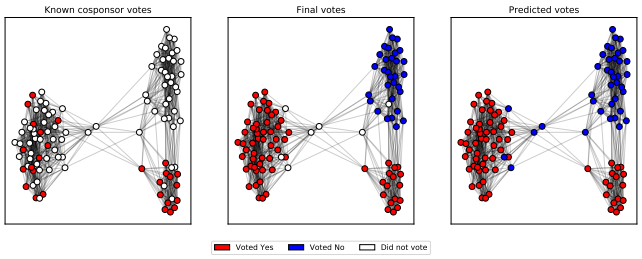

Figure 4. Input, target and predictions using graphs CNN.

B. Graph interpolation model

Another method of inferring senator votes by only considering known cosponsor vote positions is to optimize an offline variational problem, such as minimizing a semi-p-norm of the graph gradient [7], [8]. For this, we assume the underlying signal of all senator vote positions to be "regular" enough to allow inference of the missing labels. While optimizing for the 1 and 2-norms yielded similar accuracy scores, when using the 2-norm there was never much variation within the party clusters, which was more present when using the 1-norm. Since we would like to not only predict a final vote outcome but instead every senator's vote position, we chose to use the 1-norm.

For this task, we take all train set senator vote positions $y^i_{tr} \in \mathbb{R}^{N^i_{tr},N^i_S}$ and build an adjacency matrix and its incident matrix from it by using exact pairwise euclidean distances between senator nodes. The resulting graph G will have

many connections between senators that tend to vote the same. Using an optimizer, we can then compute a prediction $\tilde{z}^i_{te}$ using samples from the test set X^i_{te} by minimizing

$$\arg \min_{\tilde{z}^i_{te}|_C = y} \|\nabla_G \tilde{z}^i_{te}\|_1^1, \tag{2}$$

where C is the set of cosponsors for bill i.

The predictions $\tilde{z}^i_{te}$ are decimal values between -1 and 1 and we have to set a threshold for binarizing them into *yea* (1) and *nay* (-1). It has to be picked carefully, as the resulting signal from the variation minimization is usually very similar for neighbouring nodes.

Example 2 (Senate 115): There are cases where parties vote the complete opposite, cases where they vote the same and some cases where a small subcluster in a party votes differently than their other party members (see Figure 5), all of which require vastly different thresholds.

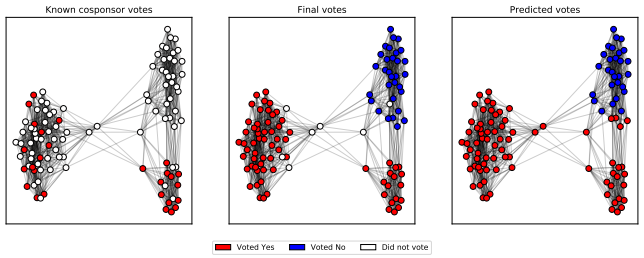

Figure 5. Input, target and predictions using transductive learning. The predictions are 96% accurate.

If an oracle were to tell us the optimal threshold value for each bill, using transductive learning could yield results with accuracy scores far beyond 90%. As this is not the case, we propose another solution.

From the training set X^i_{tr}, y^i_{tr}, we can compute the optimal threshold for each bill via a simple grid search between -1 and 1. We would like to predict this best threshold using some statistics about the input data X, like the number of cosponsors in each party voting *yea* or *nay*, a flag if there are any cosponsors for a given bill in each party, as well as a flag if there are more cosponsors in one party than the other. We build a new training set from the previous one, containing the aforementioned features and the optimal threshold as a target. This dataset is then used to train a Random Forest Regressor, which given new test samples will give us a good threshold. The Random Forest was trained using default *scikit-learn* parameters with 10 estimators and optimized using the Mean Absolute Error loss.

Having constructed the graph and having trained this Machine Learning model on the train set of each Senate, new test samples can be optimized using the transductive learning method and discriminated by predicting a good threshold.

C. Results

In this section, we chose to apply both learning methods not only to the 115$^{\text{th}}$ senate, but to all congresses between 105 and 115 (excluding 106, since the ProPublica API does not return the right data), and compute the accuracy and relative error metrics. The accuracy is only computed over all senators that voted on a given bill. The relative error between a ground truth vector x and a prediction z is given by

$$e_{rel} = \frac{\|x - z\|}{\|x\|}. \tag{3}$$

Figure 6 shows the results from the graph CNN model, while Figure 7 shows the results when using transductive learning. Overall prediction accuracy scores are very high in both models, with the graph CNN reaching up to 78.76% and the variation minimization problem reaching a 86.15% mean accuracy. The relative error reflects the good results observed through the accuracy. Inspecting results from both methods more closely, we observe that the variation minimization and random forest method performs much better than the graph CNN in cases where we don't have a lot of training data (Senates 105, 107 and 108). Training a neural network with many more parameters than a random forest naturally requires more data. For all other cases, both methods perform very similarly.

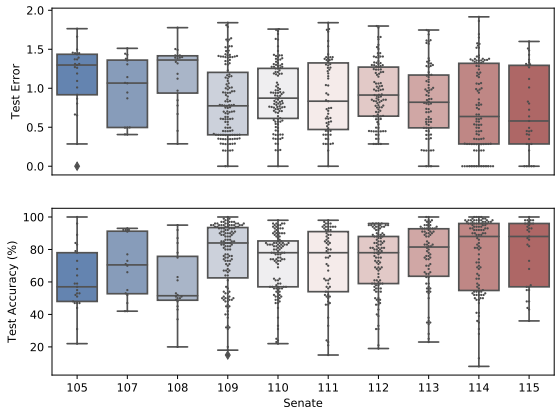

Figure 6. Results for the graph CNN method.

