Universidade Federal de Santa Catarina
Centro Tecnológico
Departamento de Informática e Estatística

Mateus Arns Kreuch

Desenvolvimento de solução para visualização e análise de votações políticas

Florianópolis, Brasil

2024

Universidade Federal de Santa Catarina
Centro Tecnológico
Departamento de Informática e Estatística

Mateus Arns Kreuch

Desenvolvimento de solução para visualização e análise de votações políticas

Trabalho de Conclusão do Curso de Graduação em Ciências da Computação da Universidade Federal de Santa Catarina, requisito parcial à obtenção do título de Bacharel em Ciências da Computação.

Orientador: José Eduardo de Lucca
Coorientador: Andre Wust Zibetti

Florianópolis, Brasil
2024

Resumo

Este trabalho visa desenvolver uma solução extensível para a visualização e análise das votações nominais da Câmara dos Deputados e do Senado Federal no Brasil, onde os votos de cada parlamentar são registrados individualmente e divulgados publicamente. Utilizando técnicas estatísticas avançadas, como redução de dimensionalidade e algoritmos de clusterização, o objetivo é criar modelos empíricos que posicionam os legisladores no espectro político de maneira quantitativa, facilitando a compreensão das relações entre partidos, identificação de coalizões e antecipação de tendências políticas. Ao facilitar uma melhor compreensão das mecânicas políticas, espera-se que este trabalho promova transparência e fortalecimento democrático.

Palavras-chave: análise de dados. aprendizado de máquina. ciência política.

Abstract

This work aims to develop an extensible solution for the visualization and analysis of roll-call data in the Brazilian Chamber of Deputies and Federal Senate, where the votes of each legislator are individually recorded and publicly disclosed. Using advanced statistical techniques, such as dimensionality reduction and clustering algorithms, the goal is to create empirical models that positions legislators on the political spectrum in a quantitative manner, facilitating the understanding of relationships between parties, the identification of coalitions, and the anticipation of political trends. By fostering a better understanding of political mechanisms, this work is expected to promote and strengthen transparency and democracy.

Keywords: data analysis. machine learning. political science.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

API	Application Programming Interface
PCA	Principal Component Analysis
MDS	Multidimensional Scaling
UMAP	Uniform Manifold Approximation and Projection
t-SNE	t-Distributed Stochastic Neighbor Embedding
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
HDBSCAN	Hierarchical DBSCAN

Sumário

1 Introdução

No Brasil, a Câmara dos Deputados e o Senado Federal empregam vários tipos de votações, que são utilizadas para deliberar sobre diferentes matérias. Em especial, se destaca a votação nominal, onde, diferentemente de outros tipos de votação, os votos de cada parlamentar são registrados individualmente, e o resultado é divulgado publicamente. A votação nominal é obrigatória em matérias de maior relevância, como propostas de emenda à Constituição, vetos presidenciais, e pedidos de urgência. Além disso, os portais da Câmara dos Deputados[1] e do Senado Federal[2] disponibilizam online os dados das votações nominais em diversos formatos.

Um sumário de determinado período histórico pode ser construido ao organizar os dados das votações nominais em uma matriz de votos (BRIGADIR et al., 2016), onde cada célula leva o valor do voto. As linhas representam legisladores e as colunas representam propostas em diferentes níveis de andamento. No entanto, devido à alta dimensionalidade dessa estrutura, é necessário o uso de técnicas estatísticas para extrair informações úteis para análise política.

Assim, é possível utilizar esta matriz para criar modelos empíricos, chamados de estimadores de ponto ideal (*ideal point estimators*), que representam o posicionamento dos legisladores em relação uns aos outros, efetivamente situando-os no espectro político (POOLE; ROSENTHAL, 1985; SILVA; SPRITZER; FREITAS, 2018). Esses modelos são fundamentais para compreender as relações entre partidos, identificar coalizões e antecipar tendências políticas e ideológicas, ou, alternativamente, permitem reconhecer partes disfuncionais da organização governamental, possibilitando uma análise mais profunda e precisa do comportamento político.

Entre as técnicas capazes de construir estimadores de ponto ideal está a redução de dimensionalidade (BRIGADIR et al., 2016), técnica aplicada no domínio da inteligência artificial que permite capturar as estruturas subjacentes dos dados originais e representá-los em um espaço de dimensão inferior, facilitando sua compreensão. De maneira complementar, algoritmos de clusterização, que visam fazer agrupamentos automáticos de dados segundo o seu grau de semelhança, permitem identificar legisladores com visões políticas similares (LEE; ZHANG; YANG, 2017).

De fato, a utilização da redução de dimensionalidade na análise política não é novidade no resto do mundo, remontando a 1985 com a publicação da família de algoritmos NOMINATE (POOLE; ROSENTHAL, 1985). No entanto, a transparência governamental

[1] https://dadosabertos.camara.leg.br
[2] https://www12.senado.leg.br/dados-abertos/

que possibilita esse tipo de pesquisa no Brasil é recente (HAGOPIAN; MAINWARING, 2005). Consequentemente, muitos dos projetos existentes contêm lacunas, como a falta de dados sobre o Senado Federal e a ausência de uma dimensão esquerda-direita na análise (SILVA; SPRITZER; FREITAS, 2018). Além disso, a maioria dos projetos foca no público geral, e não necessariamente em pesquisadores. Isto é, possuem interfaces para visualizar informações e, até certo ponto, controlar o sistema desenvolvido, mas não possibilitam a extensão e aplicação de outros algoritmos em cima dos modelos gerados.

Assim, este trabalho tem como objetivo o desenvolvimento de uma solução para a visualização e análise das votações nominais da Câmara dos Deputados e do Senado Federal, de fácil extensão e que visa complementar os trabalhos existentes. Serão exploradas diversas técnicas de clusterização e redução de dimensionalidade, avaliando sua aplicabilidade no que se refere a fornecer uma maior compreensão sobre os processos de tomada de decisão coletiva e as dinâmicas políticas envolvidas.

1.1 Objetivos

1.1.1 Objetivo geral

Desenvolver uma solução flexível e de fácil utilização para visualização e análise de votações nominais, proporcionando uma ferramenta acessível para pesquisadores da área da ciência política, legisladores e cidadãos, com o intuito de facilitar o entendimento e a compreensão dos processos democráticos, além de promover a transparência no sistema político.

1.1.2 Objetivos específicos

Os objetivos específicos deste trabalho são:

O1. Pesquisar as vantagens e desvantagens dos diversos algoritmos de análise de dados quando inseridos no contexto político.

O2. Desenvolver uma solução para a visualização e análise de votações nominais.

O3. Validar a solução através de dados reais de diferentes contextos políticos a fim de garantir precisão analítica e eficiência amostral.

1.2 Metodologia

Para alcançar os resultados desejados neste trabalho, será adotada uma abordagem que combina diversas metodologias de pesquisa, adequadas aos objetivos específicos do projeto. As etapas são conduzidas da seguinte maneira:

Etapa 1 – Elaboração da fundamentação teórica: Nesta etapa será realizado a análise e síntese da literatura referente a estudos acadêmicos e fontes confiáveis, que abordam temas como análise de dados e aprendizado de máquina, estatística, desenvolvimento de software, ciência política, entre outros. Este estudo é feito por meio de uma análise e síntese da literatura.

Atividade 1.1: Sintetizar conceitos do sistema político brasileiro.

Atividade 1.2: Sintetizar conceitos de análise de dados e aprendizado de máquina.

Atividade 1.3: Sintetizar conceitos estatísticos.

Atividade 1.4: Sintetizar conceitos de ciência política.

Etapa 2 – Levantamento do estado da arte: levantamento sobre trabalhos existentes relacionados à área do projeto. É realizado um estudo de mapeamento e análise de diferentes aspectos, como as técnicas de visualização de dados utilizados, os tipos de dados políticos analisados, as métricas e indicadores de análise empregados, as plataformas e ferramentas desenvolvidas, bem como os resultados e impactos alcançados.

Atividade 2.1: Definir o protocolo de busca;

Atividade 2.2: Executar a busca;

Atividade 2.3: Extrair e analisar as informações.

Etapa 3 – Desenvolvimento da solução: nesta etapa será realizado o desenvolvimento de toda a solução proposta.

Atividade 3.1: Modelar a proposta de solução.

Atividade 3.2: Desenvolver a proposta de solução.

Atividade 3.3: Validar a proposta de solução.

2 Fundamentação

Neste capítulo, é apresentada uma breve introdução a temas importantes para contextualizar e fornecer o devido embasamento ao trabalho. Aborda-se de maneira concisa o sistema político brasileiro e os aspectos das votações nominais na Câmara dos Deputados e no Senado Federal, além de apresentar conceitos sobre análise de votações nominais, incluindo técnicas de redução de dimensionalidade e clusterização.

2.1 Sistema político brasileiro

De acordo com a Constituição Federal de 1988 (BRASIL, 2023), o sistema político brasileiro é caracterizado como uma república federativa democrática, onde o poder é exercido pelo povo, por meio de representantes eleitos, e organizado em três poderes independentes e harmônicos entre si: Executivo, Legislativo e Judiciário. Abaixo, é apresentada uma visão geral de cada um desses poderes:

- **Poder Executivo**: É chefiado pelo Presidente da República, eleito pelo voto direto para um mandato de quatro anos, com a possibilidade de reeleição por mais um mandato consecutivo. O Presidente é o chefe de Estado e chefe de governo, responsável por administrar o país, formular políticas públicas, representar o Brasil interna e externamente, além de comandar as Forças Armadas. Auxiliando o Presidente, há diversos Ministérios e órgãos governamentais responsáveis pela implementação das políticas públicas em áreas específicas.

- **Poder Legislativo**: É bicameral, composto pela Câmara dos Deputados e pelo Senado Federal. Os deputados federais, compostos por 513 representantes, são eleitos por meio do sistema proporcional, representando a população dos estados e do Distrito Federal durante um mandato de quatro anos. Já os senadores são eleitos pelo sistema majoritário para um mandato de oito anos, três representantes por estado, totalizando 81 membros. O Congresso Nacional é responsável por elaborar e aprovar leis, fiscalizar o Executivo, além de outras atribuições constitucionais. Destaca-se que o Congresso exerce um papel fundamental no sistema político brasileiro, sendo um espaço de debate e deliberação sobre os principais temas de interesse nacional.

- **Poder Judiciário**: É encabeçado pelo Supremo Tribunal Federal (STF), órgão máximo do poder judiciário brasileiro, responsável por garantir a interpretação da Constituição Federal, além de ser a última instância para julgamento de questões constitucionais. Além do STF, o sistema judiciário brasileiro é composto por tribunais

superiores (STJ, TST, TSE), tribunais regionais e juízes de primeira instância, que atuam em diferentes esferas (federal, estadual e trabalhista), garantindo a aplicação da lei e a proteção dos direitos dos cidadãos.

Tanto o poder Legislativo como Executivo são governados pelo sistema eleitoral brasileiro, que é caracterizado por ser um sistema misto, onde coexistem elementos do sistema proporcional e do sistema majoritário. Isso significa que as eleições são realizadas tanto por meio do voto proporcional quanto pelo voto majoritário, dependendo do cargo eletivo em questão.

O voto proporcional é utilizado para eleger os representantes do Legislativo com exceção dos senadores, isto é, deputados estaduais e federais, além de vereadores. Nesse sistema, o eleitor vota no partido ou na coligação de partidos, e os candidatos mais votados dentro de cada partido ou coligação são eleitos de acordo com uma proporção estabelecida pelo quociente eleitoral, que leva em consideração o número total de votos válidos e o número de vagas disponíveis.

Por outro lado, o voto majoritário é utilizado para eleger senadores e os cargos do Executivo, como presidente, governadores e prefeitos. Nesse sistema, o candidato que obtiver a maioria simples dos votos é declarado vencedor.

O Brasil também adota o sistema de dois turnos para as eleições presidenciais e para as eleições para governador em estados onde nenhum candidato recebe mais de 50% dos votos válidos no primeiro turno. Nesse caso, os dois candidatos mais votados disputam um segundo turno, no qual é eleito aquele que obtiver a maioria dos votos válidos.

2.1.1 Votações nominais

O processo legislativo na Câmara dos Deputados e no Senado Federal segue uma série de etapas, de acordo com o Regimento Interno da respectiva casa legislativa[1], que incluem a apresentação de projetos de lei, sua análise por comissões temáticas, discussão em plenário e votação. A aprovação de uma proposta legislativa requer o apoio da maioria dos membros presentes em cada uma das casas, seguindo os trâmites regimentais estabelecidos.

Para isso, existem diferentes tipos de votações, cada uma com suas características e finalidades específicas, dentre as quais se destaca a votação nominal. No âmbito do processo legislativo brasileiro, a votação nominal representa um importante instrumento de transparência e responsabilização dos parlamentares perante a sociedade e seus eleitores.

A votação nominal consiste no registro individual do voto de cada parlamentar, sendo seus nomes e respectivas posições registrados de forma pública e oficial nos anais da casa legislativa. Esse método contrasta com outros tipos de votação, como a votação

[1] https://www.congressonacional.leg.br/legislacao-e-publicacoes/regimento-do-congresso-nacional

simbólica, onde não há a identificação individual dos votantes, dificultando a prestação de contas por parte dos representantes eleitos.

Além disso, a votação nominal possibilita o registro histórico das decisões tomadas pelo Legislativo, permitindo uma análise mais aprofundada do comportamento político dos parlamentares ao longo do tempo, além de possibilitar análises quantitativas e computacionais de períodos históricos.

Portanto, esse tipo de votação é empregada em decisões de especial relevância, como aquelas relacionadas a emendas constitucionais, projetos de lei de grande repercussão social ou econômica, e também em casos de cassação de mandatos parlamentares ou votações sobre questões éticas e disciplinares.

Os portais da Câmara dos Deputados[2] e do Senado Federal[3] disponibilizam os dados de votações nominais em diferentes tipos de arquivos (.csv, .json, .xml, etc) através de uma API pública, que pode ser utilizada por aplicações externas. Nesses dados se encontram os pares representante-voto de cada votação nominal já realizada (Apêndice A). Os formatos e possíveis valores de voto diferem entre as casas, mas os tipos mais comuns e importantes são:

- **Sim**: Indica que o parlamentar concorda com a proposta ou medida que está sendo votada. Este voto é a favor da aprovação do que está em pauta.
- **Não**: Indica que o parlamentar discorda da proposta ou medida que está sendo votada. Este voto é contra a aprovação do que está em pauta.
- **Abstenção**: Indica que o parlamentar opta por não votar nem a favor nem contra a proposta. A abstenção pode ser uma forma de expressar neutralidade ou indecisão sobre o tema em votação.
- **Obstrução**: É uma estratégia utilizada pelos parlamentares para impedir ou atrasar a votação de uma proposta. Quando um parlamentar vota pela obstrução, ele está tentando dificultar o andamento normal da votação, geralmente por considerar que a proposta não deve ser discutida naquele momento ou de forma a buscar tempo para articulações políticas.
- **Art. 17/Art. 51**: Refere-se ao voto do Presidente da Câmara/Senado, sendo um valor que difere entre as votações da Câmara e Senado, mas de mesmo significado. Faz referência ao Artigo 17 do Regimento Interno da Câmara dos Deputados[1] ou ao Artigo 51 do Regimento Interno do Senado Federal[1], que afirma que o Presidente não poderá votar, exceto no caso de escrutínio secreto ou para desempatar o resultado de votação ostensiva.

[2] https://dadosabertos.camara.leg.br
[3] https://www12.senado.leg.br/dados-abertos/

2.2 Redução de dimensionalidade

A redução de dimensionalidade é o processo de transformar dados de um espaço de alta dimensionalidade em um espaço de menor dimensionalidade, mantendo as características essenciais dos dados originais (MAATEN et al., 2009). Esta técnica têm sido fundamental para a ciência política, em especial na análise de dados de votações nominais, permitindo que os pesquisadores extraiam insights significativos destes conjuntos de dados de alta dimensionalidade (ABDULJABER, 2020).

Desde 1985, a família de algoritmos NOMINATE (POOLE; ROSENTHAL, 1985) e, mais tarde, outros métodos de redução de dimensionalidade (HOTELLING, 1933; TORGERSON, 1952; MAATEN; HINTON, 2008; MCINNES; HEALY; MELVILLE, 2018), têm sido amplamente utilizados para construir estimadores de ponto ideal, que representam as posições dos legisladores em relação uns aos outros em um modelo espacial, normalmente com base nos votos nominais. Estes modelos são fundamentais para entender e modelar o comportamento humano, especialmente quando se trata de preferências, opiniões e tomada de decisão.

O termo "estimador de ponto ideal" origina-se da ciência política e econômica, sobretudo da teoria espacial do voto (*spatial theory of voting*) (ENELOW; HINICH, 1984), que é uma abordagem teórica que examina o comportamento eleitoral a partir da perspectiva espacial, ou seja, considerando a posição dos eleitores e dos candidatos em um espectro ideológico ou de políticas públicas. Fundamentada em modelos matemáticos e estatísticos, essa teoria sugere que os eleitores escolhem candidatos ou partidos que estejam mais próximos de suas próprias preferências políticas em um espaço multidimensional de questões políticas.

Portanto, um estimador de ponto ideal faz uso de dados amostrais para calcular, de forma quantitativa, o *ponto ideal* (isto é, a melhor estimativa posicional) de atores em modelos espaciais políticos (CARROLL, 2023).

2.2.1 Escalonamento Multidimensional

O escalonamento multidimensional (MDS) (TORGERSON, 1952) é uma técnica estatística usada para visualizar o nível de similaridade ou dissimilaridade entre objetos em um conjunto de dados. O objetivo é representar as relações entre objetos em um espaço de dimensão inferior (portanto, é um tipo de redução de dimensionalidade), tipicamente 2D ou 3D, de maneira que possam ser facilmente visualizadas (BORG; GROENEN, 2007).

O MDS começa construindo uma matriz de distâncias representando distâncias entre pares de objetos. A matriz de distâncias é então duplamente centralizada (*double-centered*) para obter uma matriz de produtos internos (WICKELMAIER, 2003). É realizada a decomposição em autovalores dessa matriz centralizada para extrair os principais au-

tovalores e seus respectivos autovetores. As coordenadas dos objetos no novo espaço de dimensão inferior são então calculadas usando esses autovetores e autovalores.

Esta técnica já foi utilizada em diversos estudos de análise política. Em um estudo, o MDS foi aplicado às votações nominais de 2005 da Câmara dos Representantes dos Estados Unidos (DIACONIS; GOEL; HOLMES, 2008). Em outro, o escalonamento multidimensional ajudou a encontrar padrões de coalizão no Conselho da União Europeia (MATTILA; LANE, 2001). Além disso, foi usado para comparar seis países europeus a fim de analisar o impacto da globalização no espaço político (KRIESI et al., 2006).

2.2.2 Análise de Componentes Principais

A Análise de Componentes Principais (PCA) (HOTELLING, 1933) é uma técnica de redução de dimensionalidade para conjuntos de dados multivariados. Nela, os dados são transformados linearmente para encontrar uma sequência de vetores unitários, chamados de componentes principais, que melhor se ajustam aos dados enquanto são ortogonais entre si (JOLLIFFE; CADIMA, 2016). Esses componentes formam uma base ortonormal onde as diferentes dimensões dos dados são linearmente não correlacionadas.

Isto é realizado através do cálculo de autovetores e autovalores da matriz de covariância, onde os autovetores representam as direções de máxima variância e os autovalores indicam a quantidade de variância em cada componente. Projetando os dados nos novos eixos definidos pelos componentes principais, o PCA cria novos dados que capturam a maior parte da informação com menos dimensões do que o conjunto de dados original (JOLLIFFE, 2003).

O PCA é utilizado pelo projeto Radar Parlamentar, uma ferramenta brasileira que permite visualizar a aproximação de partidos políticos através dos dados de votações nominais (LEITE; TRENTO, 2016). Além disso, o PCA já foi empregado para avaliar a estrutura subjacente de sistemas partidários de 17 democracias avançadas de 1970 a 2013 (MAGYAR, 2022), para investigar padrões de votação em 84 distritos municipais na Escócia (LINCOLN; PIEPE; PRIOR, 1971) e para mensurar as leis eleitorais estaduais e restrições de voto nos Estados Unidos (CARMINES; STIMSON, 1982).

2.2.3 t-Distributed Stochastic Neighbor Embedding

O t-Distributed Stochastic Neighbor Embedding (t-SNE) (MAATEN; HINTON, 2008) é uma técnica de redução de dimensionalidade não-linear, o que o distingue de técnicas lineares como PCA. Essa capacidade torna o t-SNE particularmente adequado para conjuntos de dados complexos onde métodos lineares não são suficientes.

Outra característica do t-SNE é a preservação de vizinhanças locais. Pontos de dados que estão próximos no espaço de alta dimensionalidade permanecem próximos na

visualização de baixa dimensionalidade do t-SNE (ARORA; HU; KOTHARI, 2018). No entanto, como resultado, as relações globais, que envolvem distâncias mais longas, podem ser distorcidas ou completamente perdidas.

Pesquisadores da Universidade Erasmus de Roterdã usaram o t-SNE para visualizar semelhanças entre países com base em características como PIB, níveis de corrupção e liberdade econômica (GROENEN; CASTELEIN; BRUIN, 2020). Também foi aplicado às votações nominais do sexto e sétimo Parlamento Europeu (BRIGADIR et al., 2016), e uma de suas variantes foi utilizada para analisar a eleição parlamentar alemã de 2017 (BACHMANN; HENNIG; KOBAK, 2022).

Para um conjunto de dados de n elementos, o t-SNE tem uma complexidade de tempo e de espaço de $O(n^2)$ (PEZZOTTI et al., 2016) e compreende duas etapas principais. Ele começa construindo uma distribuição de probabilidade sobre os pares de objetos no espaço de alta dimensionalidade de maneira que objetos semelhantes sejam atribuídos uma probabilidade maior, enquanto objetos dissimilares recebem uma probabilidade menor. Em seguida, o t-SNE constrói uma distribuição de probabilidade similar para os objetos no mapa de baixa dimensionalidade e minimiza a divergência de Kullback-Leibler (KULLBACK; LEIBLER, 1951) entre as duas distribuições com relação às localizações dos objetos no mapa.

2.2.4 Uniform Manifold Approximation and Projection

O Uniform Manifold Approximation and Projection (UMAP) (MCINNES; HEALY; MELVILLE, 2018), assim como o t-SNE, é um algoritmo de redução de dimensionalidade não-linear. No entanto, o UMAP visa capturar tanto as relações locais dentro das vizinhanças quanto as relações globais entre vizinhanças distintas, o que confere uma preservação da estrutura original mais balanceada.

Ele opera através de várias etapas-chave para alcançar essa redução. Primeiramente, o UMAP constrói uma representação de grafo ponderado dos dados, onde cada ponto de dados é um nó e arestas conectam pontos próximos uns aos outros no espaço de alta dimensionalidade (MCINNES; HEALY; MELVILLE, 2018). O peso de cada aresta reflete a similaridade ou distância entre os pontos conectados.

Após a construção do grafo da vizinhança, o UMAP aproxima uma estrutura topológica difusa dos dados e em seguida otimiza a representação de baixa dimensionalidade minimizando a discrepância entre as estruturas topológicas difusas nos espaços de alta e baixa dimensionalidade, tipicamente usando técnicas como gradiente descendente estocástico (DIAZ-PAPKOVICH; ANDERSON-TROCMÉ; GRAVEL, 2021).

Apesar de ser um desenvolvimento relativamente recente, o UMAP já foi empregado para a modelagem e análise de padrões nos temas debatidos nas comissões permanentes

da Câmara dos Deputados do Brasil (SANTOS; ANDRADE; MORAIS, 2021). Além disso, foi utilizado para estudar a polarização política na Turquia (RASHED et al., 2021), para detecção de posicionamento em usuários do Twitter (DARWISH et al., 2020) e para prever o posicionamento político e de certos tópicos usando tweets (STEFANOV et al., 2020).

2.3 Clusterização

A clusterização é o processo de agrupar um conjunto de objetos de forma que os objetos dentro do mesmo grupo (ou cluster) apresentem maior similaridade entre si em comparação com aqueles em outros grupos. A análise de clusters tem sido usada no contexto político desde pelo menos a década de 1950, e é uma ferramenta valiosa para desenvolver tipologias objetivas e replicáveis para classificar fenômenos políticos (FILHO et al., 2014).

2.3.1 HDBSCAN

O Clustering Espacial Hierárquico Baseado em Densidade de Aplicações com Ruído (HDBSCAN) (CAMPELLO; MOULAVI; SANDER, 2013) é um algoritmo de clustering projetado para descobrir clusters em conjuntos de dados complexos. É uma extensão do algoritmo DBSCAN (Clustering Espacial Baseado em Densidade de Aplicações com Ruído) (ESTER et al., 1996), melhorando seu antecessor ao incorporar técnicas de clustering hierárquico para aumentar a flexibilidade e a precisão da identificação de clusters.

O HDBSCAN pode determinar automaticamente o número ideal de clusters sem a necessidade de especificação prévia (SCHUBERT et al., 2017), tornando-o altamente útil para uma análise exploratória quantitativa dos dados. Ele é particularmente hábil em identificar clusters que variam em densidade, forma e tamanho, incluindo clusters não convexos e irregulares, e em diferenciar entre clusters significativos e ruído.

O algoritmo utiliza a distância de alcançabilidade mútua (*mutual reachability distance*) entre os pontos, que representa quão denso uma área é, para criar uma árvore geradora mínima (MST) (STEWART; AL-KHASSAWENEH, 2022). Em seguida, é construída uma árvore hierárquica de clusters removendo recursivamente as arestas mais longas desta MST. A hierarquia é então condensada ao colapsar clusters insignificantes que não persistem, e clusters estáveis são extraídos com base na sua persistência em diferentes níveis de densidade.

2.3.2 Propagação por Afinidade

A Propagação por Afinidade (FREY; DUECK, 2007), assim como o HDBSCAN, é um algoritmo útil quando o número de clusters não é conhecido de antemão ou quando

os dados não se conformam a formas de clusters esféricas ou convexas. No entanto, a Propagação por Afinidade difere em como os clusters são definidos.

Esta técnica utiliza troca de mensagens entre pontos para identificar "exemplares" e atribuir pontos a esses exemplares automaticamente, criando clusters (DUECK, 2009). O algoritmo ajusta iterativamente as *responsabilidades* e *disponibilidades* entre pontos para determinar o número de clusters e a atribuição de pontos de dados a esses clusters. A *responsabilidade* foca na perspectiva individual de cada ponto, avaliando quão bem k se destaca como um candidato exemplar em comparação com outros candidatos. Já a *disponibilidade* leva em conta a perspectiva coletiva, avaliando quão fortemente a escolha de k como um exemplar é apoiada pelo restante dos pontos.

3 Trabalhos relacionados

Neste capítulo, são revisados os principais trabalhos relacionados à visualização de dados políticos. Foi feito um esforço para abranger a maior parte dos projetos brasileiros, além de incluir também alguns projetos internacionais relevantes. A maioria dos trabalhos analisados é de natureza acadêmica ou desenvolvidos por empresas do setor de comunicação.

3.1 Ferramentas qualitativas

Modelos espaciais, como mapas ideológicos, são ferramentas analíticas robustas na ciência política, utilizadas para compreender e visualizar as posições de atores políticos em um espaço multidimensional (CARROLL; POOLE, 2014). Esses modelos permitem a representação gráfica das preferências políticas, ideológicas ou programáticas de partidos, candidatos ou eleitores, facilitando a análise de suas interações e comportamentos e efetivamente os situando em um espectro político. A ideia subjacente é que as posições políticas podem ser mapeadas em um espaço ideológico, geralmente bidimensional.

Estes modelos começaram a ser utilizados com o desenvolvimento da teoria espacial do voto (ENELOW; HINICH, 1984). As primeiras abordagens utilizavam questionários e regressão estatística para construir tais modelos, mas a natureza qualitativa desses dados e a necessidade de modificações frequentes para adaptá-los a diferentes contextos políticos significava que os modelos continham uma quantidade significativa de viés (LESTER, 1994).

