Vote prediction of US Senators from graph properties

Mathieu Lamiot Julien Heitmann Louis Landelle Mathias Gonçalves

Abstract— **The purpose of this project is to create a classifier which can predict from graph properties whether a senator of the United State Senate will vote in favor or against a new law proposal. The graphs used were a co-voting graph, a co-sponsorship graph and a committees co-membership graph. The methods used in order to extract data were the laplacian eigenmap, the shortest paths between senator in the graph, plus some other distance computation algorithms. A logistic regression classifier and a random forest were implemented and the random forest classifier turned out to have the best accuracy of 81% against 73% for the logistic regression classifier.**

I. INTRODUCTION

The United States Senate is the upper chamber of the United States Congress which comprises the legislature of the United States along with the House of Representative (the lower chamber). The Senate is composed of 100 senators which of whom represent a particular state. There are exactly 2 senators per states so the number of senators per state is not proportional to the actual population. The main role of the senators is to vote some law proposals that are often proposed by the House of Representatives. The senators often have a party which is usually either the Republican (53%) or the Democratic (45%) party and more rarely they can be independent (2%).

The Senate also has an internal structure which are the committees. Theses committees are groups of senators that are in charge of a specific department such as Agriculture, Nutrition and Forestry, Homeland Security and Governmental Affairs or Finance for example. Those committees were made in order to have only a few senators digging on specific laws instead of all the senators having to work on all the different topics. As there are many laws they cannot put a lot of time studying each of them. That way each committee can be a voting reference for a specific bill. Usually, there is a pretty even number of democrats and republicans in each committee. There are 16 such committees each of which contain sub-committees but the latter won't be looked at in this project. The committees have exactly one chairman and some members, and the senators can be part of many different committees. The co-membership in committees will be used to create a feature for the classifiers.

Most of the time the bills that are voted are sponsored by some senators. Two senators that sponsored a specific bill are likely to have sponsored other bills together. So from the co-sponsorship data the senators will be more or less linked together depending on the number of bills they pairwise cosponsored.

II. METHODS

The objective of this project was to create a classifier which can predict whether a senator is going to vote in favor of a new law proposal or against it. All the data used in this project comes from the ProPublica Congress API [Pro,]. So for reproducibility of this project, one should get an API key on the Propublica website. The data set was created such that there is one sample per senator-bill pair and for each of those pairs, some features are added from the graphs properties. The features will be detailed in section II-B.

A. Graphs

1) Co-voting graph: The first one is the co-voting graph. It is the same that was build in the milestones during the semester. It is based on the percentage of common vote between two senators. The nodes of the graph are the senators and two nodes are connected if the two corresponding senators agree on more than 50% of the bill i.e. they vote the same for more than a given percentage of laws. The graph will then be undirected. In order to do so, we retrieved the votes of each senators and we do a vote-by-vote comparison. We obtain a symmetric matrix containing agreement percentages. We then apply the following formula to convert those "*distances*" into "*weights*":

$$weights_{i,j} = \exp\left(\frac{-distances^2}{M^2}\right) \tag{1}$$

where M is the mean value of $distances$. We then threshold the obtained matrix to create the adjacency matrix of this graph. Fig. 1 represents this graph with a "spring layout".

2) Co-sponsorship graph: The second graph used in this project is the co-sponsorship graph. The network was constructed this way: For every bill that was voted since the 111th congress, the list of co-sponsors was obtained using the API. The weight of the link between two senators is defined by the Jaccard index [Jac,] which is the ratio between the cardinality of the intersection and the carnality of the union of the sponsored bills of each senator. The Jaccard index is a way of computing how similar two sets are. Then the final weight of the graph is computed with the following formula, similar to the one used in the previous graph:

$$weight_{i,j} = \exp\left(\frac{-jaccard\ index^2}{M^2}\right) \tag{2}$$

where i, j are two different senator and M is the mean off all the Jaccard index coefficients. Using this equation the weights of the graph don't depend on the number of bill a certain senator cosponsored. The graph is as well undirected. Hence, in this graph, senators are strongly linked when they