In table II, the mean and standard deviation for the accuracy scores and relative errors are shown considering each of the learning methods. In general, standard deviation behaves in a similar way, while the average values vary considerably from one method to the other.

IV. Summary

This work consisted in analyzing the voting behavior of US senators from Senate instances 105 to 115 as if each of these constituted an independent graph. The assumption that each senator is connected to their peers by vote similarity allowed us to draw significant information from the data to

Senate	Graph CNN		Transductive Learning	
	Accuracy (%)	Relative Error	Accuracy (%)	Relative Error
105	61.56 ± 19.56	1.151 ± 0.418	84.67 ± 19.58	0.502 ± 0.553
107	69.25 ± 19.01	0.988 ± 0.409	86.15 ± 14.56	0.448 ± 0.510
108	59.20 ± 19.84	1.178 ± 0.388	82.29 ± 21.04	0.520 ± 0.530
109	75.92 ± 21.11	0.831 ± 0.457	75.55 ± 19.21	0.817 ± 0.493
110	72.05 ± 18.78	0.911 ± 0.397	72.35 ± 21.16	0.859 ± 0.489
111	71.00 ± 22.23	0.899 ± 0.477	72.40 ± 19.24	0.866 ± 0.464
112	72.40 ± 18.81	0.946 ± 0.381	74.52 ± 20.12	0.866 ± 0.444
113	75.51 ± 20.27	0.841 ± 0.449	76.79 ± 17.64	0.803 ± 0.454
114	76.50 ± 22.72	0.753 ± 0.542	81.42 ± 20.73	0.588 ± 0.551
115	78.76 ± 20.75	0.717 ± 0.528	81.55 ± 19.33	0.660 ± 0.487

Table II
ACCURACY AND RELATIVE ERROR FOR EACH SENATE USING THE GRAPH CNN AND THE TRANSDUCTIVE LEARNING METHODS.

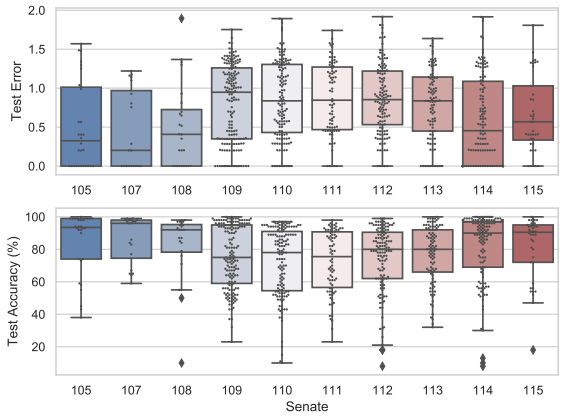

Figure 7. Results for the variation minimization method.

characterize the Senate. From a first exploratory data analysis, it was possible to perceive how polarized the US Senate is into two main groups, which can be interpreted *grosso modo* as a conservative and a progressive one. Another interesting lesson learned was the fact that, as a certain party constitutes the chamber minority, it becomes more prone to disagreeing within itself. Finally two learning methods were considered for predicting vote outcomes for each bill-related vote based on the bill's sponsorship profile. The first one, based on convolutional neural networks for graphs, provided good accuracy results especially when the dataset has a larger number of samples. The second one, based on transductive learning and graph interpolation, displayed a greater robustness to dataset size, being more accurate for Senate versions with fewer votes available for training. Overall the transductive learning method resulted in high accuracy scores, attaining up to 86% for the test set accuracy, while the CNN approach resulted in at most 78%, with standard deviations of approximately 20% for both cases. Possible extensions of this work include the optimization of the CNN parameters, and the application of such methods for analyzing multipartisan political environments such as the Swiss parliament. Besides, it would be interesting to consider the role of the House of Representatives in the bill-passing process and to eventually extend the model to general voting records, i.e. not necessarily tied to a bill.

REFERENCES

[1] J. H. Fowler, "Legislative cosponsorship networks in the us house and senate," *Social Networks*, vol. 28, no. 4, pp. 454–465, 2006.

[2] J. E. Jackson, "Statistical models of senate roll call voting," *American Political Science Review*, vol. 65, no. 2, pp. 451–470, 1971.

[3] A. Bonica, "Inferring roll-call scores from campaign contributions using supervised machine learning," *American Journal of Political Science*, vol. 62, no. 4, pp. 830–848, 2018.

[4] ProPublica, "ProPublica Congress API," https://projects.propublica.org/api-docs/congress-api/, accessed: 18.01.2019.

[5] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," *arXiv preprint arXiv:1312.6203*, 2013.

[6] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Advances in Neural Information Processing Systems*, 2016. [Online]. Available: https://arxiv.org/abs/1606.09375

[7] M. Herbster and G. Lever, "Predicting the labelling of a graph via minimum p-seminorm interpolation," in *NIPS Workshop 2010: Networks Across Disciplines: Theory and Applications*, 2009.

[8] S. Chen, A. Sandryhaila, J. M. Moura, and J. Kovacevic, "Signal recovery on graphs: Variation minimization." *IEEE Trans. Signal Processing*, vol. 63, no. 17, pp. 4609–4624, 2015.