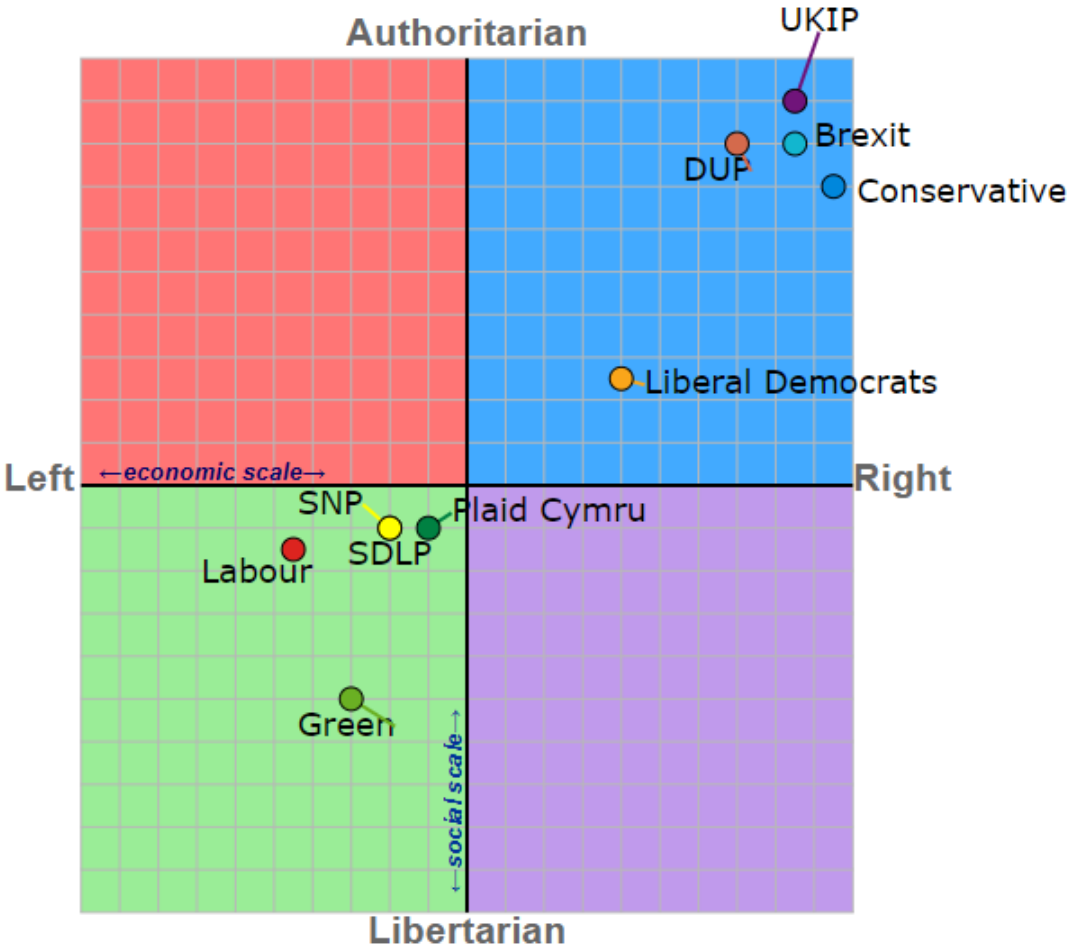
Existem ferramentas populares na internet que possibilitam usuários a se situarem no espectro político através dessa abordagem qualitativa, como o Political Compass[1], I Side With[2], Advocates for Self-Government[3] e On The Issues[4], entre outros. De maneira geral, essas ferramentas utilizam as respostas do usuário a um questionário sobre suas opiniões econômicas, sociais, religiosas, etc. para situá-lo ideologicamente. Algumas, como o Political Compass, fazem uso de um gráfico bidimensional, com um eixo horizontal que representa as opiniões econômicas (esquerda e direita) e um eixo vertical que representa as opiniões sociais (autoritarismo e libertarismo) para representar o posicionamento ideológico do usuário de maneira visual.

Estas ferramentas também realizam, periodicamente, a análise de certos cenários políticos, situando partidos ou pessoas influentes, com base em suas declarações publicas,

[1] https://www.politicalcompass.org/
[2] https://www.isidewith.com
[3] https://www.theadvocates.org
[4] https://www.ontheissues.org

no espectro político. Assim, é possível comparar o seu posicionamento não só com o posicionamento de outras pessoas que responderam o questionário, mas com o seu contexto político.

Figura 1 – Mapa ideológico do Political Compass



Mapa ideológico das eleições gerais de 2019 do Reino Unido do Political Compass. Há dois eixos, um econômico e um social. Quanto mais próximos dois grupos, mais parecidas suas opiniões no respectivo eixo. Fonte: Captura de tela do site <https://www.politicalcompass.org/uk2019>, acessado em 08 de junho de 2024

Posteriormente, as ferramentas de análise política começaram a adotar uma postura quantitativa, em geral através da análise de votações nominais. Dessa maneira, é possível utilizar informações de voto para realizar análises de coesão partidária, de alinhamento ao governo, de padrões de votação e de posicionamento ideológico, muitas vezes produzindo mapas similares ao do Political Compass, mas de maneira quantitativa.

3.2 Basômetro

O Basômetro (ESTADÃO, 2019) é um projeto de código aberto desenvolvido pelo jornal Estadão que tem como objetivo mensurar o grau de alinhamento dos deputados federais e partidos políticos em relação a diferentes governos. Para realizar essa mensuração, a ferramenta calcula a porcentagem de votos que seguem a orientação do líder do governo em cada votação nominal.

A metodologia do Basômetro considera um voto a favor do governo aquele que segue exatamente a orientação indicada pela liderança governamental. Por exemplo, se a orientação é votar "sim", apenas os votos "sim" são contabilizados como favoráveis ao governo. Todos os outros votos, incluindo "não", "obstrução" ou "abstenção", são considerados contrários ao governo.

Nos casos em que o governo não estabelece uma orientação explícita e libera a bancada para votar conforme sua preferência, esses dados são excluídos das análises. Dessa forma, as taxas de governismo e de assiduidade dos parlamentares são calculadas apenas com base nas votações em que o Executivo assumiu uma posição explícita.

O Basômetro permite visualizar tanto o alinhamento geral dos deputados em um determinado governo, como também o alinhamento de um deputado ou partido específico perante vários governos. No entanto, está disponível apenas visualizações sobre a Câmara dos Deputados, não incluindo o Senado Federal.

Figura 2 – Visualizações do Basômetro

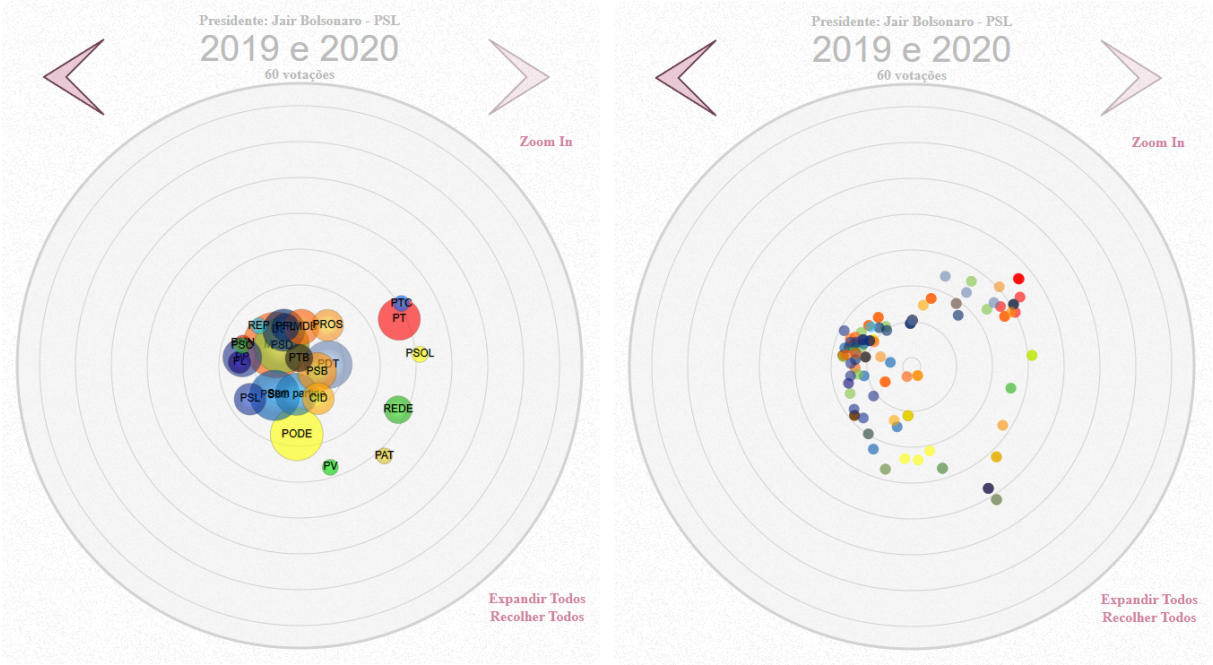
(A) Cada ponto representa um deputado. Aqueles que votam mais de acordo com a orientação do governo ficam posicionados mais à direita do gráfico. Quanto mais forte a cor do círculo, mais presente o deputado esteve em votações. (B) Mostra a quantidade de votos a favor e contra o governo de um determinado deputado/partido durante os governos em que ele esteve na Câmara. Cada barra representa um mês de votações. Fonte: Captura de tela do site <https://arte.estadao.com.br/politica/basometro/>, acessado em 08 de junho de 2024

3.3 Radar Parlamentar

Desenvolvido no início de 2012, o Radar Parlamentar (LEITE; TRENTO, 2016) é uma ferramenta de código aberto baseada na web utilizada para visualizar a proximidade entre parlamentares e partidos em diversos níveis legislativos. Através dos dados de votações nominais e da utilização da técnica de redução de dimensionalidade PCA, o Radar Parlamentar constrói e disponibiliza mapas ideológicos interativos de 2011 ao presente para a Câmara dos Deputados e para a Câmara Municipal de São Paulo, e de 2003 ao presente para o Senado Federal. Além disso, o Radar Parlamentar disponibiliza o download dos dados utilizados no formato SQL.

Os mapas do Radar Parlamentar apresentam partidos como círculos, onde o raio corresponde ao número de parlamentares do partido e a posição corresponde à média das posições dos parlamentares. Ao interagir com um destes círculos, é possível separá-lo em seus respectivos legisladores, mostrando suas posições individuais.

Figura 3 – Mapa ideológico do Radar Parlamentar



Mapa ideológico dos partidos (esquerda) e dos parlamentares (direita) do Senado Federal de 2019 a 2020. Quanto mais próximos dois partidos ou parlamentares, mais parecidos foram seus votos em plenário no período. Fonte: Captura de tela do site <https://radarparlamentar.polignu.org/radar/sen/>, acessado em 08 de junho de 2024

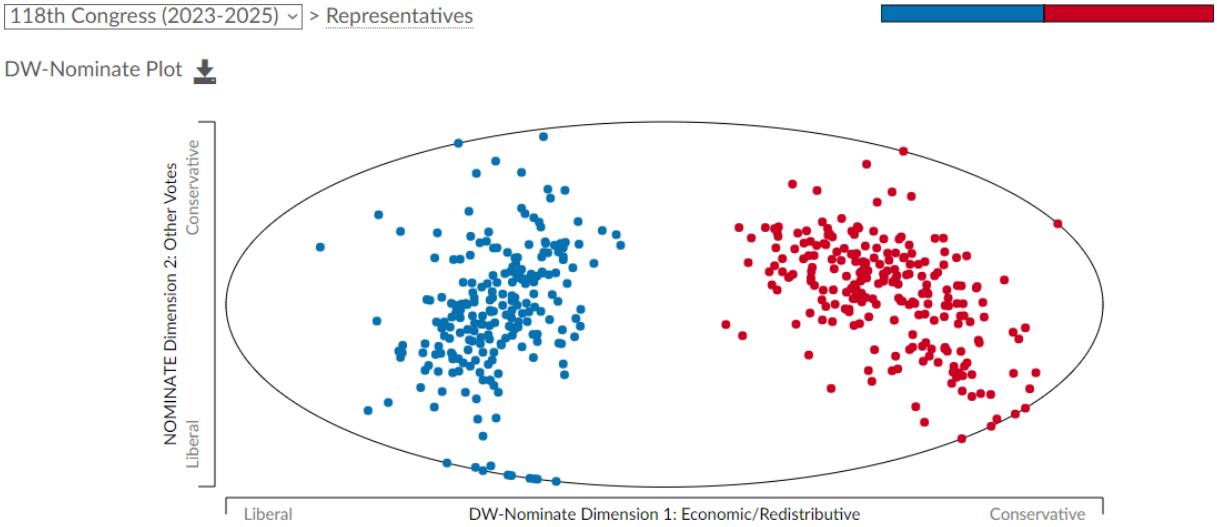
3.4 Voteview

O Voteview (LEWIS et al., 2019) é uma ferramenta analítica poderosa, projetada para fornecer uma visão abrangente dos comportamentos de votação do Congresso dos

Estados Unidos ao longo de sua história. Desenvolvida inicialmente por Keith T. Poole e Howard Rosenthal na Universidade Carnegie-Mellon entre 1989 e 1992, a ferramenta evoluiu de suas origens no DOS para interfaces mais amigáveis no Windows e baseadas na web, melhorando continuamente seus recursos. O atual Voteview integra várias funcionalidades de seus predecessores, combinando visualizações ricas com os extensos dados e estimativas DW-NOMINATE (POOLE; ROSENTHAL, 1985), medidas quantitativas da ideologia política de um legislador com base no seu comportamento em votações nominais, que fizeram do Voteview original um recurso inestimável.

Um dos instrumentos mais centrais do Voteview permite que os usuários visualizem um mapa ideológico (Figura 4), construído a partir das pontuações DW-NOMINATE derivada dos dados de votações nominais, que traça as posições liberal-conservadoras de Senadores e Representantes, tanto em assuntos econômicos quanto em assuntos diversos, tipicamente de natureza social. É possível visualizar todos os Congressos desde o 1º Congresso dos Estados Unidos, de 1789, até o presente.

Figura 4 – Mapa ideológico do Voteview

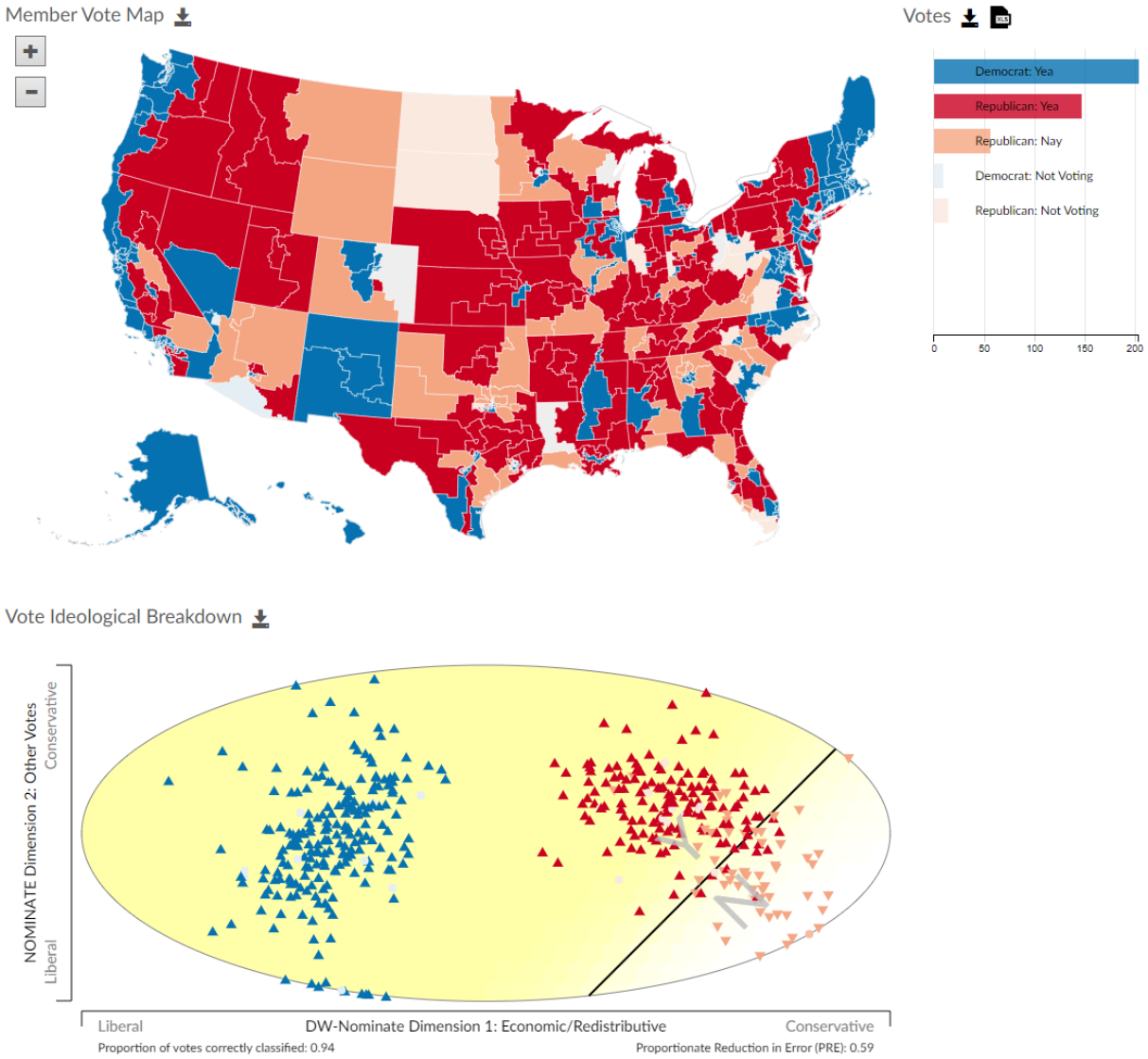


Mapa ideológico da Câmara dos Representantes do 118º Congresso dos Estados Unidos. Há dois eixos, um econômico e outro sobre assuntos diversos. Cada ponto é um parlamentar, e quanto mais próximos dois parlamentares, mais parecidos foram seus votos em plenário sobre o respectivo eixo no período. Fonte: Captura de tela do site <https://voteview.com/congress/house/>, acessado em 08 de junho de 2024

Além disso, o Voteview possibilita a visualização do perfil ideológico de uma proposta legislativa (Figura 5), de duas maneiras: ele permite ver a distribuição da votação por partido, por voto e por região através de um mapa dos Estados Unidos interativo. Ele também apresenta o mapa ideológico (Figura 4) dos legisladores durante a votação da proposta, com uma adição: uma linha divisória (que representa a neutralidade) separa o espaço nos lados “Y” (a favor) e “N” (contra), onde quanto mais distante da linha, maior

a probabilidade de determinado legislador votar de acordo com o lado em que ele está. Ao selecionar uma região, tipo de voto ou partido no mapa dos Estados Unidos, é mostrado no mapa ideológico somente os legisladores correspondentes.

Figura 5 – Perfil ideológico de uma proposta no Voteview

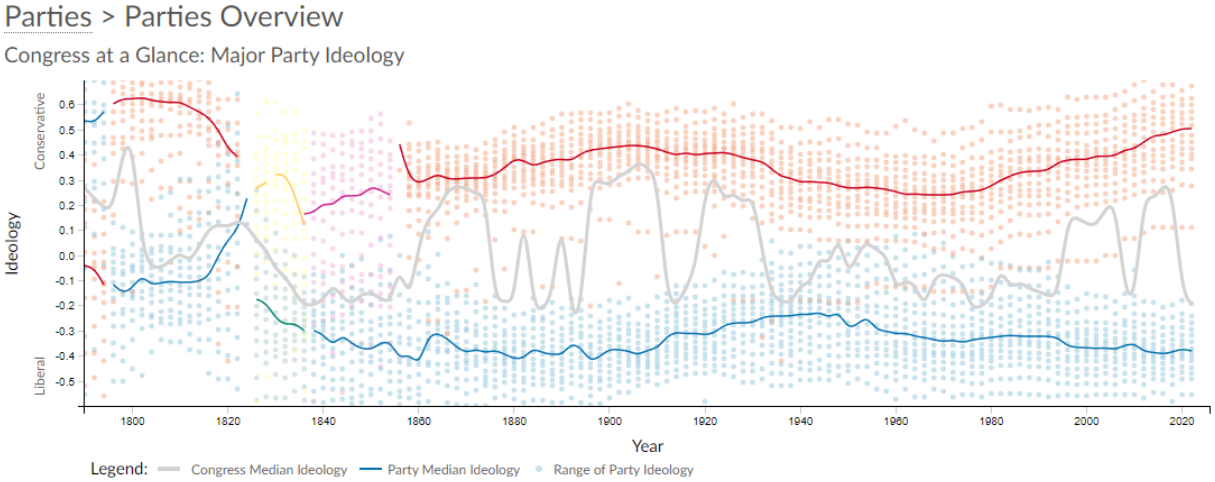


Em cima, o gráfico interativo dos Estados Unidos que permite selecionar os legisladores por região, tipo de voto e partido. Embaixo, o mapa ideológico (Figura 4) dos legisladores durante a votação da proposta, com a linha divisória. Triângulos representam legisladores votantes e círculos, não-votantes. Fonte: Captura de tela do site <https://voteview.com/rollcall/RH1180967>, acessado em 08 de junho de 2024

O Voteview também conta com um gráfico que mostra a aproximação ideológica entre os dois principais partidos estadunidenses desde o 1º Congresso dos Estados Unidos até o presente (Figura 6), incluindo partidos históricos como os Partidos Pró-Administração e Anti-Administração, Federalista e Democrata-Republicano, Jackson e Anti-Jackson e o Partido Whig, entre outros. Neste gráfico, é possível visualizar a média ideológica de cada

partido ao longo do tempo, a sua variação, e a média ideológica geral.

Figura 6 – Linha do tempo ideológica do Voteview



Fonte: Captura de tela do site <https://voteview.com/parties/all>, acessado em 08 de junho de 2024

Ainda, a plataforma permite que os usuários visualizem os históricos de votação de membros individuais do Congresso ao longo do tempo, oferecendo insights sobre seu comportamento político e mudanças de ideologia. Além disso, os usuários podem pesquisar votações nominais, membros e partidos usando um motor de busca robusto que suporta álgebra booleana e pesquisa baseada em campos (*field-based searching*). Os resultados da pesquisa podem ser exportados em formatos como Excel e JSON.

Por fim, o Voteview fornece acesso às pontuações DW-NOMINATE, através do Rvoteview, um pacote da linguagem de programação R disponibilizado pelo Voteview, permitindo que os usuários baixem e analisem dados diretamente no R. Isso facilita análises personalizadas e a integração com outras fontes de dados para pesquisas abrangentes.

3.5 CivisAnalysis

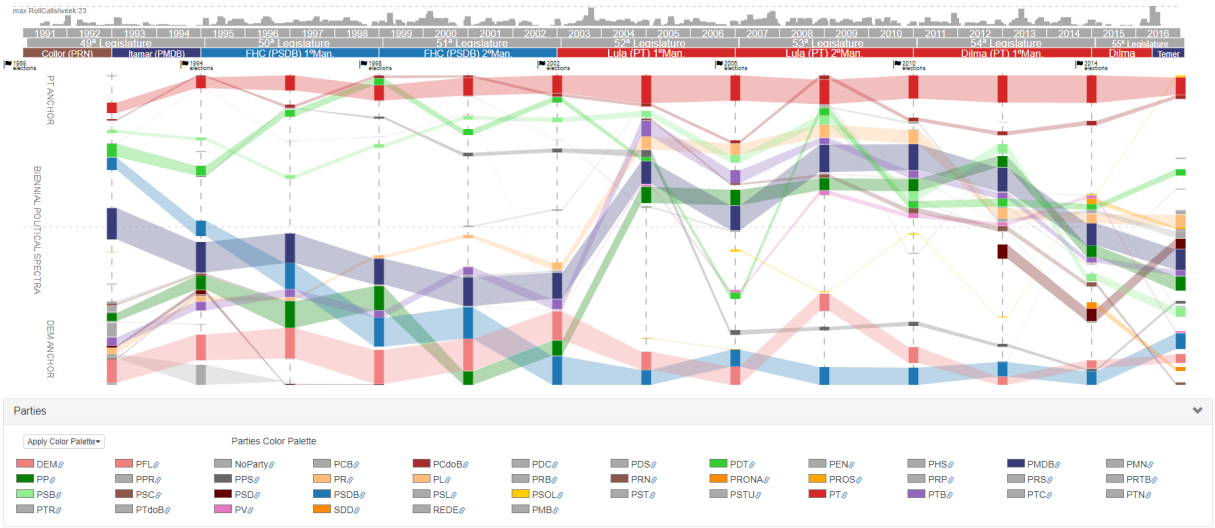
O CivisAnalysis (BORJA; FREITAS, 2015) é uma ferramenta baseada na web, projetada para visualizar dados de votações nominais da Câmara dos Deputados do Brasil. Desenvolvido para aprimorar a compreensão pública dos processos políticos e comportamentos de votação, ele oferece uma visualização abrangente das ações legislativas e dinâmicas políticas ao longo de 24 anos, abrangendo seis legislaturas e seis eleições presidenciais. O CivisAnalysis é composto por dois módulos principais: o Módulo de Visão Geral e o Modulo de Inspeção.

O Módulo de Visão Geral (Figura 7) oferece uma exibição cronológica abrangente da história política do Brasil. Ele delineia meticulosamente o espectro político de vários

partidos, capturando suas mudanças ideológicas e alinhamentos através de diferentes legislaturas e ciclos eleitorais.

Esta linha do tempo é enriquecida com anotações que marcam mandatos presidenciais, eleições gerais e volume de votações nominais, proporcionando uma narrativa detalhada da evolução política do Brasil. Este recurso oferece um contexto essencial para os dados visuais, permitindo aos usuários correlacionar eventos políticos significativos com mudanças nas ideologias partidárias e alterações legislativas.

Figura 7 – Módulo de Visão Geral do CivisAnalysis



Linhas representam partidos ao longo do tempo, onde a grossura representa a quantidade de legisladores no partido. Quanto mais próximos dois partidos, mais parecidos foram seus votos em plenário no período. Fonte: Captura de tela do site <https://www.inf.ufrgs.br/~rnmsilva/fgborja/public/>, acessado em 08 de junho de 2024

É possível selecionar um período no Modulo de Visão Geral para que este apareça no Módulo de Inspeção (Figura 8), que oferece uma exploração detalhada por meio de múltiplas visualizações coordenadas. Este módulo inclui três componentes principais: o Infográfico da Câmara, o Espectro dos Deputados e o Espectro das Votações. Cada componente tem um propósito único, oferecendo insights sobre a distribuição e o comportamento dos deputados e seus padrões de votação.

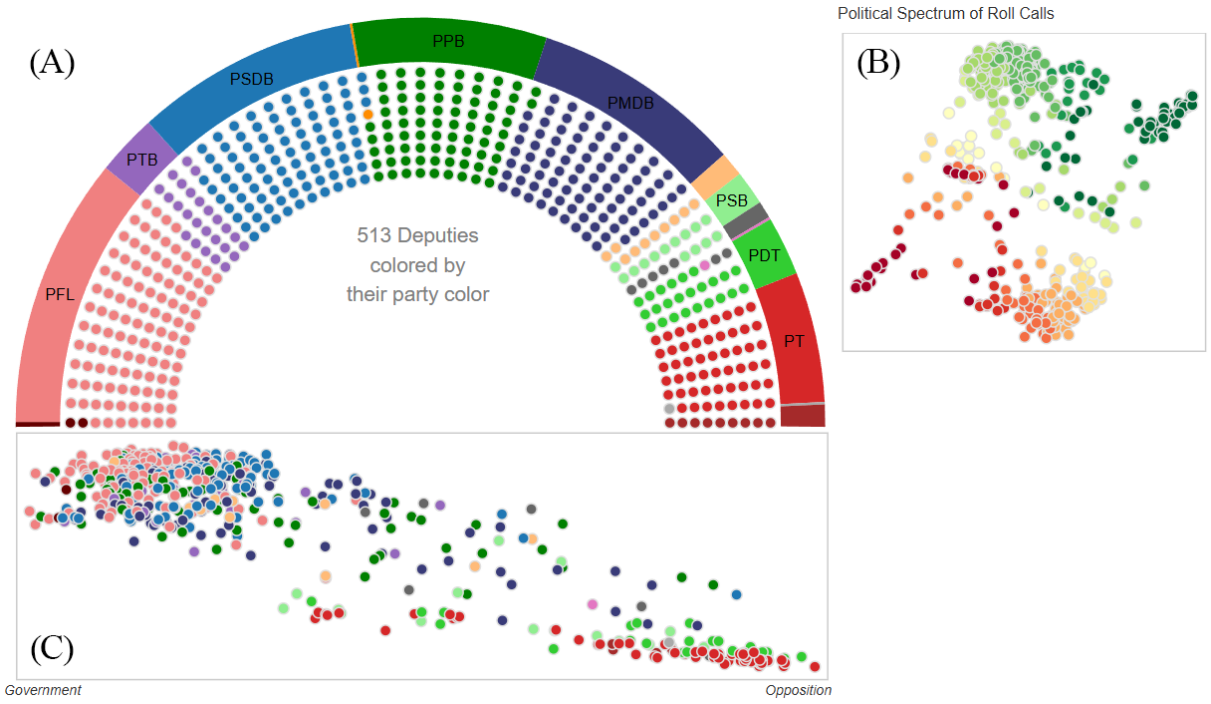
O Infográfico da Câmara apresenta a distribuição dos deputados por partido, permitindo que os usuários visualizem o número de deputados por partido em um formato de fácil compreensão. Este recurso suporta classificação e filtragem, permitindo que os usuários se concentrem em partidos ou tendências específicas. Ao fornecer uma representação visual clara da composição partidária dentro da câmara, o infográfico facilita uma melhor compreensão do cenário político.

