AMERICAN UNIVERSITY OF BEIRUT

FACULTY OF ARTS & SCIENCES

Department of Computer Science

CMPS 262 – Data Science
Capstone 3 -” *Predicting Corruption Convictions Among Brazilian Representatives Through a Voting-History Based Network”*

By - Ralph Mouawad,
Alice Karadjian & Lea Bou Sleiman

To - Dr. Fatima Abu Salem

Please refer to the Colab file submitted on Moodle for the codes.

Part I: Selected Dataset -

We have selected the paper entitled “Predicting Corruption Convictions Among Brazilian Representatives through a Voting-History Based Network” for analysis”.

Part II: Explanation & Generation of the Dataset -

Let’s first define each step of how the work will be:

In our graph, each node is a legislator who has voted at least once and the edges between each pair is made according to voting similarity.

- The dataset will include 2455 congresspersons (2455 rows), and 3407 voting sessions (3407 columns). The entries of each column will be the type of vote of each legislator: “Yes”, “No”, “Obstruction”, “Abstention”. At each session, the maximum number of persons is 513, because this is the limited number of seats. This means that the maximum number of entries in the columns of our Dataset is 513. --> Each column (for each voting session) should have a maximum of 513 entries. The total number of votes is 1,656,547 votes in the report. However, when simulating the dataset, it couldn’t make it exactly 1,656,547 but it shouldn’t be a problem.

Now we will simulate our dataset:

- Each row is for one legislator, represented with an “ID” number going from 1 to 2455.
- Each column is for one session, going from 1 to 3407.
- The entry at each column is either “Yes’, “No’, “Obstruction”, “Abstention”, or an empty value if the maximum number of legislators is reached.

For time complexity, we will simulate a smaller dataset to be able to compute the matrices and perform all the calculations needed. We will simulate the dataset to be 400x150 (400 legislators and 150 voting sessions). We also imposed to have 35 rows to be similar, to be able to replicate the presence of corruption. We will also choose 4 of them to be already convicted and we’ll identify the neighbors and use link prediction techniques, such as ‘Cosine’.
Please refer to the Python collab to see the generation of our dataset.
We didn’t include names, or what type of bill was discussed at each session because it was difficult to generate the dataset because of its large size. We only included an ID number, and the number of each session. We then computed the

different matrices – Please check the codes for that. In the next part, we will explain how to do matrices to see the similarities between the legislators.

Part III: Implementation of the Solution:

The first part in the implementation of the solution is to get the matrices defined below, before implementing link predictions.

- For each voting session (for each column), we will have a matrix $M^{(t)}$ where each entry $M_{(i,j)}$ = (1 if 'i' & 'j' voted similarly | -1 otherwise). The maximum size of this matrix is 400x400 since this is the maximum number of legislators at each session in our dataset. The column entries will be set to 0. This matrix "M" will display the people who voted similarly at one voting session (t).
- Next, we define the Weighted Matrix $W^{(t)}$ which displays the following:

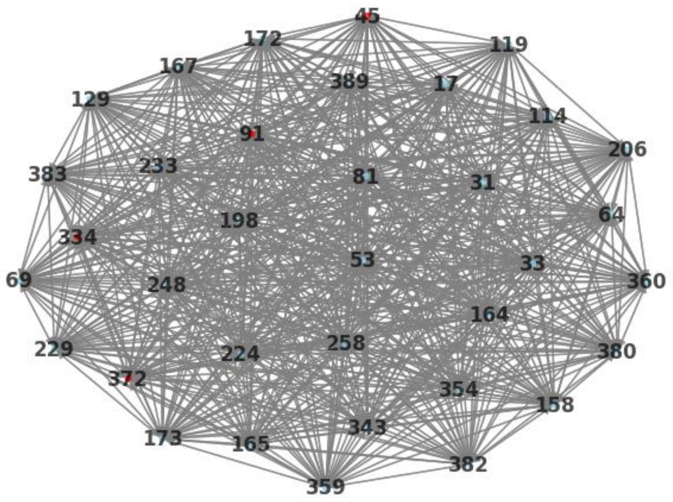
  $W^{(t)}_{(i,j)} = \Sigma\ M_{(i,j)}$ over 't'. This matrix will display the total number of times 'i' and 'j' voted similarly or differently up until time/session 't'. When we compute this matrix at session 150, we will have the voting similarity history between all legislators, and the size of the matrix will be 400x400 (the maximum number of legislators participating in the votes). The diagonal entries will remain 0. Each entry of the matrix will be in the range [-t,+t] or [-150, +150]: these are the extreme values: 150 means that all the entries of the "M" matrices were =1, meaning that at each session legislators 'i' and 'j' voted similarly.
- Finally, we define the $G^{(t)}$ matrix, obtained from the $W^{(t)}$ matrix at time 't', which will display the most important relationships between users up until time or session 't'. Moreover, let's define each entry of this matrix: $G_{(i,j)}^{(t)}$ = (1 if $W_{(i,j)}^{(t)}$ = max x for x in $W_{(i)}^{(t)}$ | 0 otherwise). This means that each row of the matrix "G" will display the most important relationship that user 'i' has with other legislators, based on the maximum number of similar votes computed from the matrix "W". We expect to have one entry at each row unless a legislator has equal similarities with more than one legislator. This is the step where we will be able to draw the edges for each node.