Fig. 1. Co-voting graph

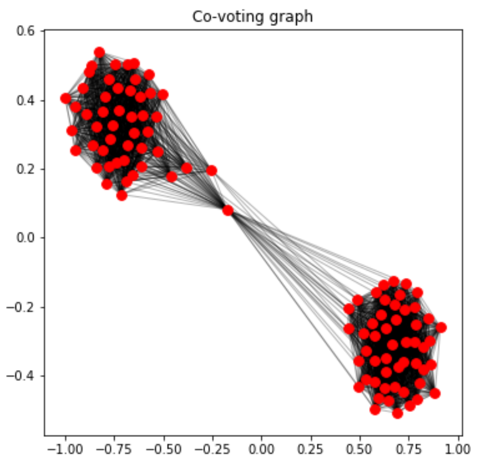
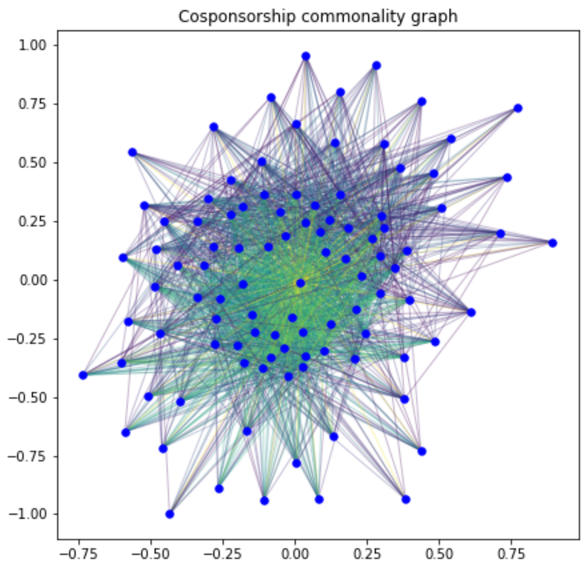


Fig. 2. Co-sponsorship Graph



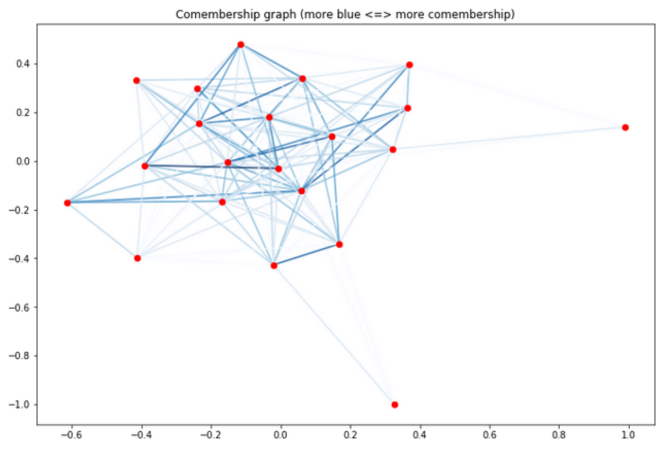
sponsored many laws in common, and they are weakly linked if they sponsored different laws. But further we wanted that every link that is under a certain threshold is set to 0 because we consider that weakly linked senators are not linked at all. Fig. 2 shows the resulting graph.

3) Committees graph: Here the fact that a senator can be part of multiple committees was used in order to build a committee-closeness graph, which would allow us to compute closeness metrics between a vote's bill committees and a senator's committees, which might give information on the vote direction of this senator on this vote.

The graph can be constructed using the following method:

- Each senate committee is represented as a node.

Fig. 3. Committee co-membership Graph



- Select two distinct committees, and their members M_{c1} and M_{c2}. We score their co-membership via:

$$C(c1,c2) = |M_{c1} \cap M_{c2}|$$

which is then normalized by dividing this score by the largest one over the graph:

$$\hat{C}(c1,c2) = \frac{C(c1,c2)}{\max_{c1,c2} C(c1,c2)}$$

This score is in $[0;1]$.
- Threshold this score by a constant for the final edges weights:

$$adj_{i,j} = \mathbb{1}\{\hat{C}(c1,c2) > k\}\hat{C}(c1,c2)$$

where $\mathbb{1}$ is the indicator function. We used $k = 0.3$ in this project. Fig. 3 shows the resulting graph.

B. Features of the classifiers

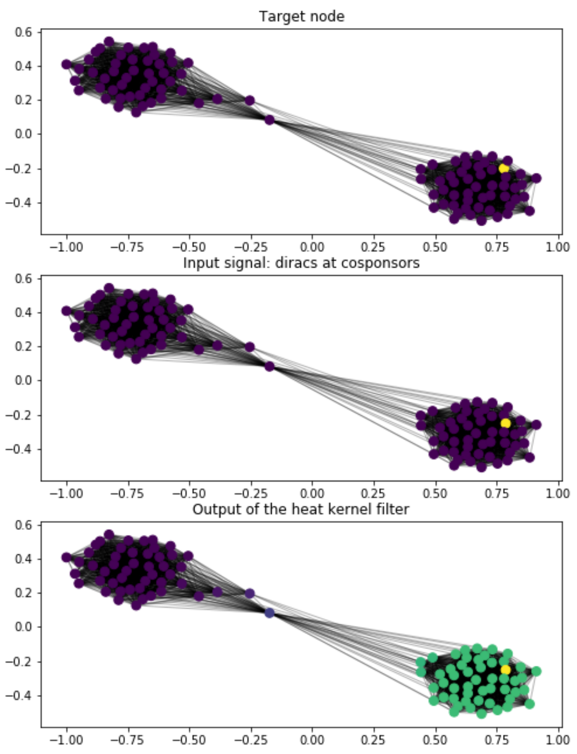
Various features were derived from the previous graphs in order to provide data to the classifiers.