O Espectro dos Deputados ilustra o espectro político dos deputados através de

um gráfico de dispersão onde cada ponto representa um deputado. Esses pontos são posicionados com base em seu comportamento de votação, utilizando os métodos de redução de dimensionalidade PCA ou t-SNE, com deputados que exibem padrões de votação semelhantes aparecendo mais próximos uns dos outros. Esta visualização ajuda a identificar alianças e semelhanças ideológicas entre os deputados. Além disso, os usuários podem passar o cursor sobre qualquer ponto para obter informações detalhadas sobre um deputado específico, aprimorando a profundidade da análise.

O Espectro das Votações visualiza as votações nominais, destacando como diferentes votações estão relacionadas. Este componente exibe aglomerados de votações semelhantes, utilizando características como o perfil dos legisladores na distribuição de votos, permitindo que os usuários identifiquem padrões e questões-chave no comportamento legislativo. Ao examinar esses aglomerados, os usuários podem obter insights sobre o processo legislativo e os fatores que influenciam as decisões de votação.

Figura 8 – Módulo de Inspeção do CivisAnalysis



O Módulo de Inspeção, abrangindo o primeiro mandato de Fernando Henrique Cardoso e utilizando a técnica PCA. (A) Infográfico da Câmara. (B) Espectro das Votações. (C) Espectro dos Deputados. Fonte: Captura de tela do site <https://www.inf.ufrgs.br/~rnmsilva/fgborja/public/>, acessado em 08 de junho de 2024

No entanto, o CivisAnalysis apresenta algumas limitações. Primeiramente, ele não abrange as legislaturas mais recentes. Além disso, sua gama de técnicas de redução de dimensionalidade é relativamente limitada, o que restringe a variedade de visualizações disponíveis para auxiliar na análise política. Embora o cenário político brasileiro seja

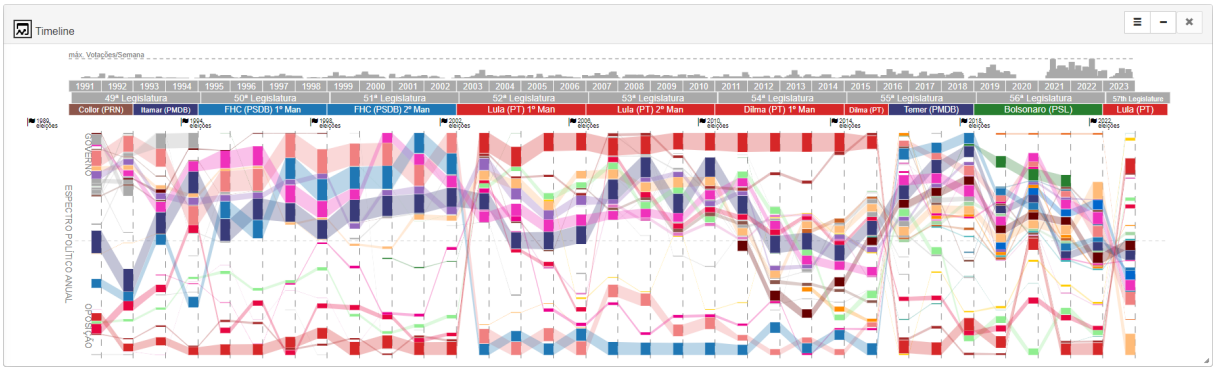
altamente plural em termos de partidos e interesses, o CivisAnalysis não oferece uma opção de clustering, que seria útil para avaliar coalizões. Adicionalmente, a ferramenta utiliza apenas dados da Câmara dos Deputados, o que impede a realização de análises sobre o Senado Federal. Por fim, não é possível exportar os modelos gerados, restringindo a flexibilidade da ferramenta.

3.6 CivisAnalysis 2.0

O CivisAnalysis 2.0 (SILVA; SPRITZER; FREITAS, 2018) é uma aplicação baseada na web projetada para facilitar a exploração e análise de dados de comportamento político da Câmara dos Deputados do Brasil. Ela é uma evolução da plataforma original CivisAnalysis, incluindo legislaturas mais recentes, incorporando novas visualizações e novos métodos estatísticos que visam melhorar a experiência do usuário e as capacidades analíticas. O sistema permite aos usuários analisar dados de votações nominais, acompanhar os padrões de votação dos deputados e avaliar a coesão partidária ao longo do tempo.

Análogo ao Módulo de Visão Geral da versão original, o CivisAnalysis 2.0 conta com uma Linha do Tempo (Figura 9), que funciona de maneira muito similar. Ela é enriquecida com anotações que marcam mandatos presidenciais, eleições gerais e volume de votações nominais e permite correlacionar eventos políticos significativos com mudanças nas ideologias partidárias e alterações legislativas. Uma melhora significativa é de que, nesta versão, em geral, os partidos com maior grau de alinhamento com determinado governo estão posicionados mais acima, facilitando a identificação de governo e oposição.

Figura 9 – Linha do Tempo do CivisAnalysis 2.0



Linhas representam partidos ao longo do tempo, onde a grossura representa a quantidade de legisladores no partido. Quanto mais próximos dois partidos, mais parecidos foram seus votos em plenário no período. Fonte: Captura de tela do site <https://www.inf.ufrgs.br/~rnmsilva/CivisAnalysis2/>, acessado em 08 de junho de 2024

Diferentemente do seu antecessor, o CivisAnalysis 2.0 utiliza uma abordagem de janelas flutuantes interconectadas, tendo a Linha do Tempo como elemento central. Assim,

ao selecionar um período de tempo na Linha do Tempo, é possível gerar diversas visualizações. Ainda, seleções de deputados ou partidos são refletidas em todas as visualizações. Ao selecionar um deputado ou grupo de deputados, é possível gerar uma Linha do Tempo personalizada (Figura 10), que indica o posicionamento destes em relação aos partidos ao longo do tempo.

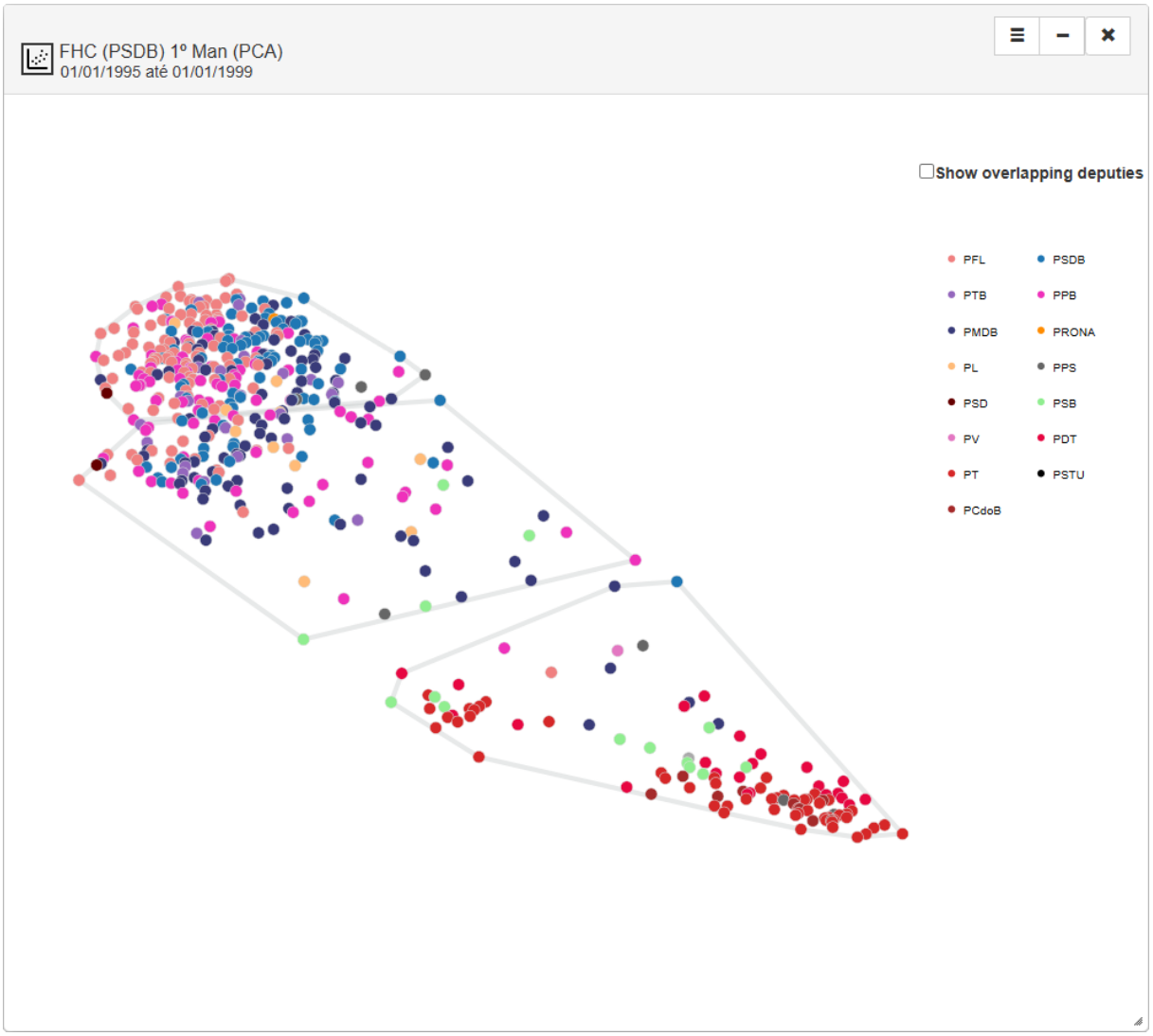
Figura 10 – Linha do Tempo de um deputado no CivisAnalysis 2.0

Linha do Tempo do deputado Aécio Neves, abrangindo o primeiro mandato de Fernando Henrique Cardoso. Linhas representam partidos ao longo do tempo, exceto a linha mais fina, que representa o deputado. Quanto mais próximas duas linhas, mais parecidos foram seus votos em plenário no período. Fonte: Captura de tela do site <https://www.inf.ufrgs.br/~rnmsilva/CivisAnalysis2/>, acessado em 08 de junho de 2024

Assim como no CivisAnalysis original, um Espectro Político de Deputados pode ser construído a partir de um recorte na Linha do Tempo. Foram adicionados novos métodos de redução de dimensionalidade: além das técnicas PCA e t-SNE, é possível utilizar os

métodos MDS e UMAP para encontrar as posições ideológicas relativas dos deputados. Além disso, é possível clusterizar o modelo a partir da técnica k-means.

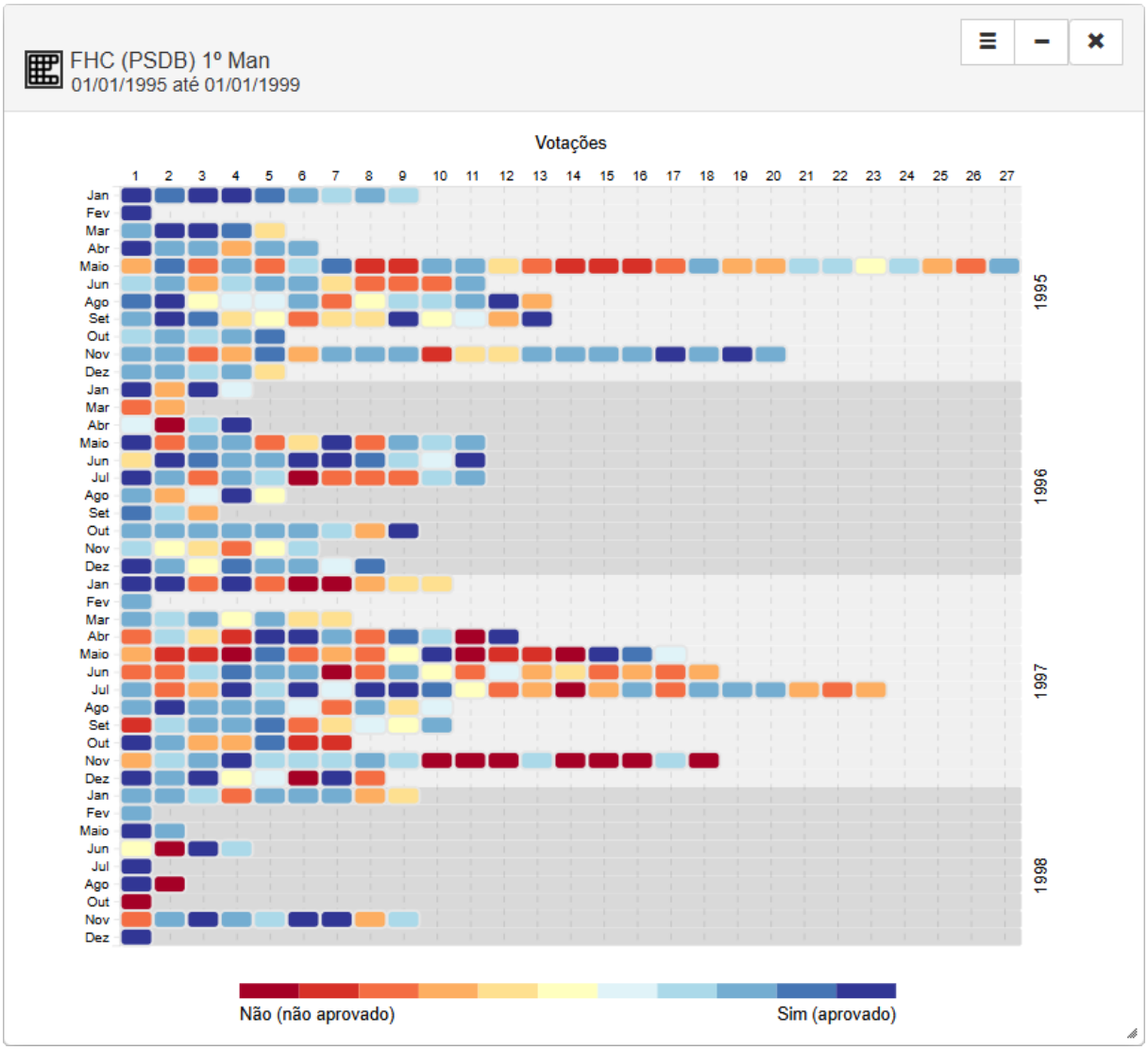
Figura 11 – Espectro Político de Deputados do CivisAnalysis 2.0



Mapa ideológico dos deputados criado utilizando a técnica PCA, agrupado em 3 clusters, abrangindo o primeiro mandato de Fernando Henrique Cardoso. Quanto mais próximos dois deputados, mais parecidos foram seus votos em plenário no período. Fonte: Captura de tela do site <https://www.inf.ufrgs.br/~rnmsilva/CivisAnalysis2/>, acessado em 08 de junho de 2024

No entanto, em vez de fornecer um Espectro das Votações, o CivisAnalysis 2.0 oferece um Mapa de Votações, que funciona de maneira diferente. Ele lista as propostas legislativas do período selecionado de maneira interativa, com cada proposta exibindo uma cor associada ao seu grau de aprovação. Ao interagir com uma proposta, são exibidas a sua identificação e a sua emenda.

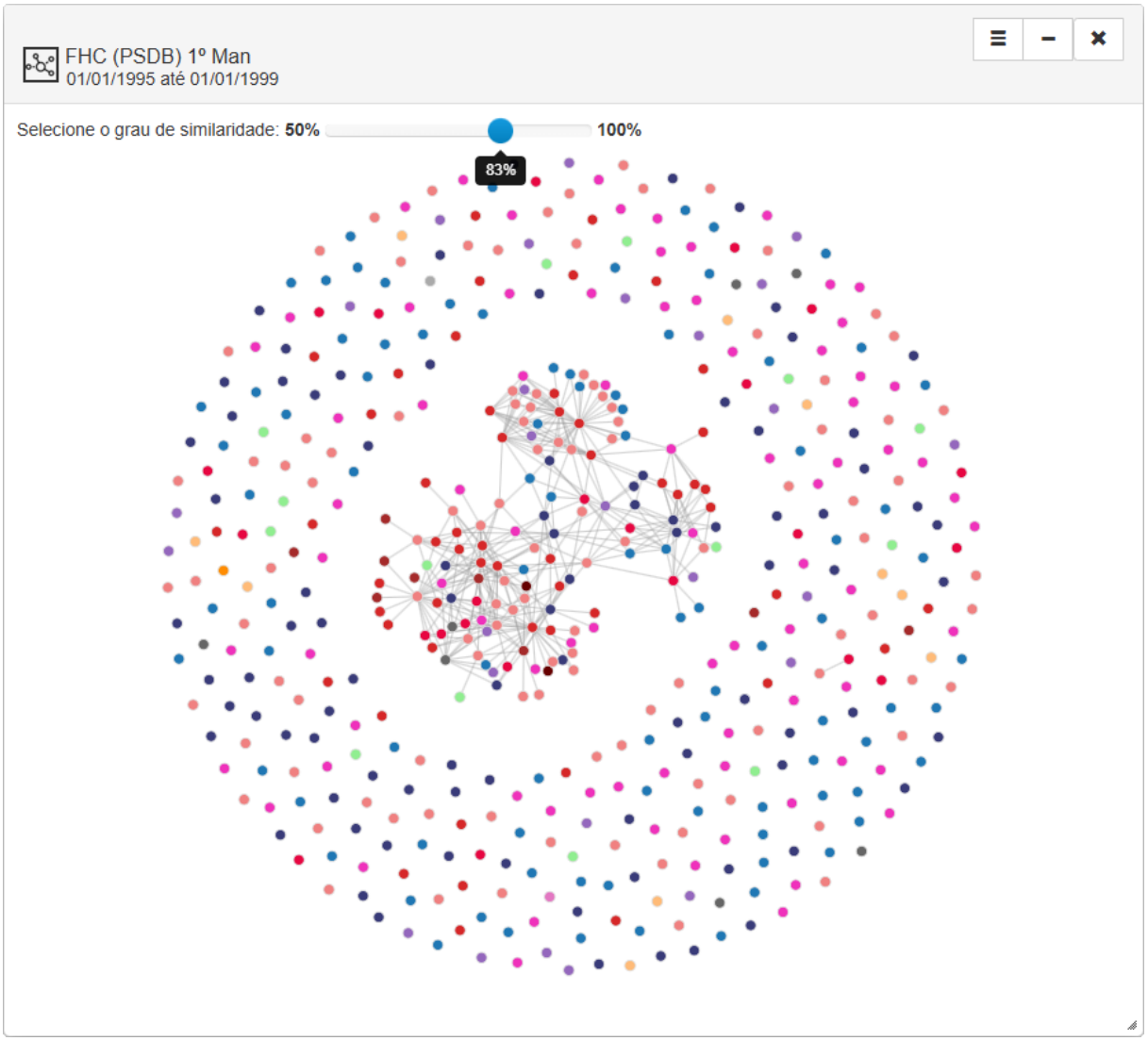
Figura 12 – Mapa de Votações do CivisAnalysis 2.0



Mapa de Votações abrangindo o primeiro mandato de Fernando Henrique Cardoso. Fonte: Captura de tela do site <https://www.inf.ufrgs.br/~rnmsilva/CivisAnalysis2/>, acessado em 08 de junho de 2024

O CivisAnalysis 2.0 também adiciona algumas novas visualizações, como o Grafo de Similaridade dos Deputados. Neste grafo é possível selecionar uma porcentagem de alinhamento, onde duas vértices, que representam deputados, só tem uma aresta entre si caso seus votos atinjam o grau de alinhamento definido. Dessa forma, é possível analisar legisladores influentes ou grupos de deputados com opiniões similares.

Figura 13 – Grafo de Similaridade dos Deputados do CivisAnalysis 2.0



O gráfico mostra grupos de deputados que votaram com um certo grau de alinhamento. Neste gráfico, um par de deputados está conectado quando os dois deputados alcançaram 83% ou mais de alinhamento nas votações. Fonte: Captura de tela do site <https://www.inf.ufrgs.br/~rnmsilva/CivisAnalysis2/>, acessado em 08 de junho de 2024

Contudo, o CivisAnalysis 2.0 não resolve todas as limitações do seu antecessor. Não há dados sobre o Senado Federal, e os modelos gerados não podem ser exportados. Além disso, a técnica de clustering disponível apresenta várias problemáticas que podem impactar negativamente a análise, além de exigir que o usuário especifique o número de clusters de antemão (AHMED; SERAJ; ISLAM, 2020).

4 Projeto

A maioria dos projetos nacionais existentes foca no usuário leigo (SILVA; SPRITZER; FREITAS, 2018; BORJA; FREITAS, 2015; LEITE; TRENTO, 2016; ESTADÃO, 2019), e de fato a facilidade de acesso é indiscutivelmente valiosa para o fortalecimento democrático. No entanto, como discutido no capítulo anterior, isso geralmente ocorre em detrimento dos pesquisadores. Os modelos subjacentes às visualizações frequentemente não são exportáveis, o que impede a aplicação de transformações adicionais ou análises estatísticas que não estão disponíveis na ferramenta originária.

Além disso, muitos dos projetos brasileiros ignoram o Senado Federal. Entre outros motivos, provavelmente isso deve acontecer pois o formato dos dados e a API do Senado Federal e da Câmara dos Deputados são diferentes, necessitando esforço extra para acessar esse conjunto de dados.

Assim, este projeto buscará equilibrar acessibilidade e utilidade acadêmica por meio do desenvolvimento de uma biblioteca em Python. No nível mais básico, os usuários poderão gerar mapas ideológicos utilizando parâmetros sugeridos e sem a necessidade de grande conhecimento estatístico. Em um nível mais avançado, a solução será inteiramente compatível com a biblioteca scikit-learn[1].

Reconhecida como uma das principais bibliotecas de estatística e aprendizado de máquina no ecossistema Python, scikit-learn é amplamente utilizada no meio acadêmico e por grandes empresas, como J.P. Morgan, Spotify e Booking.com[2]. Essa popularidade se deve não apenas ao seu desempenho robusto, mas também à sua facilidade de uso e extensa documentação, que a tornam acessível para desenvolvedores e pesquisadores.

Além disso, a solução facilitará a criação de gráficos a partir da biblioteca matplotlib[3], a principal biblioteca de visualização de dados no Python. Predominante em aplicações científicas e acadêmicas, matplotlib permite a criação de gráficos personalizados e detalhados, fundamentais para análises visuais precisas. Essa biblioteca é amplamente reconhecida e adotada em projetos de ciência de dados devido à sua flexibilidade e capacidade de integração com outras ferramentas de análise.

Além disso, as funcionalidades (Tabela 1) abrangerão a coleta de dados de votações nominais de ambas as casas legislativas e sua disponibilização através de uma API unificada. Esta API também permitirá a realização de análises e a geração de gráficos estatísticos diversos, não se limitando apenas à criação de mapas ideológicos, que é o foco central

[1] https://scikit-learn.org
[2] https://scikit-learn.org/stable/testimonials/testimonials.html
[3] https://matplotlib.org

deste trabalho.

Tabela 1 – Requisitos funcionais

Identificador	**Requisito**	**Classificação**
RF-01	A biblioteca deve ser capaz de coletar dados de votações nominais da Câmara dos Deputados e do Senado Federal	Essencial
RF-02	A biblioteca deve disponibilizar os dados coletados através de uma API unificada	Essencial
RF-03	A biblioteca deve facilitar a geração de mapas ideológicos	Essencial
RF-04	A biblioteca deverá sugerir parâmetros razoáveis caso não especificados, baseando-se na literatura disponível	Essencial
RF-05	A biblioteca deve ser compatível com a biblioteca de aprendizado de máquina scikit-learn	Essencial
RF-06	A biblioteca deve permitir a geração de outros gráficos estatísticos, com base nos dados das votações nominais	Desejável
RF-07	A biblioteca deve ser capaz de exportar dados em diversos formatos como CSV, JSON e XML	Desejável

Dessa forma, espera-se que a biblioteca desenvolvida possa ser utilizada por pesquisadores acadêmicos, jornalistas, cidadãos e ativistas, partidos políticos e desenvolvedores, conforme descrito na Tabela 2.

Tabela 2 – Perfil de Usuários

Usuário	**Uso**
Pesquisadores acadêmicos	Analisar padrões de votação e estudar o comportamento parlamentar ao longo do tempo de maneira aprofundada
Jornalistas	Identificar tendências e coalizões políticas para reportagens e análises jornalísticas
Cidadãos e ativistas	Acompanhar a atuação dos representantes eleitos e promover a transparência e responsabilidade política
Partidos políticos	Monitorar a disciplina partidária e a coesão interna
Desenvolvedores	Criar ferramentas que utilizam dados de votações nominais sem a necessidade de mapear duas APIs

5 Desenvolvimento

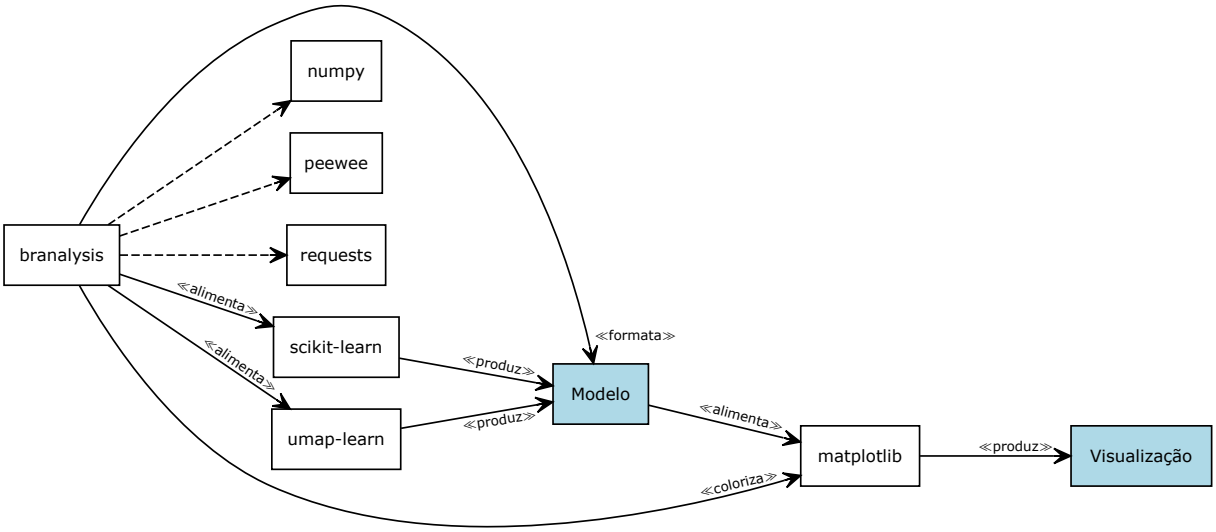
O desenvolvimento da biblioteca, doravante chamada de branalysis, foi realizado em Python puro. A escolha desta linguagem se deu por sua popularidade, facilidade de uso e por ser amplamente utilizada em projetos de ciência de dados e aprendizado de máquina. Além disso, Python possui uma vasta gama de bibliotecas de terceiros que facilitam o desenvolvimento de aplicações desse tipo.

A biblioteca foi projetada para interagir com scikit-learn e matplotlib sem incluí-las como dependências explícitas, operando apenas sobre estruturas comuns. Suas únicas dependências efetivas são as bibliotecas numpy[1], requests[2] e peewee[3]. A biblioteca numpy é utilizada para a construção das estruturas de dados necessárias para a redução de dimensionalidade, enquanto a biblioteca requests é utilizada para realizar requisições HTTP às APIs da Câmara dos Deputados e do Senado Federal. Por fim, a biblioteca peewee é utilizada para o gerenciamento do banco de dados SQLite ao qual os dados coletados são armazenados.

Dessa forma, outras bibliotecas de aprendizado de máquina podem ser utilizadas, desde que compatíveis com as mesmas estruturas de dados, como é o caso da umap-learn[4], que fornece o algoritmo UMAP ausente no scikit-learn. Conjuntamente, os facilitadores de visualização foram desenvolvidos de forma agnóstica em relação à biblioteca de visualização utilizada, providenciando recursos como formatação das estruturas e colorização. A Figura 14, a seguir, apresenta um diagrama que ilustra essas interações e destaca as bibliotecas usadas neste trabalho.