In the case of our code, we will have an "M" matrix for each column, and a Weight Matrix which is the summation of all "M" matrices defined. At the end the "G" matrix will display the nodes that have the most similarities.
All the matrices have a size of 400x400, which is the subset of our huge dataset.
After generating the matrices, the results were as following:

- The weighted matrix has many negative values, but a minority of positive and high values which could indicate corruption.
- The G matrix has some values = 1, which indicate the similarities.

Now let's move to the implementation of the graphs, which is the second part of the implementation of our solution.

- The graph that we have drawn has as a node the legislator ID 'i', and an edge drawn to another node 'j' if the entry $G_{(i,j)} = 1$. This means that there's a lot of similarities in the voting history between legislators 'i' and 'j'.
  The graph we obtained is the following:



The number of nodes in this graph is equal to 35, which is very logic since we simulated our dataset to have 35 similar rows (to create a corruption case), so our codes were able to catch all the similarities of the legislators and represent them as nodes and edges in a graph. The red-coloured nodes are those already convicted legislators.

The third part of our solution is to test the link prediction model. For that, we will do the same steps done in the paper:

1. "Create a subgraph from the built legislators' network resulting from matrix W(t), comprising only convicted congresspeople and their respective neighbors, from both incoming and outgoing edges. (*Done already above)*

2. Convert the network resulting from the subgraph to an undirected one and delete all existing links between convicted labeled nodes.
3. Apply the link prediction technique on the network.
4. Take the top n link predictions whose source node is convicted and label their target nodes as convicted ones as well".

We will execute the following steps and implement the 'Cosine" solution used in the paper, and the results obtained will be explained in the next part. Please refer to the python code to see the implementation of the above steps.

<u>Part IV – Evaluation of the Results:</u>

Let's first recap the results of the generation of the dataset, matrices & graphs:

- We generated the dataset with 35 similar rows, to replicate the presence of corruption (vote similarities).
- We computed the "M" matrix for each column (session) displaying the similarities at each voting session.
- We summed all the generated "M" matrices to get the Weighted Matrix "W" displaying the total number of similar votes up until the 150th session of voting. The values were in the interval [-150,150].
- We computed the "G" matrix that must display the maximum similarity for each legislator, and it takes the maximum value of each row of the weighted matrix to evaluate which one is the most similar. An entry of '1' is for representing that highest similarity. We obtained a certain number of '1' which indicates the eventual existence of similarities, which is logical because we imposed 35 similar rows at first.
- We generated a graph from the "G" matrix, where each node is a legislator who has similarities with another one. The total number of nodes was 35, which is consistent because this is the number of similar rows (or similar votes) that we imposed first when generating the dataset.

Now, we will talk about the implementation of the cosine similarity:

- We consider 4 nodes to be already convicted congresspersons. We made the subgraph undirected and removed the links between the convicted labeled nodes. Before, we saved the 31 neighbors in a data frame with a label 'convicted' to compare it with our predictions.
- We checked the similarity of each of the 31 neighbors with the 4 already convicted nodes using the rows of the Weighted Matrix, using the 'Cosine'

technique used in the paper; Each time we get a good cosine similarity (threshold = 0.8) for one of the 31 nodes, we connect it with an edge to the already convicted nodes. At the end, we saved the new neighbors in a new data frame and when the 2 data frames were compared, we had an 87% accuracy.
- In the paper, when this method was implemented, an accuracy of around 85% was reached, which is very close to our results, indicating that our work is very consistent.

<u>**Conclusion:**</u>

We can conclude that using link prediction techniques, such as the 'Cosine' technique, to detect corruption proved to be effective: We measured the cosine similarity between the rows to check which legislators voted very similarly. We got an accuracy of 87%, which is very good. Maybe if we decreased the cosine score threshold, we would have been able to reach a higher accuracy.

We included in our submission the following:

- ***The Python Collab file containing the codes that we used.***
- ***The simulated dataset with the different votes of legislators at each session (400x150).***
- ***The weighted matrix that we computed.***
- ***The 'G' matrix we computed to represent our graphs.***