1) Distances for co-voting graph and co-sponsorship graph: A basic way of exploiting the graphs is to compute distances between nodes. For instance, the distance between the senator we are studying and the senators that introduced or sponsored a law seems relevant to predict this senator's vote on that particular vote. Hence, we looked a bit more into that direction.

The first distance we used is the shortest path distance, which is the smallest sum of weights of edges to follow to get from one senator to the other. This is one of the most straightforward way we could think of to exploit the graphs and retrieve a feature linked to how influential a senator can be on another one. In the case of multiple cosponsors, the distance we use is the shortest shortest path between the senator we study and all the cosponsors.

However, the previous distance we presented has a major drawback, especially in highly connected networks, such as the ones we have: the shortest path only capture "one single way" on an information to be transmitted from one senator to

Fig. 4. Heat-kernel-based distance for the co-voting graph



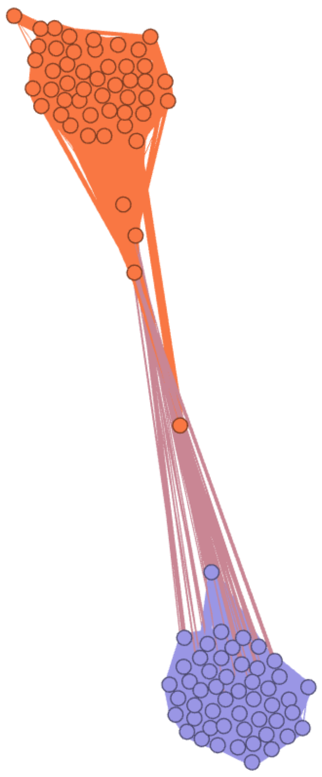
another: it doesn't make much difference if a senator knows one or many cosponsors for instance. But in reality, chances are that the more cosponsors a senator knows, the more likely he is to vote in favor of that law. To capture that phenomenon, we implemented a new way of measuring distances between one senator and a set of senators. This new "distance" is based on the principle of diffusion. We can interpret it as the time it takes for an information be diffused from the initial set to the senator we are studying. In order to estimate that time, we generate a signal on the graph as a sum of diracs on each senator from the inital set (set of cosponsors for instance). Then, we filter this signal with a heat kernel of parameter t, with a multiplication in the graph Fourier domain.

$$filter(e,t) = \exp(-e*t) \tag{3}$$

The time we are looking for is the smallest parameter t such that the filtered signal is strong enough on the node representing the senator we are studying (according to a predefined threshold). Fig. 4 presents an example.

Note that the initial set of senators is consider as a "constant source of heat" in this model, and hence, the signal is maintained at the maximum value at each iteration on those nodes.

Fig. 5. Parties over the co-vote graph

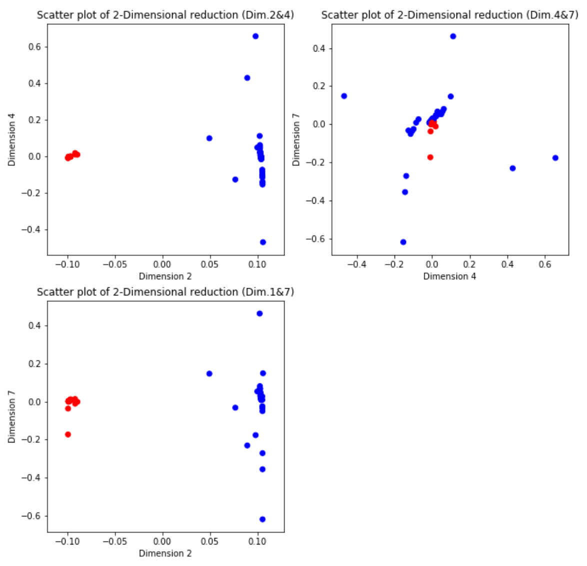


2) Parties as the 2nd Laplacian eigenvector for co-voting graph: As we can see on the co-vote graph, two highly intra-connected clusters appear. We tried to color the nodes according to the parties (Democrats and Republicans) as represented in Fig. 5 As we can see, the clusters are exactly matching the parties. Hence, parties seems to be an important feature for our vote forecast. In order to go a little bit further than simply a binary feature, we decided to find a way to quantify how republican/democrat a senator is. As we have seen in milestone 3, the laplacian eigenvectors can allow one to extract clusters from the graph. Hence, we decided to see if any of the laplacian eigenvectors could help us in this quantification. Different eigenvectors have been studied alongside with labels representing the party of each senator. Those eigenvectors are plotted in Fig. 6

As a result, the 2nd laplacian eigenvector seemed to be the most relevant one to quantify how republican/democrat a senator is. Hence, we created a new feature as follows: In the database, laws are either democrats or republicans (based on who introduced them). Hence each republican (democrat) law was given a label as the mean value of the 2nd laplacian eigenvector of republican (democrat) senators. Then, the distance between a senator and a law is the difference between the label of the law and the value of the 2nd laplacian eigenvector of the senator.