[1] https://numpy.org
[2] https://requests.readthedocs.io
[3] http://docs.peewee-orm.com
[4] https://umap-learn.readthedocs.io

Figura 14 – Diagrama do fluxo das bibliotecas utilizadas



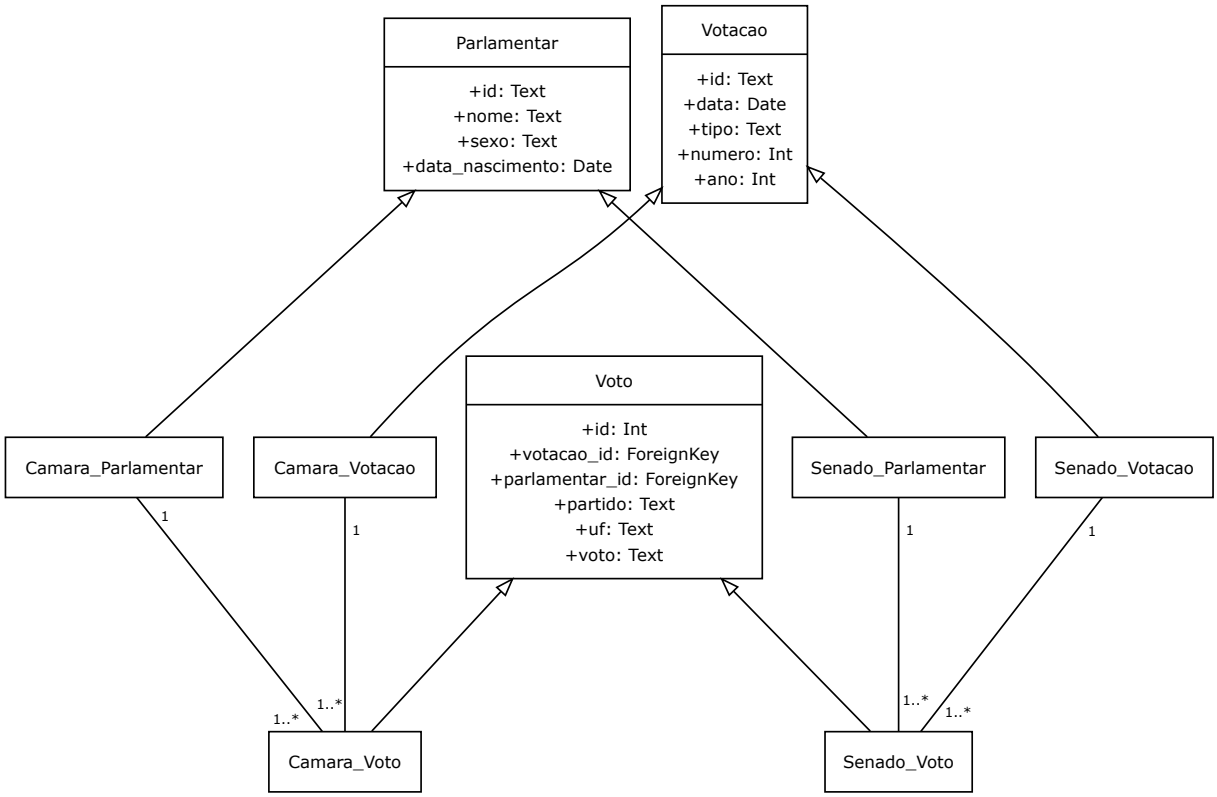
O branalysis foi empacotado utilizando a ferramenta de empacotamento de código Python setuptools[5], e distribuida através do Python Package Index (PyPI)[6]. A biblioteca foi desenvolvida para ser compatível com Python 3.9 e versões superiores.

5.1 Coleta de dados

A coleta de dados foi realizada por meio das APIs públicas disponibilizadas pela Câmara dos Deputados[7] e pelo Senado Federal[8]. O branalysis coleta os dados de votações nominais de ambas as casas legislativas, assim como informações sobre os legisladores e seus votos, os quais são armazenados em um banco de dados SQLite local, com modelagem simétrica entre as casas. O processo de coleta segue um modelo incremental: a cada execução, a biblioteca verifica se os dados referentes ao ano solicitado já foram previamente capturados. Caso positivo, a coleta é descartada para evitar duplicidade; caso contrário, os dados são extraídos e armazenados no banco de dados.

[5] https://setuptools.pypa.io
[6] https://pypi.org
[7] https://dadosabertos.camara.leg.br
[8] https://www12.senado.leg.br/dados-abertos/

Figura 15 – Diagrama do banco de dados do branalysis



A escolha de utilizar um banco de dados em vez de métodos mais primitivos de cacheamento, como arquivos CSV, foi motivada pela facilidade de uso e pela capacidade de reduzir o volume de dados, visando solucionar problemas inerentes às APIs disponíveis. Durante o desenvolvimento da biblioteca, foi constatado que diversos *endpoints* das plataformas apresentavam falhas. Requisições minimamente complexas, como a filtragem de votações por data, frequentemente resultavam em *timeout* ou não surgiam efeito.

Mesmo quando bem-sucedidas, os dados eram organizados de maneira ineficiente, exigindo múltiplas requisições adicionais para completar as informações que o branalysis buscava coletar. Essa limitação evidenciou a necessidade de métodos mais sofisticados para gerenciar o que já havia sido armazenado em cache e o que ainda faltava, a fim de evitar requisições desnecessárias e assegurar a integridade dos dados. A transacionalidade proporcionada pelo SQLite revelou-se essencial nesse processo, especialmente em cenários de interrupção inesperada da coleta de dados.

O uso de um bancos de dados relacional tem como beneficio adicional a facilidade de extração de dados, proporcionada pela ampla disponibilidade de ferramentas e bibliotecas que simplificam a conversão dos dados armazenados para diferentes formatos, como CSV, JSON ou XML, entre outros. Ademais, o SQLite, por ser um banco de dados embutido, elimina a necessidade de instalação de um servidor dedicado, o que reduz o impacto técnico para usuários com pouca ou nenhuma experiência em gerenciamento de bancos de dados.

5.2 Fluxo de análise

O fluxo foi estabelecido com o objetivo de trazer maior simplicidade e flexibilidade ao processo de análise de votações nominais. O acesso aos dados é realizado por meio da criação de um objeto que representa a Câmara dos Deputados ou o Senado Federal, o qual é responsável por iniciar o processo de cacheamento e disponibilizar as informações do banco de dados resultante de maneira acessível e interativa, por meio de seus métodos.

A criação desse objeto envolve a definição de um intervalo temporal, especificando as datas de início e fim do período a ser analisado. Com o objeto criado, é possível acessar informações detalhadas sobre a casa legislativa e suas votações nominais. De maneira geral, a biblioteca oferece, em relação aos dados coletados, as seguintes facilidades:

- Parlamentares em exercício durante o período. Cada parlamentar é representado por um objeto, cuja estrutura inclui as informações do banco de dados, como nome, sexo, data de nascimento, votações e votos. Ademais, o objeto disponibiliza métodos que permitem a obtenção de dados relativos à atuação parlamentar, incluindo frequência, filiação partidária, além das unidades federativas e macrorregiões que representaram durante o período em análise.
- Votações nominais realizadas durante o período. Cada votação é representada por um objeto, cuja estrutura inclui as informações do banco de dados, como data, número, tipo e os votos dos parlamentares.
- Tipos de votações realizadas durante o período.
- Partidos políticos em atividade durante o período.
- Unidades federativas existentes durante o período.

A partir desses dados, é possível gerar uma matriz de votos, que é a base para a aplicação dos algoritmos de redução de dimensionalidade. Essa matriz de votos $V(n \times m)$, onde n representa o número de parlamentares e m corresponde ao número de votações nominais, contém, em cada célula V_{ij}, o voto do parlamentar i na votação nominal j. O valor numérico associado a cada voto é determinado por um *transformador*.

O transformador padrão converte os votos em 1 para “Sim”, -1 para “Não” e 0 para “Abstenção” e “Obstrução”. O usuário pode definir diferentes transformadores, conforme o aspecto dos votos que deseja explorar. Por exemplo, um transformador que atribui o valor 1 a todos os votos presentes poderia ser utilizado para identificar parlamentares que costumam participar das mesmas votações.

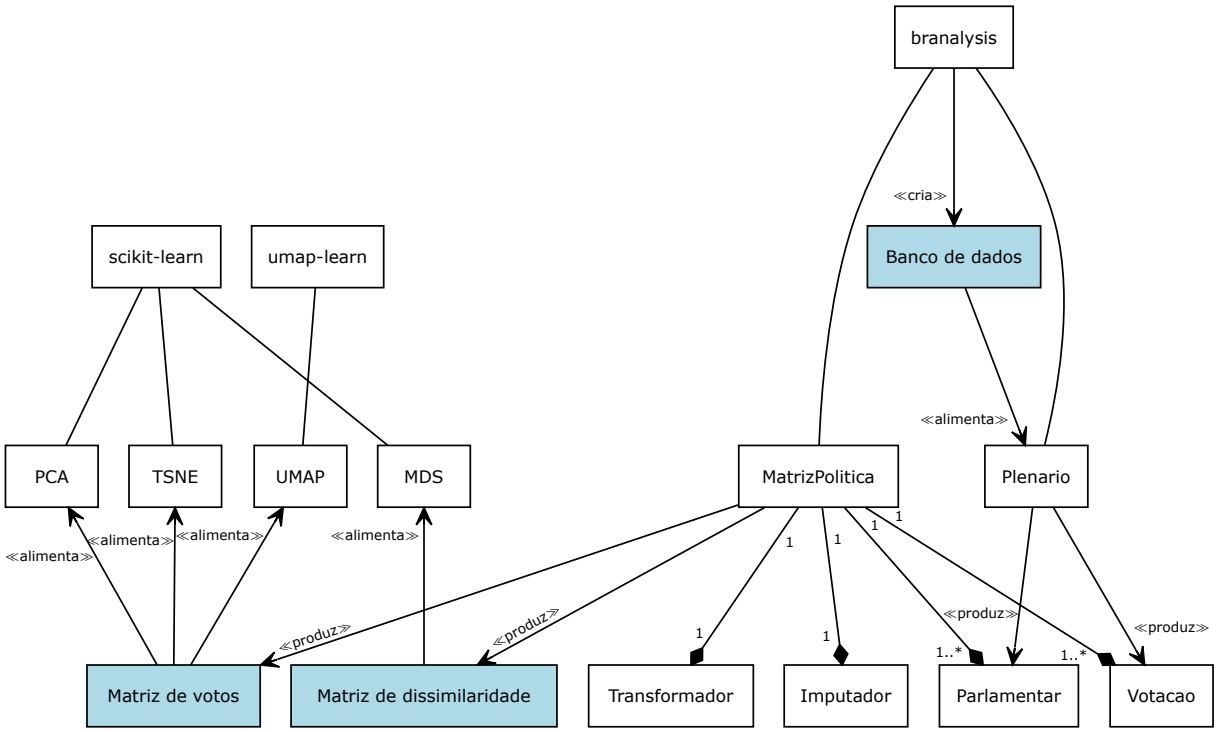
Há também um *imputador*, que define o que fazer com votos faltantes. O imputador padrão substitui os votos ausentes pela média do valor númerico dos votos do partido

do parlamentar naquela votação específica. Assim como o transformador, o imputador também pode ser configurado pelo usuário, conforme suas necessidades analíticas.

Com esses componentes definidos, foi pensada a criação de dois tipos de matrizes de votos: uma matriz de parlamentares e uma matriz de votações, sendo uma a transposta da outra. Dessa forma, torna-se possível aplicar técnicas de redução de dimensionalidade para gerar tanto mapas ideológicos de parlamentares, como já discutido, quanto mapas ideológicos de votações, que identificam quais votações são mais semelhantes entre si, com base no perfil de voto dos parlamentares.

Além disso, foi implementada também uma matriz de dissimilaridade, exigida pelo algoritmo MDS. A matriz de dissimilaridade, denotada por $D(n \times n)$, onde n representa o número de parlamentares, contém, em cada célula D_{ij}, o valor da dissimilaridade entre os parlamentares i e j. A dissimilaridade é definida como a porcentagem de votos divergentes entre os parlamentares, sendo um voto considerado divergente caso a diferença do valor numérico de seus votos, após a aplicação do transformador e imputador, seja maior que ε, que pode ser definido pelo usuário. Essas estruturas são então utilizadas para gerar modelos de redução de dimensionalidade. De modo geral, o fluxo de análise pode ser visualizado na Figura 16.

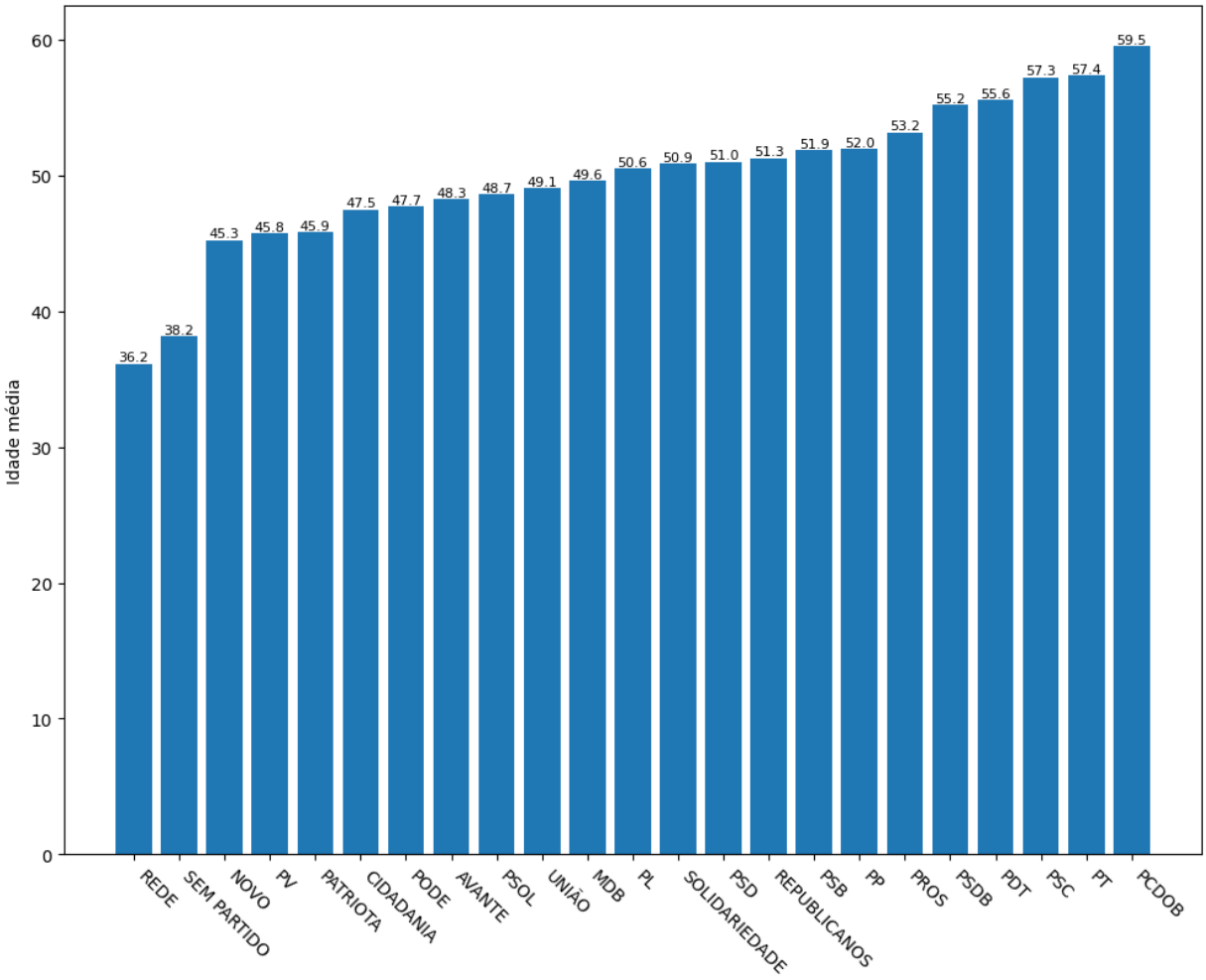
Figura 16 – Diagrama do fluxo de análise



5.3 Geração de gráficos

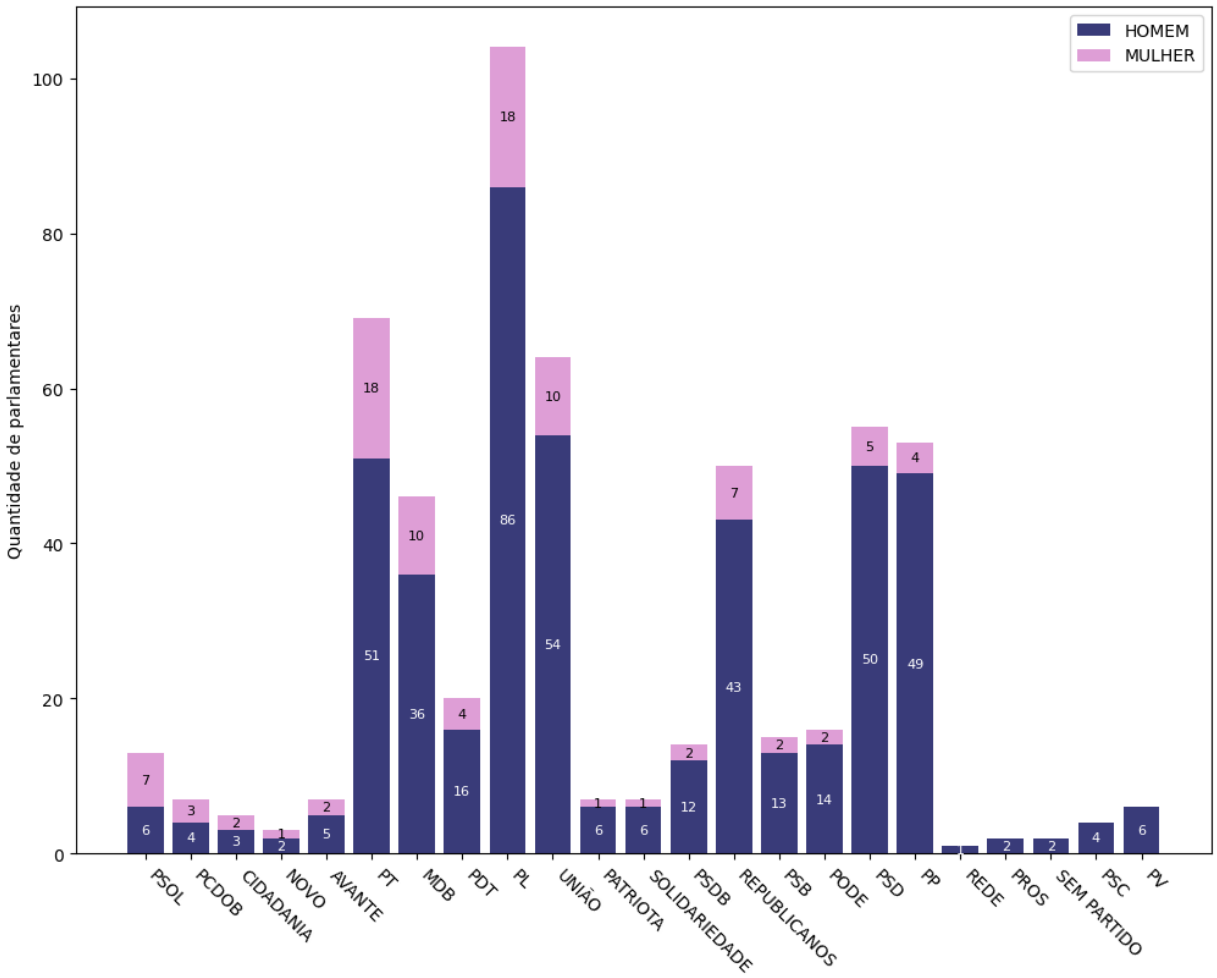
Apesar do foco deste trabalho ser a geração de mapas ideológicos, a biblioteca branalysis foi projetada para ser flexível e permitir a criação de diversos tipos de gráficos estatísticos, servindo como um facilitador no acesso a dados sobre parlamentares e votações nominais. Alguns exemplos, produzidos pelos códigos dos Apêndices B e C, são apresentados a seguir.

Figura 17 – Idade média dos partidos da Câmara dos Deputados de 2023



Fonte: autoria própria.

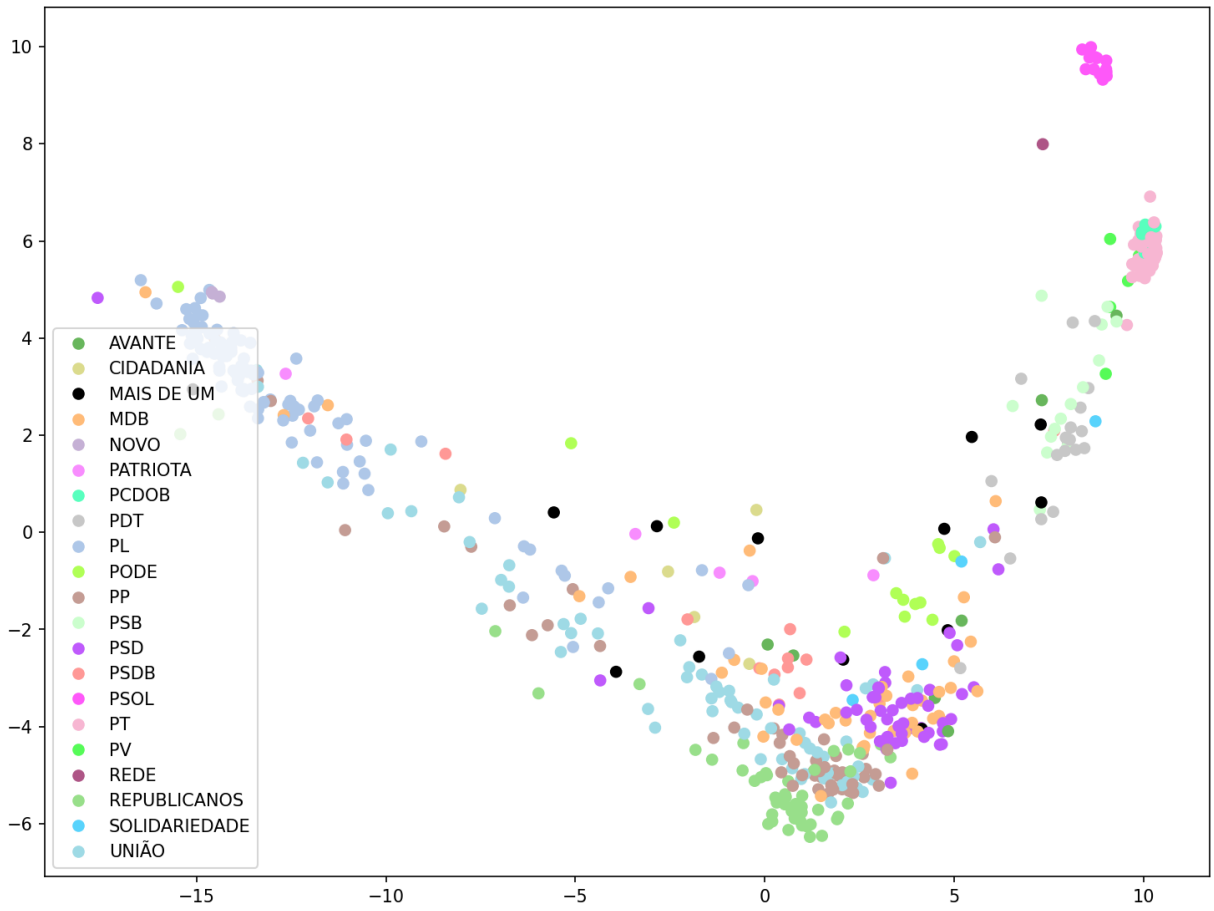
Figura 18 – Igualdade de gênero dos partidos da Câmara dos Deputados de 2023



Quanto mais à esquerda, maior a proporção de mulheres no partido. Fonte: autoria própria.

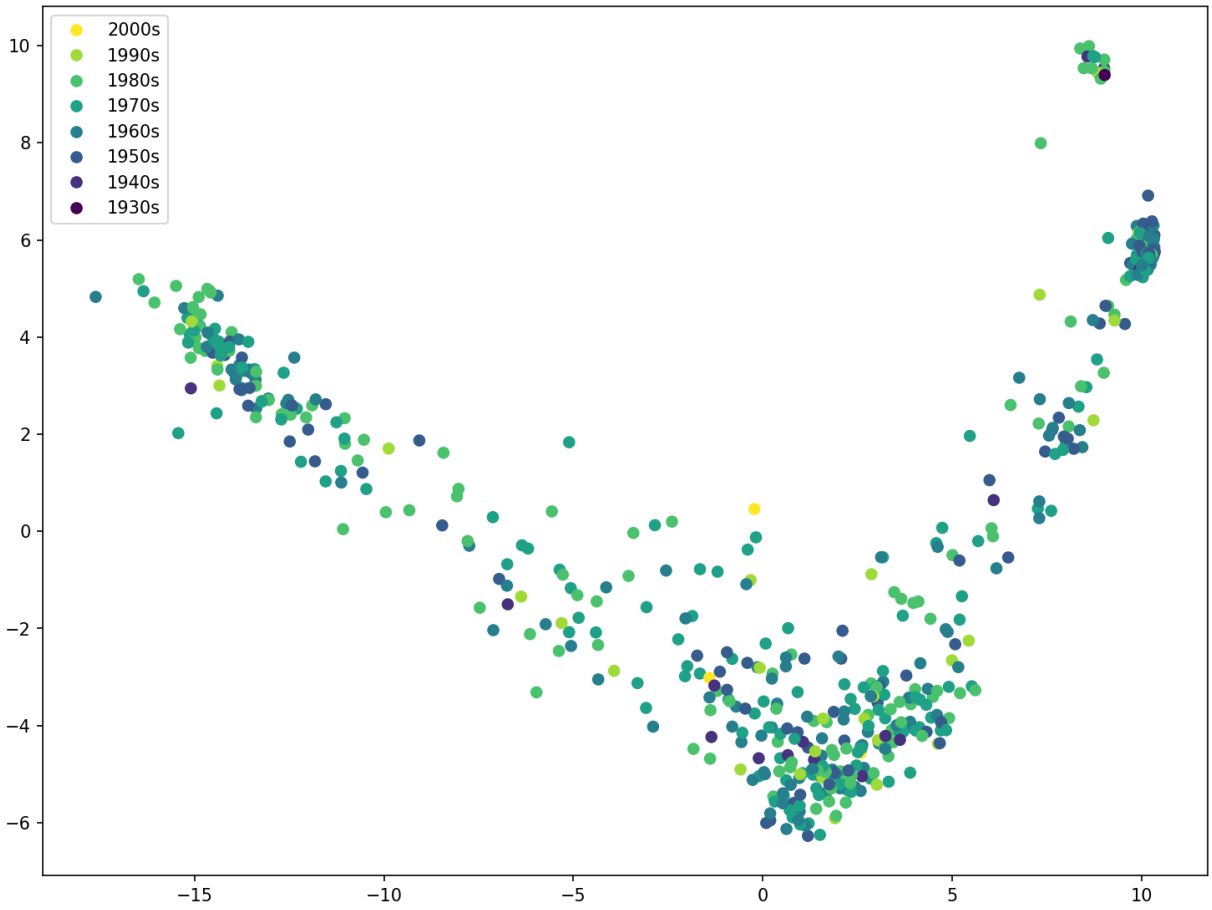
Além disso, a própria criação de mapas ideológicos pode ser personalizada, permitindo a representação de diferentes aspectos dos dados, como gênero, idade, macrorregião, entre outros. Assim, com poucas linhas de código (Apêndice D), é possível criar um mapa ideológico simples, assim como variações deste:

Figura 19 – Mapa ideológico da Câmara dos Deputados de 2023 por partido - PCA



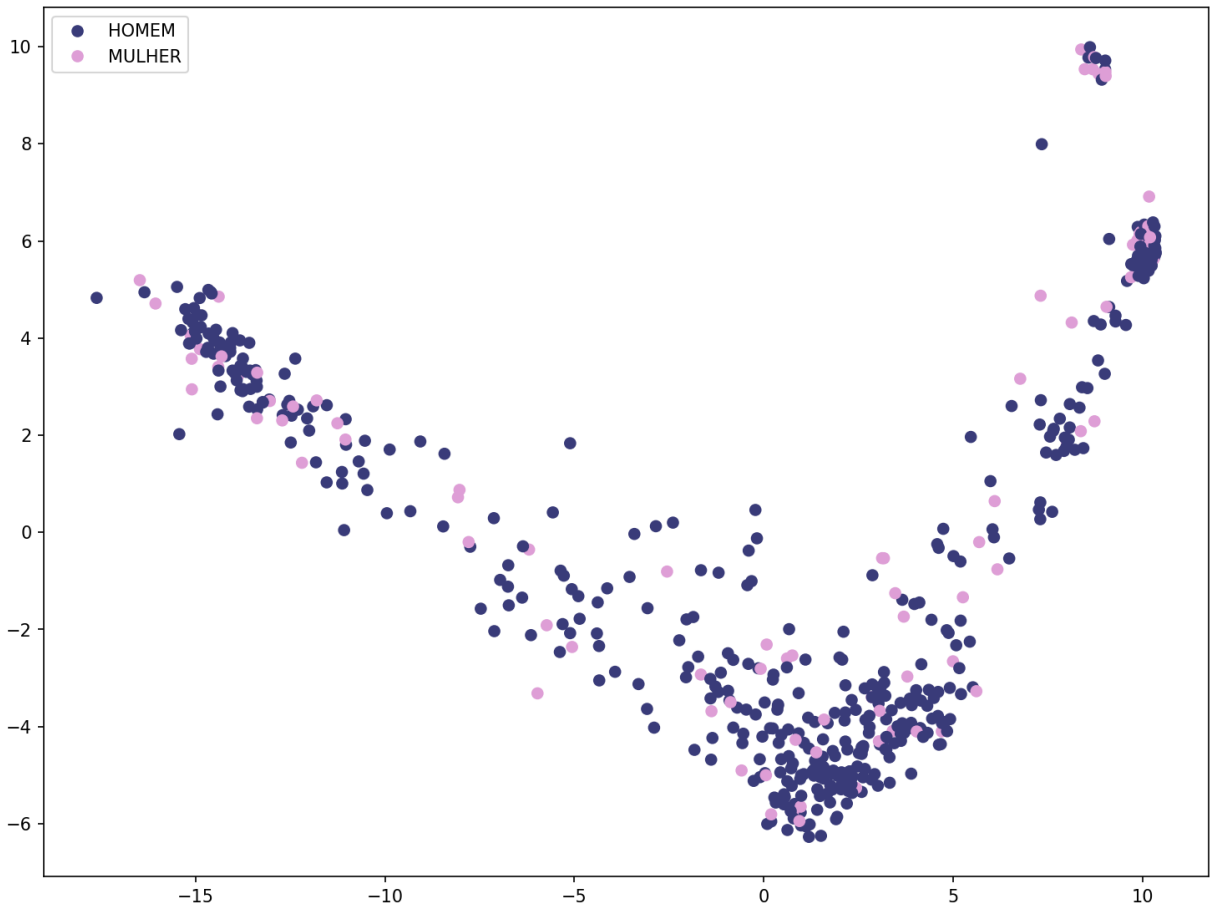
Quanto mais próximos dois parlamentares, mais parecidos foram seus votos em plenário no período. Fonte: autoria própria.

Figura 20 – Mapa ideológico da Câmara dos Deputados de 2023 por data de nascimento - PCA



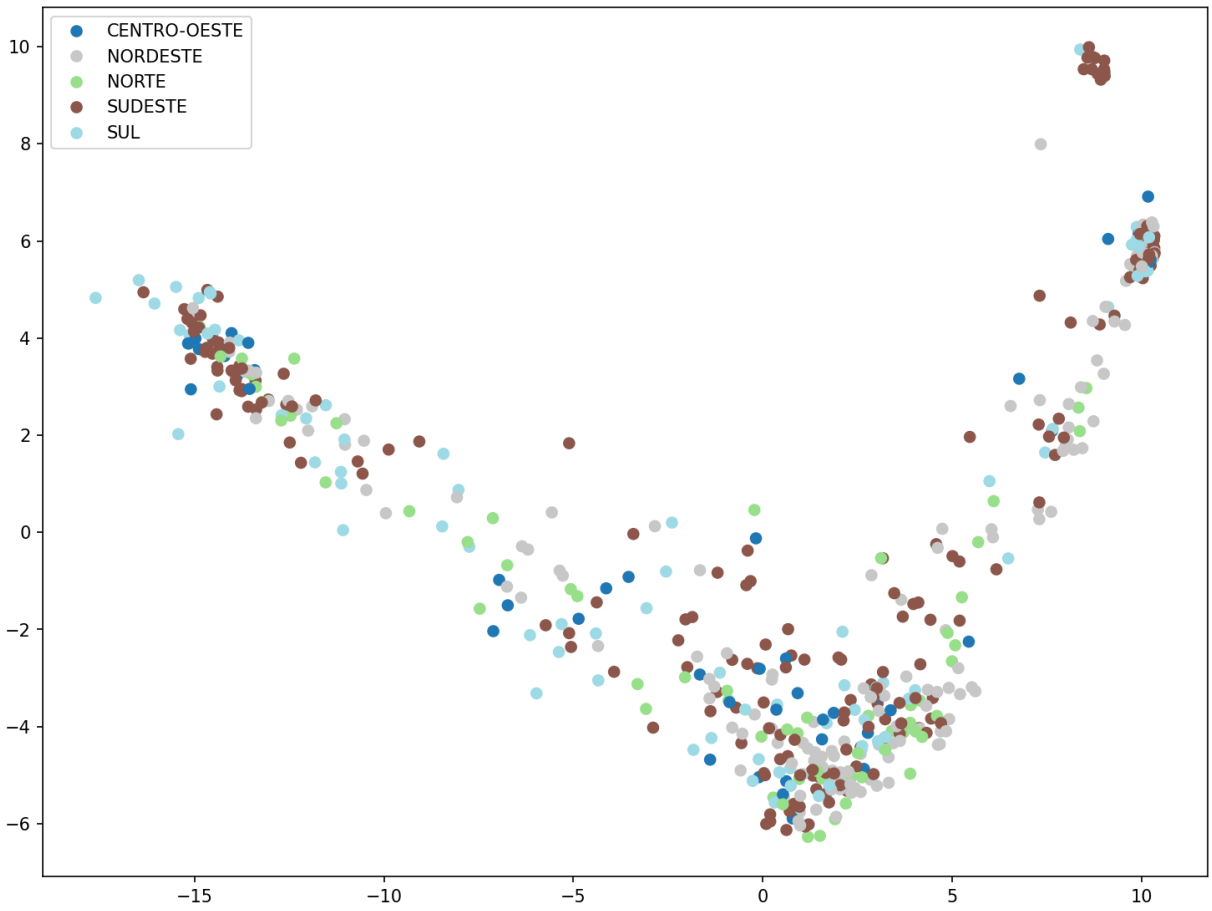
Quanto mais próximos dois parlamentares, mais parecidos foram seus votos em plenário no período. Fonte: autoria própria.

Figura 21 – Mapa ideológico da Câmara dos Deputados de 2023 por gênero - PCA



Quanto mais próximos dois parlamentares, mais parecidos foram seus votos em plenário no período. Fonte: autoria própria.

Figura 22 – Mapa ideológico da Câmara dos Deputados de 2023 por macrorregião - PCA



Quanto mais próximos dois parlamentares, mais parecidos foram seus votos em plenário no período. Fonte: autoria própria.

5.4 Validação dos resultados

Foram avaliados diversos períodos legislativos, com o objetivo de verificar a corretude dos dados coletados e a precisão dos modelos gerados. Os resultados foram comparados com os obtidos por outras ferramentas descritas neste trabalho e informações históricas. Os parâmetros utilizados em cada algoritmo foram:

- **PCA**: $whiten = False$.
- **t-SNE**: $perplexity = 30.0$, $early\_exaggeration = 12.0$, $learning\_rate = 50$, $max\_iter = 1000$.
- **MDS**: $metric = True$, $max\_iter = 300$, $eps = 0.001$.
- **UMAP**: $n\_neighbors = 15$, $min\_dist = 0.1$, $spread = 1.0$.
- **Propagação por Afinidade**: $damping = 0.7$, $max\_iter = 200$, $convergence\_iter = 15$.

- **HDBSCAN**: $min\_cluster\_size = 5$, $max\_cluster\_size = None$, $alpha = 1.0$, $leaf\_size = 40$.

Alguns dos períodos legislativos avaliados serão apresentados a seguir, com os respectivos mapas ideológicos. Foram utilizadas algumas métricas para medir a qualidade dos modelos gerados, entre elas:

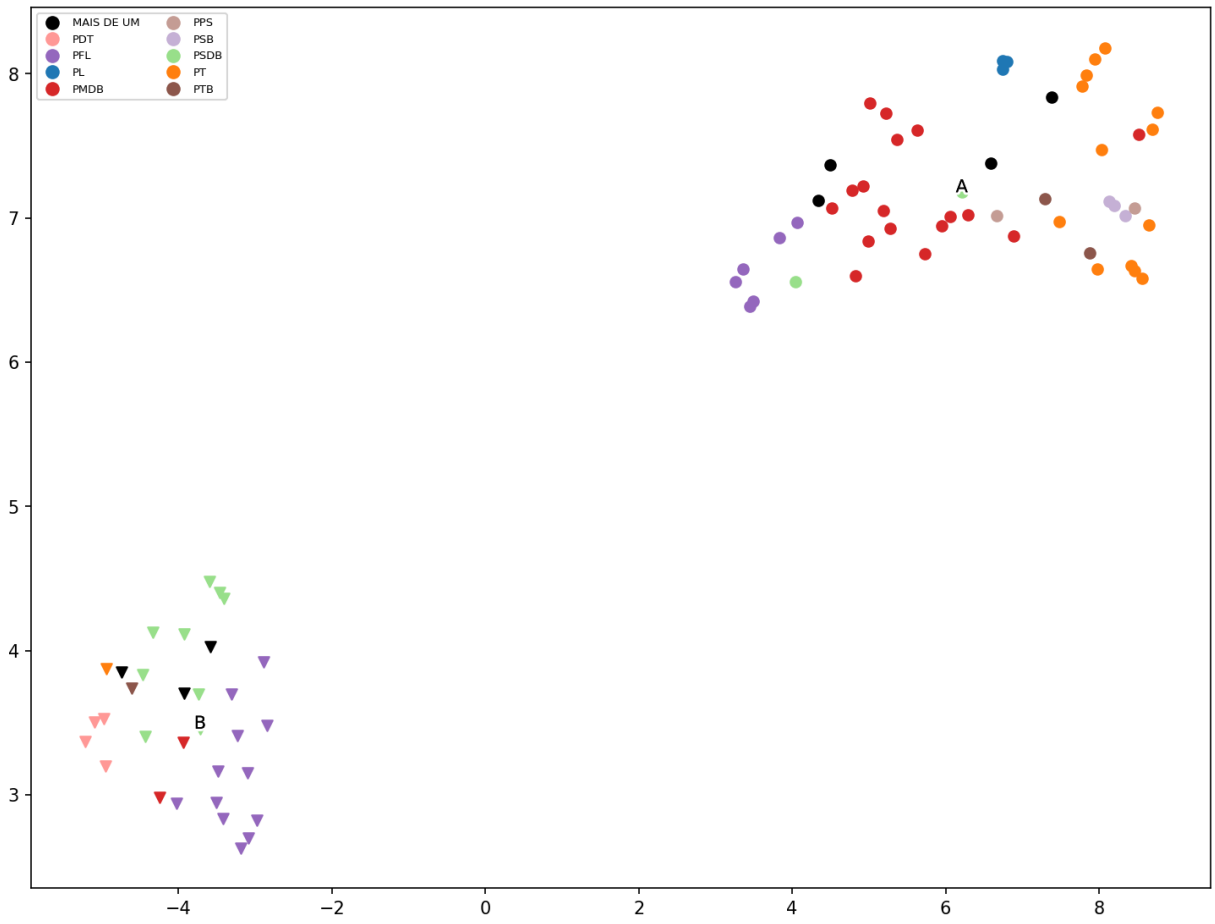
- **Variância total explicada**: A soma das variâncias explicadas por cada dimensão do modelo, nesse caso bidimensional. Um valor acima de 70% é usualmente considerado bom (JOLLIFFE; CADIMA, 2016).

- **Tensão normalizada pela escala (*Scale-normalized stress*)** (SMELSER; MILLER; KOBOUROV, 2024): A tensão indica quão bem o modelo preserva as distâncias originais entre os pontos. Para a tensão normalizada, um valor de 0% indica um modelo "perfeito", 2,5% excelente, 5% bom, 10% regular e acima de 20%, ruim (KRUSKAL, 1964).

- **Confiabilidade**: O quão bem foram preservadas as vizinhanças locais do espaço de alta dimensão. Um valor de 100% indica que todos os pares de pontos que eram vizinhos no espaço de alta dimensão continuam sendo vizinhos no espaço de baixa dimensão.

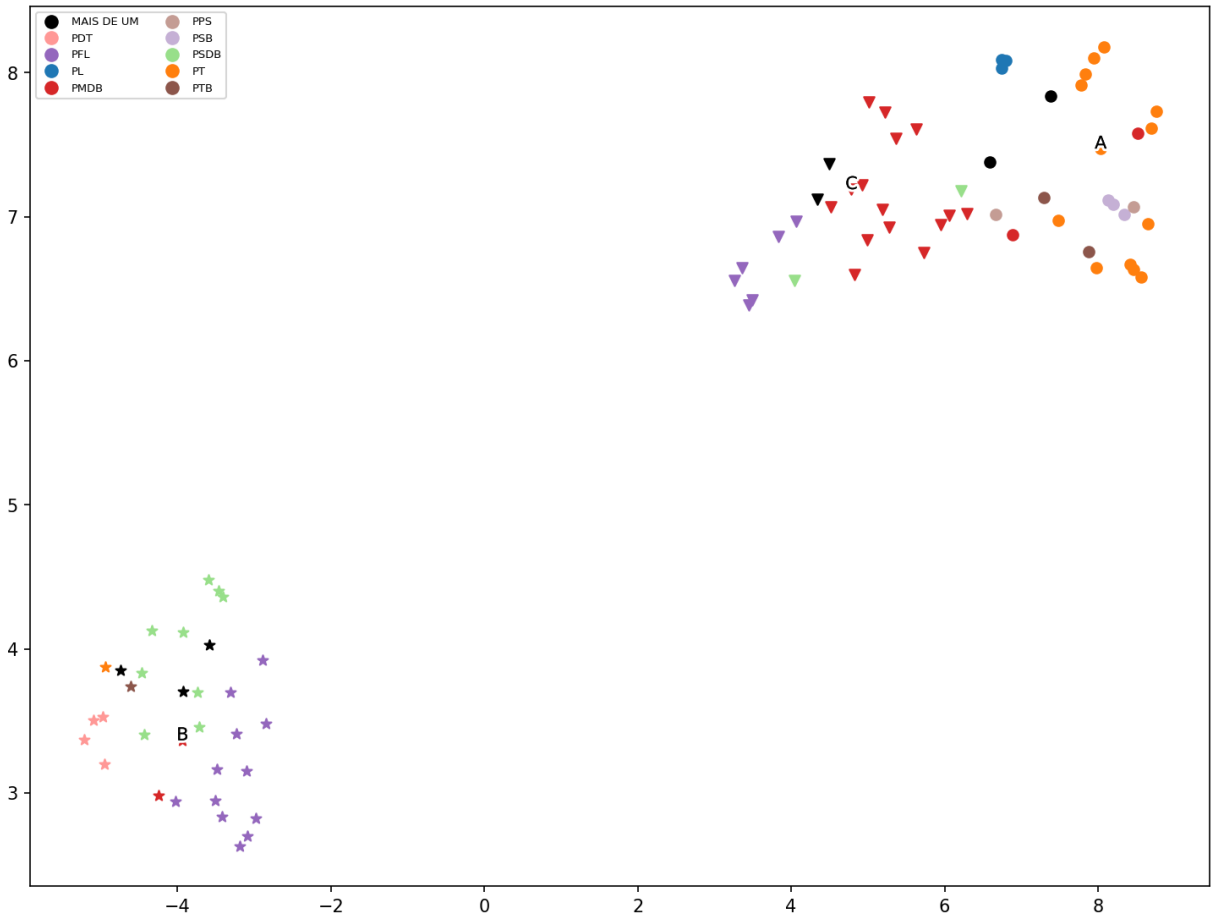
- **Homogeneidade partidária**: Quão homogêneo são os clusters em relação a afiliação partidária. Um valor de 100% indica que os clusters são compostos de membros de um único partido.

Figura 23 – Mapa ideológico do Senado Federal de 2003 - UMAP e HDBSCAN



As diferentes formas representam diferentes clusters e pontos menores representam outliers. Letras representam os centros dos diferentes clusters, onde (A) Eduardo Siqueira Campos, (B) Arthur Virgílio. Quanto mais próximos dois parlamentares, mais parecidos foram seus votos em plenário no período. Fonte: autoria própria.

Figura 24 – Mapa ideológico do Senado Federal de 2003 - UMAP e Propagação por Afinidade



As diferentes formas representam diferentes clusters e pontos menores representam outliers. Letras representam os centros dos diferentes clusters, onde (A) Ana Júlia Carepa, (B) Mão Santa, (C) Ramez Tebet. Quanto mais próximos dois parlamentares, mais parecidos foram seus votos em plenário no período. Fonte: autoria própria.

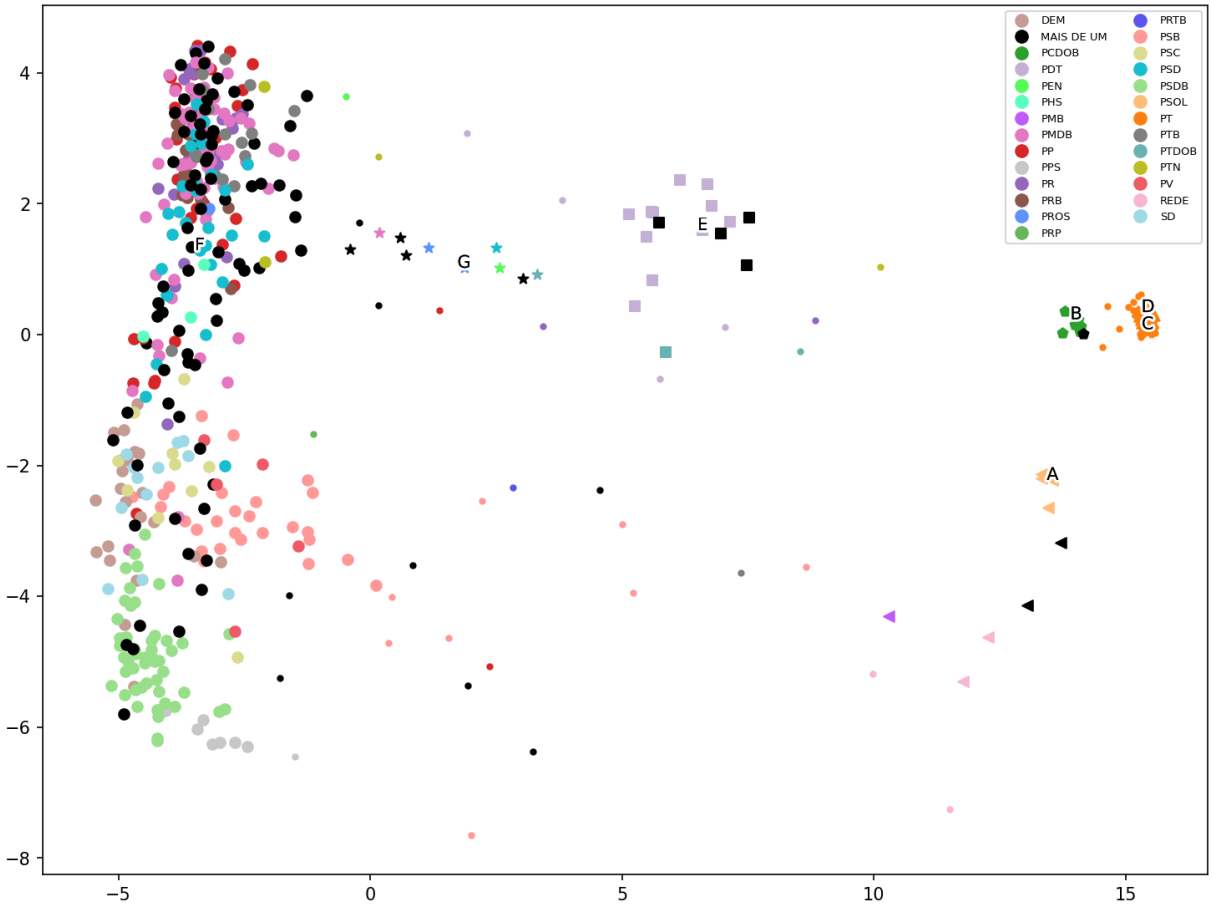
Tabela 3 – Métricas do modelo UMAP do Senado Federal de 2003

Métrica	**Valor**
Confiabilidade	94.4%
Outliers (HDBSCAN)	0%
Homogeneidade partidária (HDBSCAN)	13.2%
Homogeneidade partidária (Propagação por Afinidade)	29.1%

Em 2003, o acordo entre o PT e PMDB para dividir as presidências da Câmara e do Senado gerou uma reação imediata de exclusão por parte do PSDB, PFL e PDT, que, sem acesso ao poder no Congresso, articularam uma oposição ao governo eleito (ULHÔA; COSTA, 2002). Simultaneamente, o PFL sofria uma divisão interna, com parte do partido apoiando o governo e parte a oposição (RIBEIRO, 2014). É possível observar esses eventos

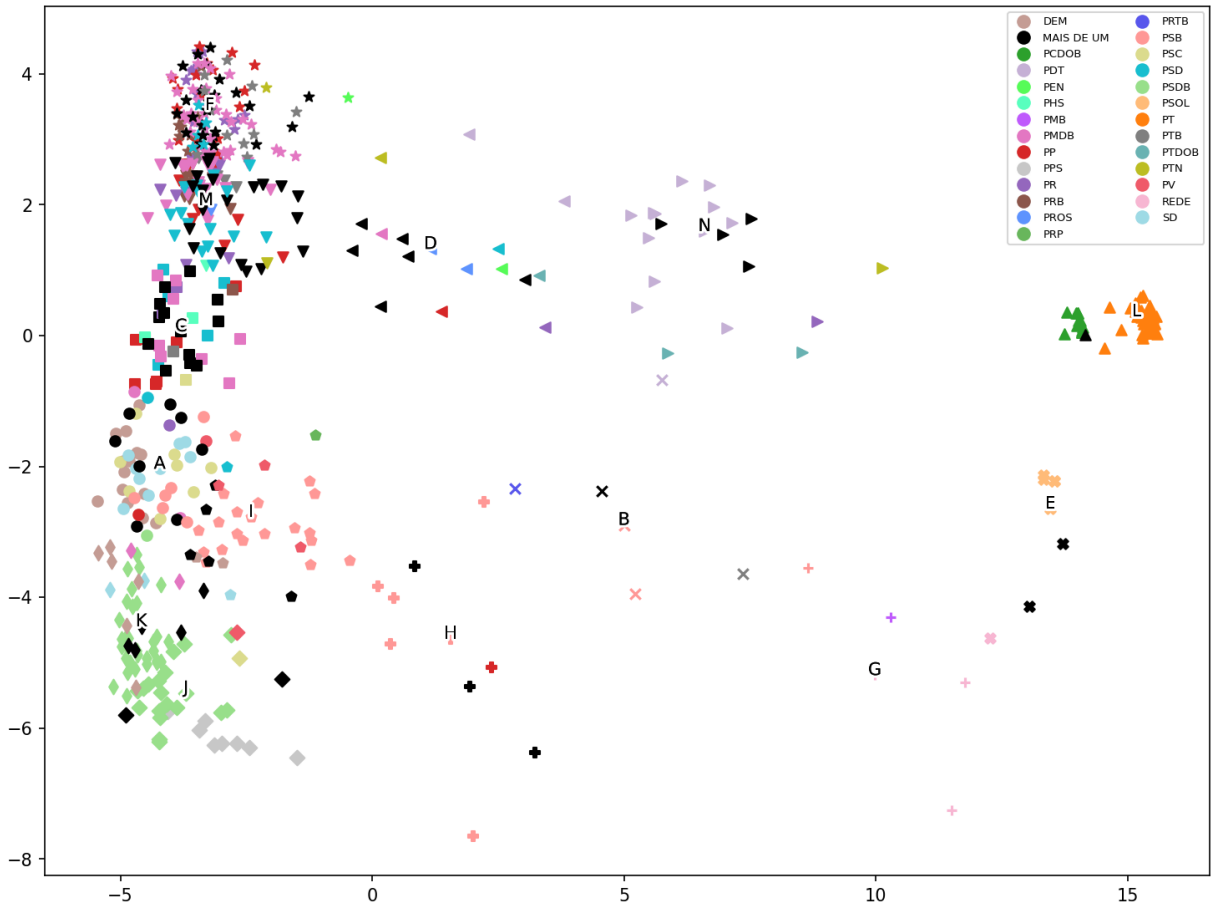
no mapa ideológico gerado (Figura 23 e 24), indicando que o algoritmo UMAP foi capaz de capturar a dinâmica política do período.

Figura 25 – Mapa ideológico da Câmara dos Deputados de 2016 - PCA e HDBSCAN



As diferentes formas representam diferentes clusters e pontos menores representam outliers. Letras representam os centros dos diferentes clusters, onde (A) Edmilson Rodrigues, (B) Wadson Ribeiro, (C) Adelmo Carneiro Leão, (D) Valmir Prascidelli, (E) Wolney Queiroz, (F) Heuler Cruvinel, (G) Bosco Costa. Quanto mais próximos dois parlamentares, mais parecidos foram seus votos em plenário no período. Fonte: autoria própria.

Figura 26 – Mapa ideológico da Câmara dos Deputados de 2016 - PCA e Propagação por Afinidade



As diferentes formas representam diferentes clusters e pontos menores representam outliers. Letras representam os centros dos diferentes clusters, onde (A) Benjamin Maranhão, (B) César Messias, (C) Felipe Bornier, (D) George Hilton, (E) Jean Wyllys, (F) Jorge Côrte Real, (G) João Derly, (H) Júlio Delgado, (I) Keiko Ota, (J) Max Filho, (K) Nelson Padovani, (L) Nelson Pellegrino, (M) Roberto Teixeira, (N) Wolney Queiroz. Quanto mais próximos dois parlamentares, mais parecidos foram seus votos em plenário no período. Fonte: autoria própria.

Tabela 4 – Métricas do modelo PCA da Câmara dos Deputados de 2016

Métrica	**Valor**
Variância explicada pelo eixo X	51.9%
Variância explicada pelo eixo Y	10%
Variância total explicada	61.9%
Outliers (HDBSCAN)	8.4%
Homogeneidade partidária (HDBSCAN)	23.9%
Homogeneidade partidária (Propagação por Afinidade)	45.1%

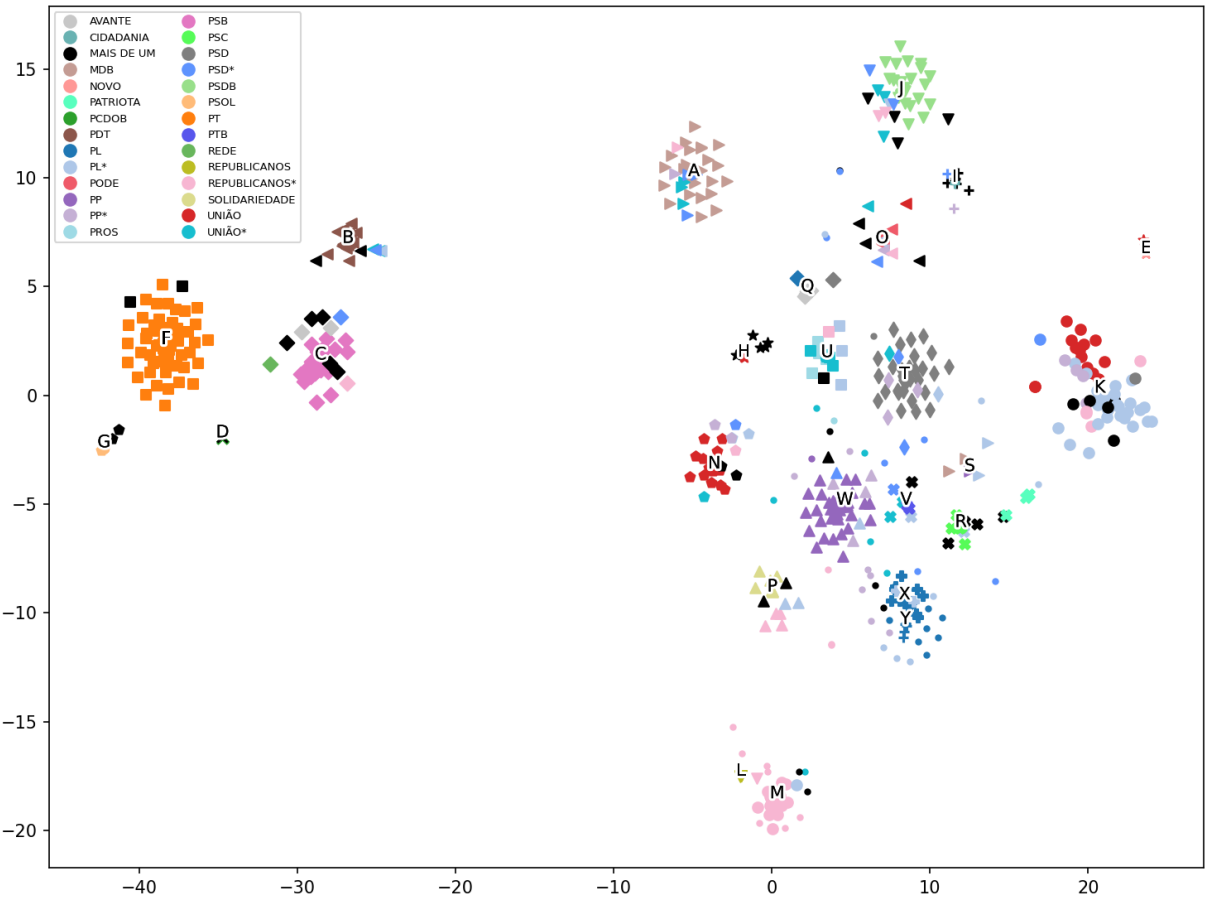
Em 2016, o processo de impeachment de Dilma Rousseff expôs o isolamento do PT e seus aliados no Congresso, especialmente diante da fragmentação partidária e

da perda de apoio de antigos parceiros de coalizão, como o PMDB, PP e PTB, cujos parlamentares votaram majoritariamente pelo impeachment de Dilma (PITOMBO, 2016). Os partidos PDT, PSOL, PC do B e Rede, por sua vez, votaram majoritariamente contra o impeachment, indicando maior proximidade ao governo.