This feature is more evolved than a binary one classifying democrats and republicans as it allows to capture the fact that

Fig. 6. Plot of eigenvectors 2, 4 and 7



some senators are more moderates than others for instance.

3) Law co-sponsorship: Even if a senator is not a sponsor of a certain bill, he might have co-sponsord some other bills with some other senators that are actually sponsoring the bill in question. So this feature measures for a certain senator and a certain bill how connected this senator is to the cosponsors of the bill based on the weights of the co-voting graph. Thus it is expected that if a senator has strong links with the sponsors of the bill, there is good chance that the senator is going to vote in favor of the bill in question. The feature is computed as the sum of the weights of the co-sponsorship graph between the senator in question and the sponsors of the bill:

$$Feature = \sum_{c \, \in \, cosponsors-id} w_{s,c} \tag{4}$$

where s denote the id of the senator in question and $w_{i,j}$ the the weight between senator i and j in the graph.

4) Committee co-membership: Using the graph shown in Fig. 3, we create a new feature. Each vote is associated to a bill, and bills are often associated to a committee. Unfortunately, a large amount of bills are sponsored by house committees, which is outside of the scope of this project. Nevertheless, the senate committee originated bills can be placed on the committee graph, and the distance between the bill and each senator committees can give us information about how a vote might go. For a senator s in committees C_s and a vote v for a bill from senate committees C_b, we define their min_dist as:

$$min\_dist(s,v) = \min\{shortest\_path(bc,sc), \forall sc \in C_s, \forall bc \in C_b\}$$

Note:

- $min\_dist(s,v) = 0$ can occur when s is in the committee originating the bill.
- Let
$$M = 2 \max_{\forall s \in senators, \forall v \in votes} min\_dist(s,v)$$
This is chosen as distance when the (s,v) has no valid min_dist, because infinite value would not be compatible with regression.
- Thus, $C_s = \emptyset \implies min\_dist(s,v) = M, \forall v \in votes$. This case happens for two senators in congress 115.
- Likewise, $C_b = \emptyset \implies min\_dist(s,v) = M, \forall s \in senators$. This case happens for house committee bills, or orphan bills.

The feature is then, for a senator s and a vote v:

$$Feature = min\_dist(s,v) \tag{5}$$

Lastly in order to generate all the features, it is required that at least one senator is a sponsor of the bill and that the bill is affiliated to a party.

C. Classifiers

Two different classifiers were implemented and compared in order to get the most accurate predictions. As classifiers are not directly in the scope of this course, we will only provide a short description of the two classifiers that we tried. No specific efforts were dedicated to optimizing those classifiers as the goal of the project was more oriented toward extracting features from the graphs.

The first classifier was the logistic regression classifier which tries to predict the vote of a senator from a linear model which thresholds the hyperplan that fits the labels the best in order to decide if the prediction is in favor or against the bill.

The second classifier was a random forest classifier. The random forest algorithm is an algorithm that tries to create the best decision tree in order to predict whether a senator is going to vote in favor or against a specific law.

III. RESULTS AND DISCUSSION

From plotting the co-voting graph, one could observe that there are two distinct communities (in this case the two communities are the republicans and the democrats). And after doing a laplacian eigenmap of the graph, we found that the best eigenvector we can use is the the second one. The distance between the senators and the cosponsors of the bill were also computed from this graph. For the co-sponsorship graph, no particular structure was found. But the distance between a senator and the co-sponsors was still possible to be used in order to build a feature for the classifier. And lastly, for the committee co-membership graph, we are able to see how close a senator is from the committee proposing the bill and this distance can be used as a feature for the classifier.

Now for the classifiers, the random forest classifier gave a result of 81% of correct prediction while the logistic regression classifier gave a result of only 73% of correct prediction on the test set. This difference could be explained by the fact the data set is not linearly separable and that some features like the 2nd laplacian eigenmap are categorical features (which the random forest is supposed to perform the best on). A problem that came up was that only very few bill satisfied the requirement that a bill must have at least one sponsoring senator and must be affiliated to a party. Thus even though the result is good, the data set might have been too small to be confident about those results.

References

[Jac,] Jaccard index wikipedia page. https://en.wikipedia.org/wiki/Jaccard_index. Accessed: 2018-01-06.

[Pro,] Propublica congress api. https://projects.propublica.org/api-docs/congress-api/overview. Accessed: 2018-01-06.