Houve também uma grande troca partidária neste ano, impulsionada pela Emenda Constitucional nº 91, que abriu uma janela partidária (momento em que um legislador pode mudar de partido sem sofrer sanções) excepcional. Durante esse período, 92 dos 513 deputados trocaram de legenda (HAJE, 2016).

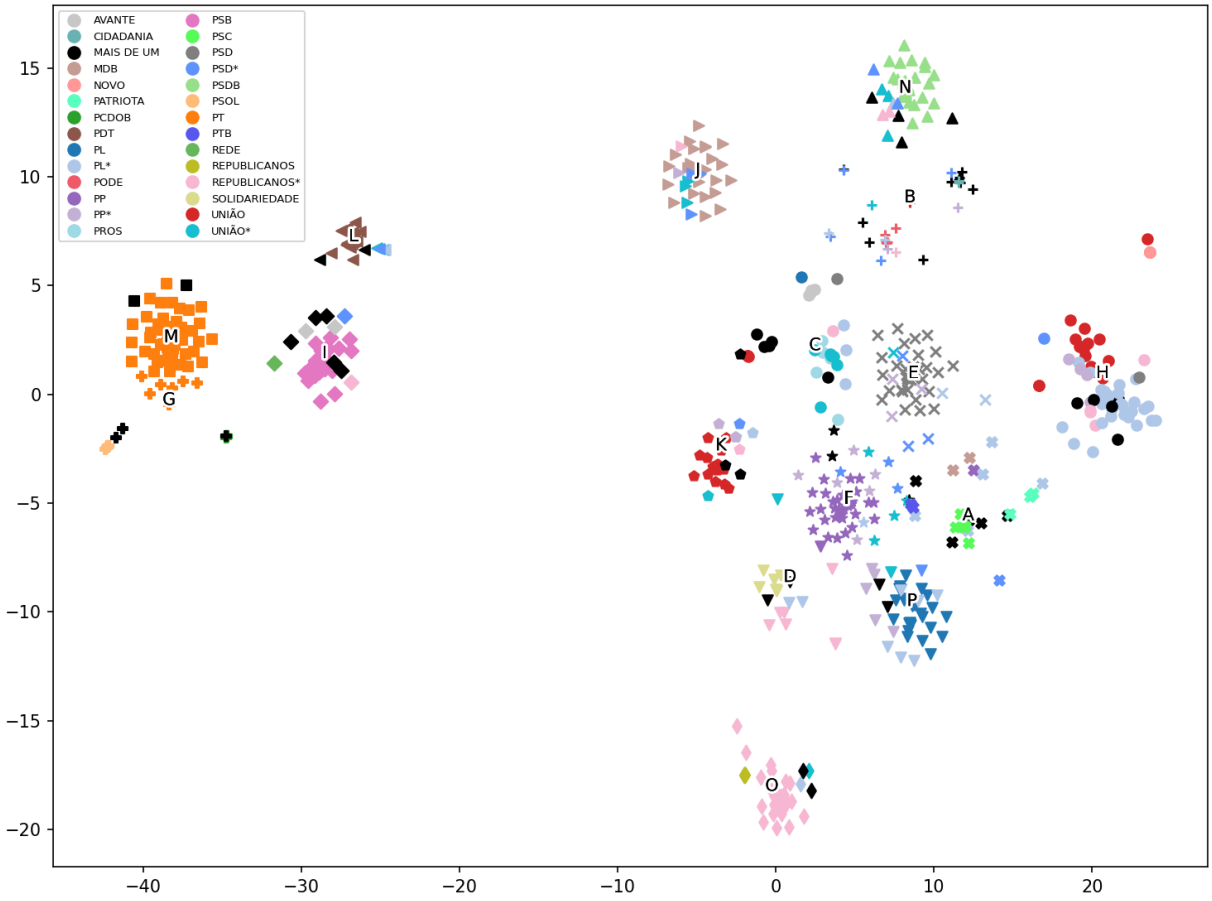
Com esse contexto, é possível observar a fragmentação partidária e a polarização política do período, assim como as mudanças na composição partidária, no mapa ideológico gerado pelo algoritmo PCA (Figura 25 e 26).

Figura 27 – Mapa ideológico da Câmara dos Deputados de 2019 a 2022 - t-SNE e HDBSCAN



As diferentes formas representam diferentes clusters e pontos menores representam outliers. Legendas com asterisco (*) denotam parlamentares que entraram no partido após o início do período. Letras representam os centros dos diferentes clusters, onde (A) Leonardo Picciani, (B) Mauro Benevides Filho, (C) João H. Campos, (D) Márcio Jerry, (E) Alexis Fonteyne, (F) Rachel Marques, (G) Áurea Carolina, (H) Zé Augusto Nalin, (I) Joceval Rodrigues, (J) Rafafá, (K) Márcio Labre, (L) Henrique do Paraíso, (M) Maria Rosas, (N) Juninho do Pneu, (O) Sargento Alexandre, (P) Ottaci Nascimento, (Q) Fernando Borja, (R) Pedro Dalua, (S) Evair Vieira de Melo, (T) Fábio Faria, (U) Dr. Agripino Magalhães, (V) Santini, (W) Marco Brasil, (X) Junior Lourenço, (Y) Gorete Pereira. Quanto mais próximos dois parlamentares, mais parecidos foram seus votos em plenário no período. Fonte: autoria própria.

Figura 28 – Mapa ideológico da Câmara dos Deputados de 2019 a 2022 - t-SNE e Propagação por Afinidade



As diferentes formas representam diferentes clusters e pontos menores representam outliers. Legendas com asterisco (*) denotam parlamentares que entraram no partido após o início do período. Letras representam os centros dos diferentes clusters, onde (A) Aluisio Mendes, (B) Bozzella, (C) Clarissa Garotinho, (D) Dra. Vanda Milani, (E) Edilázio Júnior, (F) Eliza Virgínia, (G) Erika Kokay, (H) General Peternelli, (I) João H. Campos, (J) Leonardo Picciani, (K) Marcos Soares, (L) Mário Heringer, (M) Nelson Pellegrino, (N) Roberto Pessoa, (O) Vinicius Carvalho, (P) Vinicius Gurgel. Quanto mais próximos dois parlamentares, mais parecidos foram seus votos em plenário no período. Fonte: autoria própria.

Tabela 5 – Métricas do modelo t-SNE da Câmara dos Deputados de 2019 a 2022

Métrica	**Valor**
Confiabilidade	94.9%
Outliers (HDBSCAN)	8.7%
Homogeneidade partidária (HDBSCAN)	80.6%
Homogeneidade partidária (Propagação por Afinidade)	70.5%

Em 2019, a composição partidária do Congresso sofreu mudanças significativas. O PSL, partido associado à candidatura de Jair Bolsonaro, que eventualmente se uniria ao

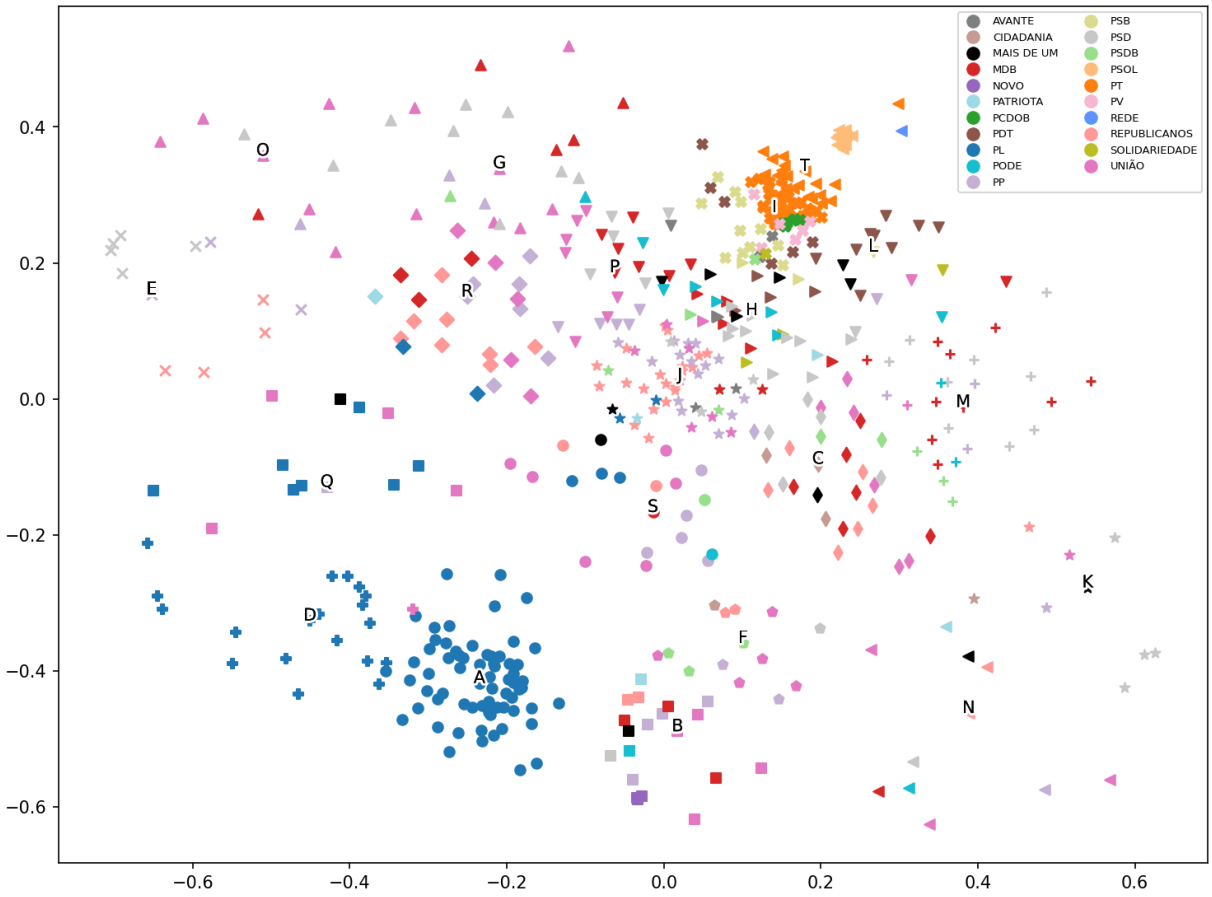
DEM para formar o partido União, emergiu como uma força expressiva na Câmara dos Deputados, juntamente com um aumento no número de cadeiras dos partidos PP, PSD e Republicanos (CAESAR, 2018). Mais tarde, ao romper com o PSL, legisladores alinhados ao presidente migraram para o PL e partidos aliados (PORTO et al., 2022).

A oposição ao governo Bolsonaro foi liderada pelos partidos PT, PSB, PDT, PSOL, PC do B e REDE, que votaram contra o governo em mais de 50% das ocasiões (ESTADÃO, 2019). O mapa ideológico gerado pelo algoritmo t-SNE (Figura 27 e 28) exibe uma clara divisão entre os parlamentares alinhados ao governo e à oposição, e as mudanças na composição partidária são evidenciadas.

Figura 29 – Mapa ideológico da Câmara dos Deputados de 2023 - MDS e HDBSCAN

As diferentes formas representam diferentes clusters e pontos menores representam outliers. Letras representam os centros dos diferentes clusters, onde (A) Leandre, (B) Alberto Fraga, (C) Vermelho, (D) Pezenti, (E) Tarcísio Motta, (F) Diego Andrade, (G) Mauricio Neves, (H) Lebrão, (I) Vitor Lippi, (J) Eduardo da Fonte, (K) Gustinho Ribeiro, (L) Waldemar Oliveira, (M) Pastor Sargento Isidório, (N) Padre João. Quanto mais próximos dois parlamentares, mais parecidos foram seus votos em plenário no período. Fonte: autoria própria.

Figura 30 – Mapa ideológico da Câmara dos Deputados de 2023 - MDS e Propagação por Afinidade



As diferentes formas representam diferentes clusters e pontos menores representam outliers. Letras representam os centros dos diferentes clusters, onde (A) Alberto Fraga, (B) Alfredo Gaspar, (C) Amom Mandel, (D) André Ferreira, (E) Arthur Lira, (F) Daniel Trzeciak, (G) Danilo Forte, (H) Diego Coronel, (I) Dimas Gadelha, (J) Gilberto Abramo, (K) Gilberto Nascimento, (L) Heitor Schuch, (M) Hercílio Coelho Diniz, (N) Julio Cesar Ribeiro, (O) Luciano Bivar, (P) Olival Marques, (Q) Pedro Lupion, (R) Pinheirinho, (S) Thiago Flores, (T) Waldenor Pereira. Quanto mais próximos dois parlamentares, mais parecidos foram seus votos em plenário no período. Fonte: autoria própria.

Tabela 6 – Métricas do modelo MDS da Câmara dos Deputados de 2023

Métrica	**Valor**
Tensão normalizada pela escala	25.3%
Outliers (HDBSCAN)	45.9%
Homogeneidade partidária (HDBSCAN)	35.1%
Homogeneidade partidária (Propagação por Afinidade)	47.7%

Por fim, em 2023, observou-se uma reconfiguração significativa das alianças políticas no Brasil. Partidos que anteriormente compunham a base de apoio ao governo de Jair Bolsonaro passaram a integrar a base governista do PT, agora no comando do governo

federal. No entanto, essa nova adesão ocorreu de forma bastante fragmentada e heterogênea, refletindo um cenário de alinhamento pragmático ao invés de ideológico (MATTOS, 2024). Estas características, juntamente com a oposição por parte do PL, são refletidas no mapa ideológico gerado pelo algoritmo MDS (Figura 29 e 30).

6 Conclusão

Este trabalho teve como objetivo geral o desenvolvimento de uma solução flexível e de fácil utilização para visualização e análise de votações nominais, oferecendo uma ferramenta acessível a pesquisadores da ciência política, legisladores e cidadãos. O objetivo foi parcialmente atingido, pois, embora tenha havido um esforço significativo para tornar a biblioteca acessível, a solução acabou focando principalmente em pesquisadores, um público que não era plenamente atendido pelas ferramentas existentes. No entanto, com os exemplos práticos e a documentação fornecida, a biblioteca ainda pode ser utilizada por qualquer pessoa interessada em explorar votações nominais, incluindo cidadãos com interesse na área. Em relação aos objetivos específicos:

O objetivo específico O1, que consiste em pesquisar as vantagens e desvantagens dos diversos algoritmos de análise de dados quando aplicados no contexto político, foi plenamente atingido. Nota-se que os modelos produzidos pelo algoritmo PCA apresentaram consistentemente uma variância total explicada abaixo do ideal (Tabela 7). Esse desempenho limitado pode ser atribuído a características do cenário político brasileiro, como a presença de um "centrão", que tende a tornar os dados mais esféricos, e a diversidade de partidos, que dificulta a separação linear dos dados.

Esses fatores podem impactar a efetividade do PCA, especialmente quando comparado ao uso desse método em contextos como o do Congresso dos Estados Unidos. Além disso, ao ser aplicado a períodos prolongados, o PCA pode introduzir artefatos devido à não linearidade dos dados, resultante de mudanças no posicionamento relativo dos parlamentares ao longo do tempo, como observado em 2023. Em contrapartida, o PCA apresentou o melhor desempenho computacional entre os métodos testados.

Tabela 7 – Média das métricas dos modelos gerados pelo PCA

Métrica	**Câmara**		**Senado**	
	Média	**Desvio padrão**	**Média**	**Desvio padrão**
Variância total explicada	55%	5%	43.5%	5.8%
Outliers (HDBSCAN)	24.9%	13.2%	18.8%	13.9%
Homogeneidade partidária (HDBSCAN)	46.5%	10.6%	35.4%	9.5%
Homogeneidade partidária (Propagação por Afinidade)	54.8%	7%	47.8%	5.4%

Média das métricas de 6 modelos, dos períodos 2003-2006 (Lula 1), 2007-2010 (Lula 2), 2011-2014 (Dilma 1), 2015-2016 (Dilma 2), 2016-2018 (Temer) e 2019-2022 (Bolsonaro).

Semelhantemente, os modelos gerados pelo MDS, especificamente da Câmara

dos Deputados, apresentaram uma tensão acima do ideal (Tabela 8). Além disso, uma quantidade significativa de outliers foi observada ao ser clusterizado com HDBSCAN, sugerindo que este algoritmo de clusterização teve dificuldade com a natureza esparsa dos modelos gerados pelo MDS.

Tabela 8 – Média das métricas dos modelos gerados pelo MDS

Métrica	**Câmara**		**Senado**	
	Média	**Desvio padrão**	**Média**	**Desvio padrão**
Tensão normalizada pela escala	26.5%	2.5%	19.9%	2.2%
Outliers (HDBSCAN)	31.6%	15.9%	20.5%	14%
Homogeneidade partidária (HDBSCAN)	39.7%	8.6%	23.3%	10%
Homogeneidade partidária (Propagação por Afinidade)	56.4%	6.2%	55.9%	3.6%

Média das métricas de 6 modelos, dos períodos 2003-2006 (Lula 1), 2007-2010 (Lula 2), 2011-2014 (Dilma 1), 2015-2016 (Dilma 2), 2016-2018 (Temer) e 2019-2022 (Bolsonaro).

Em contraste, tanto o t-SNE quanto o UMAP geraram mapas com confiabilidade superior a 90% na maioria dos períodos analisados (Tabelas 9 e 10). Além disso, ambos os algoritmos foram mais eficazes em capturar a estrutura partidária dos dados, como evidenciado pela maior homogeneidade partidária nos clusters gerados. Entretanto, esses algoritmos apresentaram o pior desempenho computacional.

Tabela 9 – Média das métricas dos modelos gerados pelo t-SNE

Métrica	**Câmara**		**Senado**	
	Média	**Desvio padrão**	**Média**	**Desvio padrão**
Confiabilidade	94.9%	2%	92.4%	2%
Outliers (HDBSCAN)	12.6%	9.6%	21.7%	9%
Homogeneidade partidária (HDBSCAN)	77.4%	7%	44.7%	11.4%
Homogeneidade partidária (Propagação por Afinidade)	70.3%	1.6%	56%	7.2%

Média das métricas de 6 modelos, dos períodos 2003-2006 (Lula 1), 2007-2010 (Lula 2), 2011-2014 (Dilma 1), 2015-2016 (Dilma 2), 2016-2018 (Temer) e 2019-2022 (Bolsonaro).

Tabela 10 – Média das métricas dos modelos gerados pelo UMAP

Métrica	**Câmara**		**Senado**	
	Média	**Desvio padrão**	**Média**	**Desvio padrão**
Confiabilidade	92.4%	2.7%	90.3%	2.1%
Outliers (HDBSCAN)	2.4%	2.9%	5%	8.5%
Homogeneidade partidária (HDBSCAN)	78.6%	5.4%	38.5%	19.8%
Homogeneidade partidária (Propagação por Afinidade)	70%	7.8%	50.6%	10.9%

Média das métricas de 6 modelos, dos períodos 2003-2006 (Lula 1), 2007-2010 (Lula 2), 2011-2014 (Dilma 1), 2015-2016 (Dilma 2), 2016-2018 (Temer) e 2019-2022 (Bolsonaro).

Quanto aos algoritmos de clusterização, o HDBSCAN produziu clusters mais significativos, que melhor representaram as coalizões políticas presentes. No entanto, nenhum dos algoritmos foi capaz de gerar clusters satisfatórios quando parametrizados para classificar os parlamentares em grupos mais amplos, como governo e oposição, ou em blocos ideológicos. Para esses casos, pode ser necessário o uso de algoritmos onde o número de clusters seja definido a priori, permitindo uma análise mais direcionada.

O objetivo específico O2, que consiste em desenvolver uma solução para a visualização e análise de votações nominais, também foi plenamente atingido. A biblioteca desenvolvida cumpriu seu objetivo como uma solução para a visualização e análise de votações nominais, fornecendo uma API amigável e flexiva para a exploração dos dados abertos disponibilizados pela Câmara dos Deputados e Senado Federal. A biblioteca foi disponibilizada publicamente em um repositório no GitHub[1] e na plataforma PyPI[2], permitindo que qualquer pessoa possa utilizá-la e contribuir com o seu desenvolvimento.

Por fim, o objetivo específico O3, que abrange a validação da solução através de dados reais de diferentes contextos políticos, foi plenamente atingido. Os modelos gerados pela biblioteca foram validados em diferentes contextos políticos, fornecendo uma visão detalhada e precisa da dinâmica parlamentar. Os resultados obtidos foram consistentes com o cenário político brasileiro, refletindo a fragmentação partidária, a polarização política e as mudanças na composição partidária ao longo dos anos.

6.1 Trabalhos Futuros

Alguns estados, como São Paulo[3], disponibilizam dados de votações nominais de suas assembleias legislativas, o que abre a possibilidade de incluir suporte a essas fontes

[1] https://github.com/mateuskreuch/branalysis
[2] https://pypi.org/project/branalysis/
[3] https://www.al.sp.gov.br/alesp/votacoes-no-plenario/

na biblioteca. Além disso, integrar outros dados políticos, como proposições legislativas, discursos, gastos parlamentares e informações complementares, em uma API unificada, facilitaria significativamente a análise política e aumentaria o valor da ferramenta para pesquisadores e cidadãos interessados em uma visão mais ampla e detalhada da atuação parlamentar.

Ademais, seria relevante explorar o uso de outros algoritmos de clusterização, como o agrupamento espectral (*spectral clustering*) e o agrupamento aglomerativo (*agglomerative clustering*), que permitem definir o número de clusters a priori. Isso abriria a possibilidade de realizar análises de grupos mais amplos de parlamentares, como discutido anteriormente. Por fim, a biblioteca poderia fornecer mais facilidades em relação a criação de outros tipos de gráficos estatísticos.

Referências

ABDULJABER, M. A dimension reduction method application to a political science question: Using exploratory factor analysis to generate the dimensionality of political ideology in the Arab World. *Journal of Information & Knowledge Management*, World Scientific, v. 19, n. 01, p. 2040002, 2020. Citado na página 20.

AHMED, M.; SERAJ, R.; ISLAM, S. M. S. The k-means algorithm: A comprehensive survey and performance evaluation. *Electronics*, MDPI, v. 9, n. 8, p. 1295, 2020. Citado na página 38.

ARORA, S.; HU, W.; KOTHARI, P. K. An analysis of the t-sne algorithm for data visualization. In: PMLR. *Conference on learning theory*. [S.l.], 2018. p. 1455–1462. Citado na página 22.

BACHMANN, F.; HENNIG, P.; KOBAK, D. Wasserstein t-SNE. In: SPRINGER. *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. [S.l.], 2022. p. 104–120. Citado na página 22.

BORG, I.; GROENEN, P. J. *Modern multidimensional scaling: Theory and applications*. [S.l.]: Springer Science & Business Media, 2007. Citado na página 20.

BORJA, F. G. de; FREITAS, C. M. CivisAnalysis: Interactive visualization for exploring roll call data and representatives' voting behaviour. In: IEEE. *2015 28th SIBGRAPI Conference on Graphics, Patterns and Images*. [S.l.], 2015. p. 257–264. Citado 2 vezes nas páginas 31 e 39.

BRASIL. *[Constituição (1988)]. Constituição da República Federativa do Brasil*. Brasília, DF: Câmara dos Deputados, Edições Câmara, 2023. 65. ed. Disponível em: <https://bd.camara.leg.br/bd/bitstream/handle/bdcamara/15261/constituicao_federal_65ed.pdf>. Acesso em: 1 jun. 2024. Citado na página 17.

BRIGADIR, I. et al. Dimensionality reduction and visualisation tools for voting record. In: CEUR WORKSHOP PROCEEDINGS. *Greene, D., Mac Namee, B. and Ross, R.(eds.). Proceedings of the 24th Irish Conference on Artificial Intelligence and Cognitive Science*. [S.l.], 2016. Citado 2 vezes nas páginas 13 e 22.

CAESAR, G. *Saiba como eram e como ficaram as bancadas na Câmara dos Deputados, partido a partido*. [S.l.]: G1, 2018. <https://g1.globo.com/politica/eleicoes/2018/eleicao-em-numeros/noticia/2018/10/08/pt-perde-deputados-mas-ainda-tem-maior-bancada-da-camara-psl-de-bolsonaro-ganha-52-representan ghtml>. Acesso em: 1 nov. 2024. Citado na página 60.

CAMPELLO, R. J.; MOULAVI, D.; SANDER, J. Density-based clustering based on hierarchical density estimates. In: SPRINGER. *Pacific-Asia conference on knowledge discovery and data mining*. [S.l.], 2013. p. 160–172. Citado na página 23.

CARMINES, E. G.; STIMSON, J. A. Racial issues and the structure of mass belief systems. *The Journal of Politics*, Southern Political Science Association, v. 44, n. 1, p. 2–20, 1982. Citado na página 21.

CARROLL, R. *Ideal Point Estimation*. 2023. <https://www.oxfordbibliographies.com/display/document/obo-9780199756223/obo-9780199756223-0360.xml>. Acesso em: 24 mai. 2024. Citado na página 20.

CARROLL, R.; POOLE, K. Roll call analysis and the study of legislatures. *The Oxford handbook of legislative studies*, Oxford University Press Oxford, p. 103–125, 2014. Citado na página 25.

DARWISH, K. et al. Unsupervised user stance detection on Twitter. In: *Proceedings of the international AAAI conference on web and social media*. [S.l.: s.n.], 2020. v. 14, p. 141–152. Citado na página 23.

DIACONIS, P.; GOEL, S.; HOLMES, S. Horseshoes in multidimensional scaling and local kernel methods. 2008. Citado na página 21.

DIAZ-PAPKOVICH, A.; ANDERSON-TROCMÉ, L.; GRAVEL, S. A review of UMAP in population genetics. *Journal of Human Genetics*, Springer Singapore Singapore, v. 66, n. 1, p. 85–91, 2021. Citado na página 22.

DUECK, D. *Affinity propagation: clustering data by passing messages*. [S.l.]: University of Toronto Toronto, ON, Canada, 2009. Citado na página 24.

ENELOW, J. M.; HINICH, M. J. *The spatial theory of voting: An introduction*. [S.l.]: CUP Archive, 1984. Citado 2 vezes nas páginas 20 e 25.

ESTADÃO. *Basômetro: quanto apoio o governo tem na Câmara?* 2019. <https://arte.estadao.com.br/politica/basometro/>. Acesso em: 6 mai. 2024. Citado 3 vezes nas páginas 27, 39 e 60.

ESTER, M. et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In: *kdd*. [S.l.: s.n.], 1996. v. 96, n. 34, p. 226–231. Citado na página 23.

FILHO, D. B. F. et al. Cluster analysis for political scientists. *Applied Mathematics*, Scientific Research Publishing, v. 2014, 2014. Citado na página 23.

FREY, B. J.; DUECK, D. Clustering by passing messages between data points. *science*, American Association for the Advancement of Science, v. 315, n. 5814, p. 972–976, 2007. Citado na página 23.

GROENEN, P.; CASTELEIN, A.; BRUIN, J. de. Data Visualization using t-SNE and Local Multidimensional Scaling. 2020. Citado na página 22.

HAGOPIAN, F.; MAINWARING, S. P. *The third wave of democratization in Latin America: advances and setbacks*. [S.l.]: Cambridge University Press, 2005. Citado na página 14.

HAJE, L. *Janela para troca partidária permitiu mais de 90 mudanças entre legendas*. [S.l.]: Agência Câmara de Notícias, 2016. <https://www.camara.leg.br/noticias/483820-janela-para-troca-partidaria-permitiu-mais-de-90-mudancas-entre-legendas/>. Acesso em: 1 nov. 2024. Citado na página 57.

HOTELLING, H. Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, Warwick & York, v. 24, n. 6, p. 417, 1933. Citado 2 vezes nas páginas 20 e 21.

JOLLIFFE, I. T. Principal component analysis. *Technometrics*, American Society for Quality, v. 45, n. 3, p. 276, 2003. Citado na página 21.

JOLLIFFE, I. T.; CADIMA, J. Principal component analysis: a review and recent developments. *Philosophical transactions of the royal society A: Mathematical, Physical and Engineering Sciences*, the Royal Society publishing, v. 374, n. 2065, p. 20150202, 2016. Citado 2 vezes nas páginas 21 e 52.

KRIESI, H. et al. Globalization and the transformation of the national political space: Six European countries compared. *European Journal of Political Research*, Wiley Online Library, v. 45, n. 6, p. 921–956, 2006. Citado na página 21.

KRUSKAL, J. B. Nonmetric multidimensional scaling: a numerical method. *Psychometrika*, Springer-Verlag New York, v. 29, n. 2, p. 115–129, 1964. Citado na página 52.

KULLBACK, S.; LEIBLER, R. A. On information and sufficiency. *The annals of mathematical statistics*, JSTOR, v. 22, n. 1, p. 79–86, 1951. Citado na página 22.

LEE, L.; ZHANG, S.; YANG, V. C. Do two parties represent the US? Clustering analysis of US public ideology survey. *arXiv preprint arXiv:1710.09347*, 2017. Citado na página 13.

LEITE, L.; TRENTO, S. Análise de votações nominais do legislativo brasileiro utilizando componentes principais. *Leviathan (São Paulo)*, n. 12, p. 120–163, 2016. Citado 3 vezes nas páginas 21, 28 e 39.

LESTER, J. C. The evolution of the political compass (and why libertarianism is not right-wing). *Journal of Social and Evolutionary Systems*, Elsevier, v. 17, n. 3, p. 231–241, 1994. Citado na página 25.

LEWIS, J. B. et al. *Voteview: Congressional roll-call votes database*. 2019. <https://voteview.com>. Acesso em: 8 jun. 2024. Citado na página 28.

LINCOLN, S.; PIEPE, A.; PRIOR, R. An application of principal components analysis to voting in Scottish Municipal elections 1967-9. *The Statistician*, JSTOR, p. 73–88, 1971. Citado na página 21.

MAATEN, L. V. D. et al. Dimensionality reduction: A comparative review. *Journal of Machine Learning Research*, v. 10, n. 66-71, p. 13, 2009. Citado na página 20.

MAATEN, L. Van der; HINTON, G. Visualizing data using t-SNE. *Journal of machine learning research*, v. 9, n. 11, 2008. Citado 2 vezes nas páginas 20 e 21.

MAGYAR, Z. B. What makes party systems different? A principal component analysis of 17 advanced democracies 1970–2013. *Political Analysis*, Cambridge University Press, v. 30, n. 2, p. 250–268, 2022. Citado na página 21.

MATTILA, M.; LANE, J.-E. Why unanimity in the Council? A roll call analysis of Council voting. *European Union Politics*, SAGE Publications, v. 2, n. 1, p. 31–52, 2001. Citado na página 21.

MATTOS, M. *Hoje aliados, partidos que dividem o governo não garantem apoio em 2026*. [S.l.]: Veja, 2024. <https://veja.abril.com.br/politica/hoje-aliados-partidos-que-dividem-o-governo-nao-garantem-apoio-em-2026>. Acesso em: 1 nov. 2024. Citado na página 62.

MCINNES, L.; HEALY, J.; MELVILLE, J. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018. Citado 2 vezes nas páginas 20 e 22.

PEZZOTTI, N. et al. Approximated and user steerable tSNE for progressive visual analytics. *IEEE transactions on visualization and computer graphics*, IEEE, v. 23, n. 7, p. 1739–1752, 2016. Citado na página 22.

PITOMBO, J. P. *Isolado, PT lava as mãos em 12 capitais no segundo turno*. [S.l.]: Folha de S. Paulo, 2016. <https://www1.folha.uol.com.br/poder/eleicoes-2016/2016/10/1822297-isolado-pt-lava-as-maos-em-12-capitais-no-segundo-turno.shtml>. Acesso em: 1 nov. 2024. Citado na página 57.

POOLE, K. T.; ROSENTHAL, H. A spatial model for legislative roll call analysis. *American journal of political science*, JSTOR, p. 357–384, 1985. Citado 3 vezes nas páginas 13, 20 e 29.

PORTO, D. et al. *Eleições 2022: veja quem mudou de legenda no fim da janela partidária*. [S.l.]: CNN, 2022. <https://www.cnnbrasil.com.br/politica/eleicoes-2022-veja-quem-mudou-de-legenda-no-fim-da-janela-partidaria/>. Acesso em: 1 nov. 2024. Citado na página 60.

RASHED, A. et al. Embeddings-based clustering for target specific stances: The case of a polarized turkey. In: *Proceedings of the International AAAI Conference on web and social media*. [S.l.: s.n.], 2021. v. 15, p. 537–548. Citado na página 23.

RIBEIRO, R. L. M. Decadência longe do poder: refundação e crise do PFL. *Revista de Sociologia e Política*, SciELO Brasil, v. 22, p. 5–37, 2014. Citado na página 54.

SANTOS, M. dos; ANDRADE, N.; MORAIS, F. Topic Modeling of Committee Discussions in the Brazilian Chamber of Deputies. In: SBC. *Anais do IX Symposium on Knowledge Discovery, Mining and Learning*. [S.l.], 2021. p. 49–56. Citado na página 23.

SCHUBERT, E. et al. DBSCAN revisited, revisited: why and how you should (still) use DBSCAN. *ACM Transactions on Database Systems (TODS)*, ACM New York, NY, USA, v. 42, n. 3, p. 1–21, 2017. Citado na página 23.

SILVA, R. N. M. da; SPRITZER, A.; FREITAS, C. D. S. Visualization of roll call data for supporting analyses of political profiles. In: IEEE. *2018 31st SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*. [S.l.], 2018. p. 150–157. Citado 4 vezes nas páginas 13, 14, 34 e 39.

SMELSER, K.; MILLER, J.; KOBOUROV, S. "Normalized Stress"is Not Normalized: How to Interpret Stress Correctly. *arXiv preprint arXiv:2408.07724*, 2024. Citado na página 52.

STEFANOV, P. et al. Predicting the topical stance and political leaning of media using tweets. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. [S.l.: s.n.], 2020. p. 527–537. Citado na página 23.

STEWART, G.; AL-KHASSAWENEH, M. An implementation of the HDBSCAN* clustering algorithm. *Applied Sciences*, MDPI, v. 12, n. 5, p. 2405, 2022. Citado na página 23.

TORGERSON, W. S. Multidimensional scaling: I. Theory and method. *Psychometrika*, Springer, v. 17, n. 4, p. 401–419, 1952. Citado na página 20.

ULHÔA, R.; COSTA, R. *PSDB e PFL reagem ao acordo PT-PMDB*. [S.l.]: Folha de S. Paulo, 2002. <https://www1.folha.uol.com.br/folha/brasil/ult96u42226.shtml>. Acesso em: 1 nov. 2024. Citado na página 54.

WICKELMAIER, F. An introduction to MDS. *Sound Quality Research Unit, Aalborg University, Denmark*, Citeseer, v. 46, n. 5, p. 1–26, 2003. Citado na página 20.

Apêndices

APÊNDICE A – Recorte do JSON da votação nominal do PRC 248/2005 na Câmara dos Deputados


```
{
  "dados": [
      ...
      {
      "tipoVoto": "Obstrução",
      "dataRegistroVoto": "2005-06-29T20:41:24",
      "deputado_": {
        "id": 74549,
        "uri": "https://dadosabertos.camara.leg.br/api/v2/deputados/74549",
        "nome": "João Almeida",
        "siglaPartido": "PSDB",
        "uriPartido": "https://dadosabertos.camara.leg.br/api/v2/partidos
           /36835",
        "siglaUf": "BA",
        "idLegislatura": 52,
        "urlFoto": "https://www.camara.leg.br/internet/deputado/bandep/74549.
           jpg",
        "email": null
      }
    },
    {
      "tipoVoto": "Sim",
      "dataRegistroVoto": "2005-06-29T20:41:24",
      "deputado_": {
        "id": 74675,
        "uri": "https://dadosabertos.camara.leg.br/api/v2/deputados/74675",
        "nome": "Almerinda de Carvalho",
        "siglaPartido": "PMDB",
        "uriPartido": "https://dadosabertos.camara.leg.br/api/v2/partidos
           /36800",
        "siglaUf": "RJ",
        "idLegislatura": 52,
        "urlFoto": "https://www.camara.leg.br/internet/deputado/bandep/74675.
```

```
          jpg",
        "email": null
      }
    },
    {
      "tipoVoto": "Não",
      "dataRegistroVoto": "2005-06-29T20:41:23",
      "deputado_": {
        "id": 74489,
        "uri": "https://dadosabertos.camara.leg.br/api/v2/deputados/74489",
        "nome": "Hamilton Casara",
        "siglaPartido": "PL",
        "uriPartido": "https://dadosabertos.camara.leg.br/api/v2/partidos
          /36795",
        "siglaUf": "RO",
        "idLegislatura": 52,
        "urlFoto": "https://www.camara.leg.br/internet/deputado/bandep/74489.
          jpg",
        "email": null
      }
    },
    ...
  ]
}
```

APÊNDICE B – Código da Figura 17

```
from branalysis.plenario import Camara
from datetime import date
from matplotlib import pyplot as plt
from statistics import mean

ANO = 2023

def calcula_idade(parlamentar):
   return (date(ANO, 12, 31) - parlamentar.data_nascimento).days / 365

plenario = Camara(ANO)

# Organiza os partidos por idade média, criando tuplas (partido, idade média)
partido_idade_media = [
   (partido, mean(calcula_idade(p) for p in parlamentares))
   for partido, parlamentares in plenario.parlamentares_por_partido().items()
]

# Ordena por idade média crescente
partido_idade_media.sort(key=lambda x: x[1])

# Plota o gráfico de barras
barras = plt.bar(*zip(*partido_idade_media))

plt.bar_label(barras, fmt=lambda x: f'{x:.1f}', label_type='edge')
plt.ylabel('Idade␣média')
plt.xticks(rotation=-45, ha='left')
plt.show()
```

APÊNDICE C – Código da Figura 18

```
from branalysis import SexoColorizador
from branalysis.plenario import Camara
from matplotlib import pyplot as plt
import numpy as np

CORES = plt.get_cmap("tab20b")

def count_sexos(parlamentares):
   homens = sum(1 for p in parlamentares if p.sexo == 'M')

   return homens, len(parlamentares) - homens

plenario = Camara(2023)
colorizador = SexoColorizador()

# Cria uma lista de tuplas (partido, quantia de homens, quantia de mulheres)
partido_sexo = [
   (partido, *count_sexos(parlamentares))
   for partido, parlamentares in plenario.parlamentares_por_partido().items()
]

# Ordena por proporção de mulheres
partido_sexo.sort(key=lambda x: x[1] / (x[2] + 0.0001))

# Plota o gráfico de barras
partidos, n_homens, n_mulheres = zip(*partido_sexo)
offset_y_barras = np.zeros(len(partidos))

for sexo, quantia in (('HOMEM', n_homens), ('MULHER', n_mulheres)):
   barras = plt.bar(partidos, quantia, color=CORES(colorizador[sexo]), label=
       sexo, bottom=offset_y_barras)

   plt.bar_label(barras, fmt=lambda x: '%d' % x if x > 0 else '', label_type='
       center')

   offset_y_barras += quantia

plt.legend()
```

```
plt.ylabel('Quantidade␣de␣parlamentares')
plt.xticks(rotation=-45, ha='left')
plt.show()
```

APÊNDICE D – Código da Figura 19

```
from branalysis import Camara, MatrizPolitica, Colorizador, CORES
from matplotlib.colors import ListedColormap
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt

plenario = Camara(2023)
parlamentares = plenario.parlamentares()

# Cria a matriz de votos dos parlamentares
matriz = MatrizPolitica(parlamentares, plenario.votacoes()).de_parlamentares()

# Aplica a redução de dimensionalidade
modelo = PCA(n_components=2).fit_transform(matriz)

# Cria um colorizador, que mapeia cores para cada partido
colorizador = Colorizador(fallback='MAIS␣DE␣UM')
cor_por_parlamentar = [colorizador[p.partidos()] for p in parlamentares]

# Plota o mapa ideológico
grafico = plt.scatter(modelo[:, 0], modelo[:, 1], cmap=ListedColormap(CORES), c
    =cor_por_parlamentar)
texto_legenda, cores = colorizador.textos_e_cores()
cores_legenda = grafico.legend_elements(num=cores)[0]

plt.legend(cores_legenda, texto_legenda)
plt.show()
```

APÊNDICE E – Código do Branalysis

E.1 agrupador.py

```
from collections import defaultdict
from typing import Any
import numpy as np

def agrupar_posicoes(modelo: np.ndarray, caracteristicas: list[Any | tuple[Any
    ]]) -> dict[Any | tuple[Any], np.ndarray]:
   """
␣␣␣Agrupa␣as␣posições␣de␣um␣modelo␣baseando-se␣em␣suas␣características.␣Exemplo
    :

␣␣␣```python
␣␣␣modelo␣=␣PCA().fit_transform(matrix)
␣␣␣clusters␣=␣HDBSCAN().fit_predict(modelo)
␣␣␣caracteristicas␣=␣zip(clusters,␣(p.partidos()␣for␣p␣in␣parlamentares))

␣␣␣for␣subcaracteristicas,␣submodelo␣in␣agrupar_posicoes(modelo,␣
    caracteristicas).items():
␣␣␣␣␣␣cluster,␣partido␣=␣subcaracteristicas

␣␣␣␣␣␣ax.scatter(submodelo[:,␣0],␣submodelo[:,␣1],␣color=cor[partido],␣marker=
    marcador[cluster])
␣␣␣```
␣␣␣"""
   grupos = defaultdict(list)

   for i, caracteristica in enumerate(caracteristicas):
      grupos[caracteristica].append(modelo[i, :])

   return {k: np.array(v) for k, v in grupos.items()}
```

E.2 camara.py

```
from . import camara_proposicoes, camara_votos
from .model import *
from .utils import *
```

```
from peewee import *

def fetch(year):
   return get_json(f'{CAMARA_FILES_API}/votacoes/json/votacoes-{year}.json')

def is_nominal(votacao):
   return votacao['votosSim'] + votacao['votosNao'] + votacao['votosOutros'] >
       0

def filter_nominais(votacoes):
   filtered = []

   for votacao in votacoes['dados']:
      if votacao['siglaOrgao'] == 'PLEN' and is_nominal(votacao):
         filtered.append(votacao)

   return filtered

def get_proposicao_id(votacao):
   return votacao['id'].split('-')[0]

def get_proposicao(votacao):
   proposicao_id = get_proposicao_id(votacao)

   return Camara_Proposicao.select().where(Camara_Proposicao.id ==
       proposicao_id).get()

def convert(votacoes):
   for votacao in votacoes:
      proposicao = get_proposicao(votacao)

      yield Camara_Votacao(
         id=votacao['id'],
         data=votacao['data'],
         tipo=proposicao.tipo,
         numero=proposicao.numero,
         ano=proposicao.ano
      )

def has_records(year):
   return Camara_Votacao.select().where(Camara_Votacao.data.year == year).
       exists()
```

```
def cache(year):
   if has_records(year):
      print(f'Cmara␣|␣Votações␣{year}␣|␣Já␣cacheadas.␣Pulando.')
      return

   votacoes = fetch(year)
   votacoes = filter_nominais(votacoes)

   with DB.atomic():
      for votacao in votacoes:
         proposicao_id = get_proposicao_id(votacao)

         camara_proposicoes.cache(proposicao_id)

   camara_votos.cache(year)

   print(f'Cmara␣|␣Votações␣{year}␣|␣Cacheando.')

   with DB.atomic():
      for camara_votacao in convert(votacoes):
         camara_votacao.save(force_insert=True)
```

E.3 camara_parlamentares.py

```
from .model import *
from .utils import *
from peewee import *

def fetch():
   return get_json(f'{CAMARA_FILES_API}/deputados/json/deputados.json')

def convert(deputados):
   for deputado in deputados['dados']:
      yield Camara_Parlamentar(
         id=deputado['uri'].split('/')[-1],
         nome=deputado['nome'],
         sexo=deputado['siglaSexo'],
         data_nascimento=deputado.get('dataNascimento', NO_DATE)
      )

def has_records():
```

```
    return Camara_Parlamentar.select().exists()

def cache():
    if has_records():
        print(f'Cmara | Deputados | Já cacheados. Pulando.')
        return

    deputados = fetch()

    print(f'Cmara | Deputados | Cacheando.')

    with DB.atomic():
        for camara_deputado in convert(deputados):
            camara_deputado.save(force_insert=True)
```

E.4 camara_proposicoes.py

```
from .model import *
from .utils import *
from peewee import *

def fetch(id):
    return get_json(f'{CAMARA_API}/proposicoes/{id}')

def convert(proposicao):
    proposicao = proposicao['dados']

    return Camara_Proposicao(
        id=proposicao['id'],
        tipo=proposicao['siglaTipo'],
        numero=proposicao['numero'],
        ano=proposicao['ano']
    )

def has_record(id):
    return Camara_Proposicao.select().where(Camara_Proposicao.id == id).exists()

def cache(id):
    justified_id = id.ljust(7)

    if has_record(id):
        print(f'Cmara | Proposição {justified_id} | Já cacheada. Pulando.')
```

```
      return

   proposicao = fetch(id)
   camara_proposicao = convert(proposicao)

   print(f'Cmara␣|␣Proposição␣{justified_id}␣|␣Cacheando.')

   camara_proposicao.save(force_insert=True)
```

E.5 camara_votos.py

```
from . import camara_parlamentares
from .model import *
from .utils import *
from peewee import *

def fetch(year):
   return get_json(f'{CAMARA_FILES_API}/votacoesVotos/json/votacoesVotos-{year}.
       json')

def convert_partido(partido):
   return NO_PARTY if partido == 'S.PART.' else partido

def convert(votos):
   for voto in votos['dados']:
      if 'idVotacao' in voto and 'id' in voto['deputado_']:
         yield Camara_Voto(
            votacao=voto['idVotacao'],
            parlamentar=voto['deputado_']['id'],
            partido=convert_partido(voto['deputado_'].get('siglaPartido',
                NO_ENTRY)),
            uf=voto['deputado_'].get('siglaUf', NO_ENTRY),
            voto=voto.get('voto', NO_ENTRY)
         )

def cache(year):
   camara_parlamentares.cache()

   votos = fetch(year)

   print(f'Cmara␣|␣Votos␣{year}␣|␣Cacheando.')
```

```
   with DB.atomic():
      for camara_voto in convert(votos):
         camara_voto.save(force_insert=True)
```

E.6 colorizador.py

```
from .plenario import Plenario
from typing import Any, Callable

CORES = [(0       , 0        , 0        , 1),
        (0.12156863, 0.46666667, 0.70588235, 1),
        (0.68235294, 0.78039216, 0.90980392, 1),
        (1        , 0.49803922, 0.05490196, 1),
        (1        , 0.73333333, 0.47058824, 1),
        (0.17254902, 0.62745098, 0.17254902, 1),
        (0.59607843, 0.8745098 , 0.54117647, 1),
        (0.83921569, 0.15294118, 0.15686275, 1),
        (1        , 0.59607843, 0.58823529, 1),
        (0.58039216, 0.40392157, 0.74117647, 1),
        (0.77254902, 0.69019608, 0.83529412, 1),
        (0.54901961, 0.3372549 , 0.29411765, 1),
        (0.76862745, 0.61176471, 0.58039216, 1),
        (0.89019608, 0.46666667, 0.76078431, 1),
        (0.96862745, 0.71372549, 0.82352941, 1),
        (0.49803922, 0.49803922, 0.49803922, 1),
        (0.78039216, 0.78039216, 0.78039216, 1),
        (0.7372549 , 0.74117647, 0.13333333, 1),
        (0.85882353, 0.85882353, 0.55294118, 1),
        (0.09019608, 0.74509804, 0.81176471, 1),
        (0.61960784, 0.85490196, 0.89803922, 1),
        (0.36746226, 0.57657207, 0.99569878, 1),
        (0.34374094, 0.99982132, 0.74416974, 1),
        (0.93693770, 0.35445788, 0.41093034, 1),
        (0.33537410, 0.98556435, 0.34436210, 1),
        (0.41227477, 0.69779199, 0.69629407, 1),
        (0.40796869, 0.71298256, 0.36128163, 1),
        (0.34350654, 0.33919606, 0.92732104, 1),
        (0.74948717, 0.35481691, 0.98683299, 1),
        (0.96067351, 0.99845293, 0.33948472, 1),
        (0.34555829, 0.82555298, 0.98921235, 1),
        (0.72532787, 0.48748777, 0.35086953, 1),
        (0.79460877, 0.99795462, 0.80524173, 1),
```

```
        (0.65089013, 0.54692150, 0.99562189, 1),
        (0.69060265, 0.99990261, 0.33661118, 1),
        (0.33850281, 0.34870083, 0.64869003, 1),
        (0.99635084, 0.34399350, 0.97769131, 1),
        (0.34568734, 0.85420426, 0.52686518, 1),
        (0.97384107, 0.55338914, 0.99604514, 1),
        (0.68163970, 0.33507463, 0.52441538, 1)]

class CorDict(dict):
   def textos_e_cores(self, key=None, reverse=False) -> tuple[list[str], list[
       int]]:
      if key is not None:
         key = lambda x: key(x[0])

      return zip(*sorted(self.items(), key=key, reverse=reverse))

class Colorizador(CorDict):
   """
␣␣␣O␣Colorizador␣é␣um␣dicionário␣especializado␣que␣mapeia␣chaves␣a␣índices␣de
␣␣␣cores.␣Toda␣chave,␣ao␣ser␣acessada␣pela␣primeira␣vez,␣é␣mapeada␣a␣um␣índice
␣␣␣único.␣Quando␣uma␣chave␣é␣acessada␣novamente,␣o␣mesma␣índice␣é␣retornada.

␣␣␣O␣Colorizador␣também␣descompacta␣chaves␣que␣são␣tuplas␣automaticamente,␣caso
␣␣␣elas␣tenham␣apenas␣um␣elemento.␣Caso␣uma␣tupla␣tenha␣mais␣de␣um␣elemento,␣
    ela
␣␣␣contará␣como␣o␣‘fallback‘␣definido.

␣␣␣Além␣disso,␣é␣possível␣definir␣um␣‘agrupador‘,␣que␣é␣uma␣função␣que␣
    transforma
␣␣␣chaves␣em␣outras␣chaves.
␣␣␣"""

   def __init__(self, fallback='DESCONHECIDO', agrupador: Callable[[Any], str]=
       None, reservados: list[tuple[str]]=None):
      super().__init__()

      self._fallback = fallback
      self._agrupador = agrupador
      self._reservados = { chave: i + 1 for i, chave in enumerate(reservados) }
          if reservados is not None else {}

      self._reset()
```

```
def clear(self):
   super().clear()
   self._reset()

def set_agrupador(self, agrupador: Callable[[Any], str]):
   self._agrupador = agrupador

def __getitem__(self, key):
   if self._agrupador is not None:
      key = self._agrupador(key)

   if key is None:
      self[self._fallback] = 0

      return 0

   elif type(key) != tuple:
      return super().__getitem__(self._increase_color(key))

   elif len(key) != 1:
      self[self._fallback] = 0

      return 0

   return super().__getitem__(self._increase_color(key[0]))

def _reset(self):
   self._last_color = len(self._reservados)

def _increase_color(self, key):
   if key not in self:
      if key in self._reservados:
         self[key] = self._reservados[key]

      else:
         self._last_color += 1

         self[key] = self._last_color

   return key
```

```
class SexoColorizador(CorDict):
    """
␣␣␣O␣SexoColorizador␣é␣um␣dicionário␣especializado␣que␣mapeia␣chaves␣de␣sexo␣a
␣␣␣índices␣de␣cores.␣Ele␣mapeia␣as␣chaves␣"M",␣"MASCULINO"␣e␣"HOMEM"␣ao␣índice␣
    0,
␣␣␣as␣chaves␣"F",␣"FEMININO"␣e␣"MULHER"␣ao␣índice␣19␣e␣todas␣as␣outras␣chaves␣
    ao
␣␣␣índice␣10.
␣␣␣"""
    def __init__(self, homem_texto='HOMEM', mulher_texto='MULHER', outro_texto='
        OUTRO'):
        super().__init__()

        self._homem_texto = homem_texto
        self._mulher_texto = mulher_texto
        self._outro_texto = outro_texto

    def __getitem__(self, key):
        key = key.upper()

        if key == 'M' or key == 'MASCULINO' or key == 'HOMEM':
            self[self._homem_texto] = 0
            return super().__getitem__(self._homem_texto)

        elif key == 'F' or key == 'FEMININO' or key == 'MULHER':
            self[self._mulher_texto] = 19
            return super().__getitem__(self._mulher_texto)

        else:
            self[self._outro_texto] = 10
            return super().__getitem__(self._outro_texto)

class TempoColorizador(CorDict):
    """
␣␣␣O␣TempoColorizador␣é␣um␣dicionário␣especializado␣que␣mapeia␣chaves␣de␣datas␣
    a
␣␣␣índices␣de␣cores.␣Ele␣mapeia␣as␣chaves␣a␣índices␣de␣cores␣de␣acordo␣com␣o␣
    ano
␣␣␣mais␣próximo␣que␣seja␣múltiplo␣do␣intervalo␣definido.
␣␣␣"""
    def __init__(self, plenario: Plenario, intervalo=10):
        super().__init__()
```

```
      _, data_final = plenario.periodo()
      self._intervalo = intervalo
      self._ano_final = self._get_closest(data_final.year)

   def __getitem__(self, key):
      ano = self._get_closest(key.year)

      if ano not in self:
         self[ano] = (ano - self._ano_final) // self._intervalo

      return super().__getitem__(ano)

   def _get_closest(self, ano):
      return ano // self._intervalo * self._intervalo
```

E.7 matriz_politica.py

```
from .db.model import Parlamentar, Votacao, Voto
from .db.utils import get_parlamentar_id
from collections import defaultdict
from statistics import mean
from typing import Callable, Iterable
import numpy as np, functools

def transformador_sim_nao(matriz_politica: 'MatrizPolitica', voto: Voto) ->
    float:
   """
␣␣␣Transforma␣um␣voto␣textual␣em␣um␣número,␣sendo␣1␣para␣"SIM",␣-1␣para␣"NO"␣e
␣␣␣0␣para␣" ABSTENO ",␣" OBSTRUO "␣e␣"P-NRV".␣Outros␣valores␣são␣considerados
␣␣␣faltantes␣e␣serão␣imputados␣pelo␣imputador.
␣␣␣"""
   match voto.voto:
      case 'SIM': return 1
      case 'NO': return -1
      case ' ABSTENO ', ' OBSTRUO ', 'P-NRV': return 0

def imputador_zero(matriz_politica: 'MatrizPolitica', votacao: Votacao, votos:
    list[Voto], votos_numericos: list[float]):
   """
␣␣␣Imputa␣votos␣faltantes␣com␣0.
␣␣␣"""
```

```
    return [x if x is not None else 0 for x in votos_numericos]

def imputador_vota_com_partido(matriz_politica: 'MatrizPolitica', votacao:
     Votacao, votos: list[Voto], votos_numericos: list[float]):
    """
␣␣␣␣Imputa␣votos␣faltantes␣com␣a␣média␣dos␣votos␣números␣do␣partido␣do␣
     parlamentar.
␣␣␣"""
    data = votacao.data
    direcao_partidaria = defaultdict(int)

    for i, voto in enumerate(votos):
        if votos_numericos[i] is not None:
            direcao_partidaria.setdefault(voto.partido, []).append(votos_numericos[
                i])

    for k in direcao_partidaria:
        direcao_partidaria[k] = mean(direcao_partidaria[k])

    for i, parlamentar in enumerate(matriz_politica.parlamentares()):
        if votos_numericos[i] is None:
            votos_numericos[i] = direcao_partidaria[parlamentar.partido(data)]

    return votos_numericos

TransformadorVoto = Callable[['MatrizPolitica', Voto], float]
Imputador = Callable[['MatrizPolitica', Votacao, list[Voto], list[float]], list
    [float]]

class MatrizPolitica:
    def __init__(
        self,
        parlamentares: list[Parlamentar],
        votacoes: Iterable[Votacao],
        imputador: Imputador = imputador_vota_com_partido,
        transformador_voto: TransformadorVoto = transformador_sim_nao
    ):
        self._votacoes = votacoes
        self._parlamentares = parlamentares
        self._transformador_voto = transformador_voto
        self._imputador = imputador
```

```
   def de_parlamentares(self):
      """
␣␣␣␣␣␣Retorna␣a␣matriz␣de␣votos␣dos␣parlamentares.␣Com␣um␣algoritmo␣de␣redução
␣␣␣␣␣␣de␣dimensionalidade,␣gera␣um␣mapa␣ideológico␣dos␣parlamentares,␣onde
␣␣␣␣␣␣parlamentares␣próximos␣votam␣de␣maneira␣similar.
␣␣␣␣␣␣"""
      return self.de_votacoes().T

   def de_votacoes(self):
      """
␣␣␣␣␣␣Retorna␣a␣matriz␣de␣votos␣das␣votações.␣Com␣um␣algoritmo␣de␣redução␣de
␣␣␣␣␣␣dimensionalidade,␣gera␣um␣mapa␣ideológico␣das␣votações,␣onde␣votações
␣␣␣␣␣␣similares␣possuem␣perfis␣de␣votos␣similares.
␣␣␣␣␣␣"""
      parlamentares_index = self._parlamentares_to_index()
      parlamentares_len = len(self._parlamentares)
      matrix = []

      for votacao in self._votacoes:
         votos = [None] * parlamentares_len
         matrix.append([None] * parlamentares_len)

         for voto in votacao.votos:
            parlamentar_id = get_parlamentar_id(voto)

            if parlamentar_id in parlamentares_index:
               votos[parlamentares_index[parlamentar_id]] = voto
               matrix[-1][parlamentares_index[parlamentar_id]] = self.
                   _transformador_voto(self, voto)

         matrix[-1] = self._imputador(self, votacao, votos, matrix[-1])

      return np.array(matrix)

   def de_similaridade(self, epsilon=0.01):
      """
␣␣␣␣␣␣Retorna␣a␣matriz␣de␣similaridade␣entre␣os␣parlamentares.␣A
␣␣␣␣␣␣similaridade␣é␣definida␣como␣a␣porcentagem␣de␣votos␣convergentes␣entre
␣␣␣␣␣␣os␣parlamentares,␣sendo␣um␣voto␣considerado␣convergente␣caso␣a␣diferença␣
    do
␣␣␣␣␣␣valor␣numérico␣de␣seus␣votos␣seja␣menor␣que␣‘epsilon‘.
␣␣␣␣␣␣"""
```

```
      return 1 - self.de_dissimilaridade(epsilon)

   def de_dissimilaridade(self, epsilon=0.01):
      """
␣␣␣␣␣␣Retorna␣a␣matriz␣de␣dissimilaridade␣entre␣os␣parlamentares.␣A
␣␣␣␣␣␣dissimilaridade␣é␣definida␣como␣a␣porcentagem␣de␣votos␣divergentes␣entre
␣␣␣␣␣␣os␣parlamentares,␣sendo␣um␣voto␣considerado␣divergente␣caso␣a␣diferença␣
    do
␣␣␣␣␣␣valor␣numérico␣de␣seus␣votos␣ultrapasse␣‘epsilon‘.
␣␣␣␣␣␣"""
      votos = self.de_parlamentares()
      matrix = np.zeros((votos.shape[0], votos.shape[0]))

      for pa in range(votos.shape[0]):
         for pb in range(pa + 1, votos.shape[0]):
            dissimilaridade = 0

            for i in range(votos.shape[1]):
               if abs(votos[pa, i] - votos[pb, i]) > epsilon:
                  dissimilaridade += 1

            dissimilaridade /= votos.shape[1]

            matrix[pa, pb] = dissimilaridade
            matrix[pb, pa] = dissimilaridade

      return matrix

   def votacoes(self):
      return self._votacoes

   def parlamentares(self):
      return self._parlamentares

   def imputador(self):
      return self._imputador

   def transformador_voto(self):
      return self._transformador_voto

   @functools.cache
   def _parlamentares_to_index(self):
```

```
        return { x.id: i for i, x in enumerate(self._parlamentares) }
```

E.8 model.py

```
from datetime import date
from peewee import *
from typing import TYPE_CHECKING
import functools

if TYPE_CHECKING:
   from plenario import Plenario

DB = SqliteDatabase('branalysis.db')
SENADO_API = 'https://legis.senado.leg.br/dadosabertos'
CAMARA_API = 'https://dadosabertos.camara.leg.br/api/v2'
CAMARA_FILES_API = 'https://dadosabertos.camara.leg.br/arquivos'

def search_by_date(info, data):
   for x, inicio, fim in info:
      if inicio <= data <= fim:
         return x

class BaseModel(Model):
   class Meta:
      database = DB

class Parlamentar(BaseModel):
   id = TextField(primary_key=True)
   nome = TextField()
   sexo = TextField()
   data_nascimento = DateField(null=True)

   def set_plenario(self, plenario: 'Plenario'):
      self._plenario = plenario

      return self

   @functools.cache
   def partido(self, data: date) -> str:
      """
␣␣␣␣␣␣Retorna␣o␣partido␣do␣parlamentar␣na␣data␣especificada.
```

```
␣␣␣␣␣␣"""
      return search_by_date(self.partidos(com_data=True), data)

   def partidos(self, com_data=False) -> tuple[str] | tuple[tuple[str, date,
       date]]:
      """
␣␣␣␣␣␣Retorna␣os␣partidos␣do␣parlamentar␣no␣período␣do␣Plenario␣associado.␣Como
␣␣␣␣␣␣parlamentares␣podem␣pertencer␣a␣mais␣de␣um␣partido␣durante␣o␣período
␣␣␣␣␣␣escolhido,␣seus␣partidos␣são␣representados␣como␣uma␣tupla,␣organizados␣em
␣␣␣␣␣␣ordem␣cronológica.␣Isto␣é,␣um␣parlamentar␣com␣a␣tupla
␣␣␣␣␣␣‘("PT",␣"PSDB",␣"PT")‘␣significa␣que␣ele␣se␣candidatou␣pelo␣PT,␣depois
␣␣␣␣␣␣pelo␣PSDB␣e␣depois␣voltou␣para␣o␣PT.

␣␣␣␣␣␣Também␣é␣possível␣passar␣o␣argumento␣‘com_data=True‘␣para␣obter␣a␣data␣de
␣␣␣␣␣␣início␣e␣fim␣associadas:

␣␣␣␣␣␣‘(("PT",␣date(2022,␣5,␣2),␣date(2022,␣6,␣3)),␣("PSDB",␣date(2022,␣6,␣4),␣
    date(2022,␣7,␣5)),␣("PT",␣date(2022,␣7,␣6),␣date(2022,␣8,␣7))‘

␣␣␣␣␣␣"""
      return self._plenario.partidos_por_parlamentar(com_data)[self.id]

   @functools.cache
   def uf(self, data: date) -> str:
      """
␣␣␣␣␣␣Retorna␣a␣UF␣do␣parlamentar␣na␣data␣especificada.
␣␣␣␣␣␣"""
      return search_by_date(self.ufs(com_data=True), data)

   def ufs(self, com_data=False) -> tuple[str] | tuple[tuple[str, date, date]]:
      """
␣␣␣␣␣␣Retorna␣as␣UFs␣do␣parlamentar␣no␣período␣do␣Plenario␣associado.␣Como
␣␣␣␣␣␣parlamentares␣podem␣pertencer␣a␣mais␣de␣uma␣UF␣durante␣o␣período␣
    escolhido,
␣␣␣␣␣␣suas␣UFs␣são␣representadas␣como␣uma␣tupla,␣organizados␣em␣ordem
␣␣␣␣␣␣cronológica.␣Isto␣é,␣um␣parlamentar␣com␣a␣tupla␣‘("SC",␣"RS",␣"SC")‘
␣␣␣␣␣␣significa␣que␣ele␣se␣candidatou␣em␣Santa␣Catarina,␣depois␣no␣Rio␣Grande␣
    do
␣␣␣␣␣␣Sul␣e␣depois␣voltou␣para␣Santa␣Catarina.

␣␣␣␣␣␣Também␣é␣possível␣passar␣o␣argumento␣‘com_data=True‘␣para␣obter␣a␣data␣de
␣␣␣␣␣␣início␣e␣fim␣associadas:
```

```
␣␣␣␣␣␣‘(("SC",␣date(2022,␣5,␣2),␣date(2022,␣6,␣3)),␣("RS",␣date(2022,␣6,␣4),␣
    date(2022,␣7,␣5)),␣("SC",␣date(2022,␣7,␣6),␣date(2022,␣8,␣7)))‘
␣␣␣␣␣␣"""
      return self._plenario.ufs_por_parlamentar(com_data)[self.id]

   @functools.cache
   def macroregiao(self, data: date) -> str:
      """
␣␣␣␣␣␣Retorna␣a␣macroregião␣do␣parlamentar␣na␣data␣especificada.
␣␣␣␣␣␣"""
      return search_by_date(self.macroregioes(com_data=True), data)

   def macroregioes(self, com_data=False) -> tuple[str] | tuple[tuple[str, date
       , date]]:
      """
␣␣␣␣␣␣Retorna␣as␣macroregiões␣do␣parlamentar␣no␣período␣do␣Plenario␣associado.
␣␣␣␣␣␣Como␣parlamentares␣podem␣pertencer␣a␣mais␣de␣uma␣macroregião␣durante␣o
␣␣␣␣␣␣período␣escolhido,␣suas␣macroregiões␣são␣representadas␣como␣uma␣tupla,
␣␣␣␣␣␣organizados␣em␣ordem␣cronológica.␣Isto␣é,␣um␣parlamentar␣com␣a
␣␣␣␣␣␣tupla␣‘("SUL",␣"SUDESTE",␣"SUL")‘␣significa␣que␣ele␣se␣candidatou␣no␣Sul,
␣␣␣␣␣␣depois␣no␣Sudeste␣e␣depois␣voltou␣para␣o␣Sul.

␣␣␣␣␣␣Também␣é␣possível␣passar␣o␣argumento␣‘com_data=True‘␣para␣obter␣a␣data␣de
␣␣␣␣␣␣início␣e␣fim␣associadas:

␣␣␣␣␣␣‘(("SUL",␣date(2022,␣5,␣2),␣date(2022,␣6,␣3)),␣("SUDESTE",␣date(2022,␣6,␣
    4),␣date(2022,␣7,␣5)),␣("SUL",␣date(2022,␣7,␣6),␣date(2022,␣8,␣7))‘
␣␣␣␣␣␣"""
      return self._plenario.macroregioes_por_parlamentar(com_data)[self.id]

   def presenca(self) -> float:
      """
␣␣␣␣␣␣Retorna␣a␣presença␣do␣parlamentar␣no␣período␣do␣Plenario␣associado.␣Não␣é
    ␣100%␣fiel.
␣␣␣␣␣␣"""
      return self._plenario.presenca_por_parlamentar()[self.id]

class Voto(BaseModel):
   votacao: ForeignKeyField
   parlamentar: ForeignKeyField
```

```
    partido = TextField()
    uf = TextField()
    voto = TextField()

class Votacao(BaseModel):
    id = TextField(primary_key=True)
    data = DateField()
    tipo = TextField()
    numero = IntegerField()
    ano = IntegerField()

class Camara_Votacao(Votacao):
    pass

class Camara_Parlamentar(Parlamentar):
    pass

class Camara_Voto(Voto):
    votacao = ForeignKeyField(Camara_Votacao, backref='votos')
    parlamentar = ForeignKeyField(Camara_Parlamentar, backref='votos')

# Used only a facilitator cache, fields are duplicated to Camara_Votacao
class Camara_Proposicao(BaseModel):
    id = TextField(primary_key=True)
    tipo = TextField()
    numero = IntegerField()
    ano = IntegerField()

class Senado_Votacao(Votacao):
    pass

class Senado_Parlamentar(Parlamentar):
    pass

class Senado_Voto(Voto):
    votacao = ForeignKeyField(Senado_Votacao, backref='votos')
    parlamentar = ForeignKeyField(Senado_Parlamentar, backref='votos')

DB.connect()
DB.create_tables([
    Camara_Votacao, Camara_Parlamentar, Camara_Voto, Camara_Proposicao,
    Senado_Votacao, Senado_Parlamentar, Senado_Voto
```

```
])
```

E.9 plenario.py

```
from .db import camara, senado
from .db.model import *
from .db.utils import get_parlamentar_id
from collections import defaultdict
from datetime import date, timedelta
from types import MappingProxyType
from typing import Iterable
import functools

ONE_DAY = timedelta(days=1)
MACROREGIOES = ('NORTE', 'NORDESTE', 'CENTRO-OESTE', 'SUDESTE', 'SUL')
MACROREGIOES_POR_UF = ({ uf : MACROREGIOES[0] for uf in ('AC', 'AM', 'AP', 'PA'
    , 'RO', 'RR', 'TO') }
                | { uf : MACROREGIOES[1] for uf in ('AL', 'BA', 'CE', 'MA',
                    'PB', 'PE', 'PI', 'RN', 'SE') }
                | { uf : MACROREGIOES[2] for uf in ('DF', 'GO', 'MS', 'MT')
                    }
                | { uf : MACROREGIOES[3] for uf in ('ES', 'MG', 'RJ', 'SP')
                    }
                | { uf : MACROREGIOES[4] for uf in ('PR', 'RS', 'SC') })

def cast_periodo(periodo, end=False):
   if isinstance(periodo, date):
      return periodo.strftime('%Y-%m-%d')

   periodo = str(periodo)

   if len(periodo) == 4:
      periodo += '-01-01' if not end else '-12-31'

   return periodo

class Plenario:
   _PARLAMENTAR_CLS: Parlamentar
   _VOTACAO_CLS: Votacao
   _VOTO_CLS: Voto

   def __init__(self, periodo_comeco, periodo_fim = None):
```

```
        periodo_fim = periodo_fim or periodo_comeco

        periodo_comeco = cast_periodo(periodo_comeco, end=False)
        periodo_fim = cast_periodo(periodo_fim, end=True)

        self._periodo = (periodo_comeco, periodo_fim)

    def parlamentares(self) -> list[Parlamentar]:
        return list(p.set_plenario(self) for p in self._PARLAMENTAR_CLS
                                                .select()
                                                .join(self._VOTO_CLS)
                                                .join(self._VOTACAO_CLS)
                                                .where(self._VOTACAO_CLS.data.
                                                    between(*self._periodo))
                                                .group_by(self._PARLAMENTAR_CLS)
                                                .order_by(self._PARLAMENTAR_CLS.
                                                    nome))

    @functools.cache
    def tipos_voto(self) -> tuple[str]:
        return tuple(v.voto for v in self._votos(self._VOTO_CLS.voto)
                                    .order_by(self._VOTO_CLS.voto)
                                    .distinct())

    @functools.cache
    def tipos_votacao(self) -> tuple[str]:
        return tuple(v.tipo for v in self._votacoes(self._VOTACAO_CLS.tipo)
                                    .order_by(self._VOTACAO_CLS.tipo)
                                    .distinct())

    @functools.cache
    def partidos(self) -> tuple[str]:
        return tuple(v.partido for v in self._votos(self._VOTO_CLS.partido)
                                        .order_by(self._VOTO_CLS.partido)
                                        .distinct())

    @functools.cache
    def ufs(self) -> tuple[str]:
        return tuple(v.uf for v in self._votos(self._VOTO_CLS.uf)
                                    .order_by(self._VOTO_CLS.uf)
                                    .distinct())
```

```
   def macroregioes(self) -> tuple[str]:
      return MACROREGIOES

   def parlamentares_por_partido(self) -> dict[str, list[Parlamentar]]:
      """
␣␣␣␣␣␣Retorna␣os␣parlamentares␣agrupados␣por␣partido.␣Parlamentares␣podem
␣␣␣␣␣␣estar␣associados␣a␣mais␣de␣um␣partido,␣devido␣a␣mudanças␣partidárias
␣␣␣␣␣␣durante␣o␣período␣escolhido.
␣␣␣␣␣␣"""
      votos = (self._votos(self._VOTO_CLS.parlamentar, self._VOTO_CLS.partido)
                  .order_by(self._VOTO_CLS.partido)
                  .distinct()
                  .iterator())

      result = {}

      for voto in votos:
         result.setdefault(voto.partido, []).append(voto.parlamentar.
             set_plenario(self))

      return result

   def parlamentares_por_uf(self) -> dict[str, list[Parlamentar]]:
      """
␣␣␣␣␣␣Retorna␣os␣parlamentares␣agrupados␣por␣UF.␣Parlamentares␣podem
␣␣␣␣␣␣estar␣associados␣a␣mais␣de␣uma␣UF,␣devido␣a␣mudanças␣de␣domicílio
␣␣␣␣␣␣durante␣o␣período␣escolhido.
␣␣␣␣␣␣"""
      votos = (self._votos(self._VOTO_CLS.parlamentar, self._VOTO_CLS.uf)
                  .order_by(self._VOTO_CLS.uf)
                  .distinct()
                  .iterator())

      result = {}

      for voto in votos:
         result.setdefault(voto.uf, []).append(voto.parlamentar.set_plenario(
             self))

      return result

   @functools.cache
```

```
   def presenca_por_parlamentar(self) -> dict[str, float]:
      """
␣␣␣␣␣␣Retorna␣a␣presença␣de␣cada␣parlamentar␣no␣período␣escolhido.␣Não␣é␣100%␣
    fiel.
␣␣␣␣␣␣"""
      result = defaultdict(int)
      total_votacoes = self._votacoes().count()
      votos = self._votos(self._VOTO_CLS.parlamentar).iterator()

      for voto in votos:
         result[get_parlamentar_id(voto)] += 1

      return MappingProxyType({k: v / total_votacoes for k, v in result.items()
          })

   @functools.cache
   def ufs_por_parlamentar(self, com_data=False) -> dict[str, tuple[str]] |
       dict[str, tuple[tuple[str, date, date]]]:
      """
␣␣␣␣␣␣Retorna␣as␣UFs␣de␣cada␣parlamentar␣no␣período␣escolhido.␣Como
␣␣␣␣␣␣parlamentares␣podem␣pertencer␣a␣mais␣de␣uma␣UF␣durante␣o␣período␣
    escolhido,
␣␣␣␣␣␣cada␣parlamentar␣tem␣suas␣UFs␣representadas␣como␣uma␣tupla,␣organizados␣
    em
␣␣␣␣␣␣ordem␣cronológica.␣Isto␣é,␣um␣parlamentar␣com␣a␣tupla␣‘("SC",␣"RS",␣"SC")
    ‘
␣␣␣␣␣␣significa␣que␣ele␣se␣candidatou␣em␣Santa␣Catarina,␣depois␣no␣Rio␣Grande␣
    do
␣␣␣␣␣␣Sul␣e␣depois␣voltou␣para␣Santa␣Catarina.

␣␣␣␣␣␣Também␣é␣possível␣passar␣o␣argumento␣‘com_data=True‘␣para␣obter␣a␣data␣de
␣␣␣␣␣␣início␣e␣fim␣associadas:

␣␣␣␣␣␣‘(("SC",␣date(2022,␣5,␣2),␣date(2022,␣6,␣3)),␣("RS",␣date(2022,␣6,␣4),␣
    date(2022,␣7,␣5)),␣("SC",␣date(2022,␣7,␣6),␣date(2022,␣8,␣7)))‘
␣␣␣␣␣␣"""
      votos = (self._votos(self._VOTO_CLS.parlamentar, self._VOTACAO_CLS.data,
          self._VOTO_CLS.uf)
                .order_by(self._VOTACAO_CLS.data)
                .iterator())

      return self._aggregate_votos(com_data, votos, lambda voto: voto.uf)
```

```
   @functools.cache
   def macroregioes_por_parlamentar(self, com_data=False) -> dict[str, tuple[
       str]] | dict[str, tuple[tuple[str, date, date]]]:
      """
␣␣␣␣␣␣Retorna␣as␣macrorregiões␣de␣cada␣parlamentar␣no␣período␣escolhido.␣Como
␣␣␣␣␣␣parlamentares␣podem␣pertencer␣a␣mais␣de␣uma␣macrorregião␣durante␣o␣perí
    odo
␣␣␣␣␣␣escolhido,␣cada␣parlamentar␣tem␣suas␣macrorregiões␣representadas␣como␣uma
␣␣␣␣␣␣tupla,␣organizados␣em␣ordem␣cronológica.␣Isto␣é,␣um␣parlamentar␣com␣a
␣␣␣␣␣␣tupla␣‘("SUL",␣"SUDESTE",␣"SUL")‘␣significa␣que␣ele␣se␣candidatou␣no␣Sul,
␣␣␣␣␣␣depois␣no␣Sudeste␣e␣depois␣voltou␣para␣o␣Sul.

␣␣␣␣␣␣Também␣é␣possível␣passar␣o␣argumento␣‘com_data=True‘␣para␣obter␣a␣data␣de
␣␣␣␣␣␣início␣e␣fim␣associadas:

␣␣␣␣␣␣‘(("SUL",␣date(2022,␣5,␣2),␣date(2022,␣6,␣3)),␣("SUDESTE",␣date(2022,␣6,␣
    4),␣date(2022,␣7,␣5)),␣("SUL",␣date(2022,␣7,␣6),␣date(2022,␣8,␣7))‘
␣␣␣␣␣␣"""
      votos = (self._votos(self._VOTO_CLS.parlamentar, self._VOTACAO_CLS.data,
          self._VOTO_CLS.uf)
                 .order_by(self._VOTACAO_CLS.data)
                 .iterator())

      return self._aggregate_votos(com_data, votos, lambda voto:
          MACROREGIOES_POR_UF[voto.uf])

   @functools.cache
   def partidos_por_parlamentar(self, com_data=False) -> dict[str, tuple[str]]
       | dict[str, tuple[tuple[str, date, date]]]:
      """
␣␣␣␣␣␣Retorna␣os␣partidos␣de␣cada␣parlamentar␣no␣período␣escolhido.␣Como
␣␣␣␣␣␣parlamentares␣podem␣pertencer␣a␣mais␣de␣um␣partido␣durante␣o␣período
␣␣␣␣␣␣escolhido,␣cada␣parlamentar␣tem␣seus␣partidos␣representados␣como␣uma␣
    tupla,
␣␣␣␣␣␣organizados␣em␣ordem␣cronológica.␣Isto␣é,␣um␣parlamentar␣com␣a␣tupla
␣␣␣␣␣␣‘("PT",␣"PSDB",␣"PT")‘␣significa␣que␣ele␣se␣candidatou␣pelo␣PT,␣depois
␣␣␣␣␣␣pelo␣PSDB␣e␣depois␣voltou␣para␣o␣PT.

␣␣␣␣␣␣Também␣é␣possível␣passar␣o␣argumento␣‘com_data=True‘␣para␣obter␣a␣data␣de
␣␣␣␣␣␣início␣e␣fim␣associadas:
```

```
␣␣␣␣␣␣‘(("PT",␣date(2022,␣5,␣2),␣date(2022,␣6,␣3)),␣("PSDB",␣date(2022,␣6,␣4),␣
    date(2022,␣7,␣5)),␣("PT",␣date(2022,␣7,␣6),␣date(2022,␣8,␣7))‘
␣␣␣␣␣␣"""
      votos = (self._votos(self._VOTO_CLS.parlamentar, self._VOTACAO_CLS.data,
          self._VOTO_CLS.partido)
                  .order_by(self._VOTACAO_CLS.data)
                  .iterator())

      return self._aggregate_votos(com_data, votos, lambda voto: voto.partido)

   def votacoes(self) -> list[Votacao]:
      return list(self._votacoes())

   def votos(self) -> list[Voto]:
      return list(self._votos())

   def _votacoes(self, *fields) -> Iterable[Votacao]:
      return (self._VOTACAO_CLS
                  .select(*fields)
                  .where(self._VOTACAO_CLS.data.between(*self._periodo))
                  .order_by(self._VOTACAO_CLS.data))

   def _votos(self, *fields) -> Iterable[Voto]:
      return (self._VOTO_CLS
               .select(*fields)
               .join(self._VOTACAO_CLS)
               .where(self._VOTACAO_CLS.data.between(*self._periodo))
               .order_by(self._VOTO_CLS.id))

   def periodo(self) -> tuple[date]:
      return (date.fromisoformat(self._periodo[0]), date.fromisoformat(self.
          _periodo[1]))

   def _aggregate_votos(self, com_data, votos: Iterable[Voto], key):
      if com_data:
         return self._aggregate_votos_com_data(votos, key)

      else:
         return self._aggregate_votos_sem_data(votos, key)

   def _aggregate_votos_sem_data(self, votos: Iterable[Voto], key):
      result = {}
```

```
      for voto in votos:
         parlamentar_id = get_parlamentar_id(voto)
         x = key(voto)

         if parlamentar_id not in result:
            result[parlamentar_id] = [x]

         elif result[parlamentar_id][-1] != x:
            result[parlamentar_id].append(x)

      return MappingProxyType({k: tuple(v) for k, v in result.items()})

   def _aggregate_votos_com_data(self, votos: Iterable[Voto], key):
      data_inicial, data_final = self.periodo()
      result = {}

      for voto in votos:
         parlamentar_id = get_parlamentar_id(voto)
         x = key(voto)

         if parlamentar_id not in result:
            result[parlamentar_id] = [[x, data_inicial, None]]

         elif result[parlamentar_id][-1][0] != x:
            result[parlamentar_id][-1][2] = voto.votacao.data - ONE_DAY
            result[parlamentar_id].append([x, voto.votacao.data, None])

      for parlamentar_id in result:
         result[parlamentar_id][-1][2] = data_final
         result[parlamentar_id] = tuple((x, inicio, fim) for x, inicio, fim in
             result[parlamentar_id])

      return MappingProxyType(result)

class Camara(Plenario):
   _PARLAMENTAR_CLS = Camara_Parlamentar
   _VOTACAO_CLS = Camara_Votacao
   _VOTO_CLS = Camara_Voto

   def __init__(self, periodo_comeco, periodo_fim = None):
      super().__init__(periodo_comeco, periodo_fim)
```

```
      data_inicial, data_final = self.periodo()

      for ano in range(data_inicial.year, data_final.year + 1):
         camara.cache(ano)

class Senado(Plenario):
   _PARLAMENTAR_CLS = Senado_Parlamentar
   _VOTACAO_CLS = Senado_Votacao
   _VOTO_CLS = Senado_Voto

   def __init__(self, periodo_comeco, periodo_fim = None):
      super().__init__(periodo_comeco, periodo_fim)

      data_inicial, data_final = self.periodo()

      for ano in range(data_inicial.year, data_final.year + 1):
         senado.cache(ano)
```

E.10 senado_parlamentares.py

```
from .model import *
from .utils import *
from peewee import *

def fetch(id):
   return get_json(f'{SENADO_API}/senador/{id}.json')

def get_parlamentares_sem_data_nascimento():
   return Senado_Parlamentar.select(Senado_Parlamentar.id).where(
       Senado_Parlamentar.data_nascimento.is_null())

def cache():
   parlamentares = get_parlamentares_sem_data_nascimento()

   if len(parlamentares) == 0:
      return

   with DB.atomic():
      for parlamentar in parlamentares:
         data = fetch(parlamentar.id)
```

```
        print(f'Senado␣|␣Data␣de␣nascimento␣{parlamentar.id}␣|␣Cacheando.')

        parlamentar.data_nascimento = data['DetalheParlamentar']['Parlamentar'
            ]['DadosBasicosParlamentar']['DataNascimento']
        parlamentar.save()
```

E.11 senado.py

```
from . import senado_parlamentares
from .model import *
from .utils import *
from peewee import *

def fetch(year):
   return get_json(f'{SENADO_API}/plenario/votacao/nominal/{year}.json')

def is_nominal(votacao):
   return votacao['Secreta'] == 'N' and 'Votos' in votacao

def filter_nominais(votacoes):
   filtered = []

   for votacao in votacoes['ListaVotacoes']['Votacoes']['Votacao']:
      if votacao['SiglaCasa'] == 'SF' and is_nominal(votacao):
         filtered.append(votacao)

   return filtered

def get_votacao_id(votacao):
   codigo_sessao = votacao['CodigoSessao']
   sequencial_sessao = votacao['CodigoSessaoVotacao']

   return f'{codigo_sessao}-{sequencial_sessao}'

def convert_votacao(votacao):
   return Senado_Votacao(
      id=get_votacao_id(votacao),
      data=votacao['DataSessao'],
      tipo=votacao['SiglaMateria'],
      numero=votacao['NumeroMateria'],
      ano=votacao['AnoMateria']
   )
```

```
def convert_partido(partido):
   return NO_PARTY if partido == 'S/PARTIDO' else partido

def convert_valor_voto(voto):
   return ' OBSTRUO ' if voto == 'P-OD' else voto

def convert_voto(votacao, voto):
   if 'CodigoSessao' in votacao and 'CodigoSessaoVotacao' in votacao and '
       CodigoParlamentar' in voto:
      return Senado_Voto(
         votacao=get_votacao_id(votacao),
         parlamentar=voto['CodigoParlamentar'],
         partido=convert_partido(voto.get('SiglaPartido', NO_ENTRY)),
         uf=voto.get('SiglaUF', NO_ENTRY),
         voto=convert_valor_voto(voto.get('Voto', NO_ENTRY))
      )

def convert_parlamentar(voto):
   return Senado_Parlamentar(
      id=voto['CodigoParlamentar'],
      nome=voto['NomeParlamentar'],
      sexo=voto['SexoParlamentar']
   )

def has_parlamentar(id):
   return Senado_Parlamentar.select().where(Senado_Parlamentar.id == id).exists
       ()

def has_records(year):
   return Senado_Votacao.select().where(Senado_Votacao.data.year == year).
       exists()

def cache(year):
   senado_parlamentares.cache()

   if has_records(year):
      print(f'Senado | Votações, votos e senadores de {year} | Já cacheadas.
          Pulando.')
      return

   votacoes = fetch(year)
```

```
   votacoes = filter_nominais(votacoes)

   print(f'Senado | Votações, votos e senadores de {year} | Cacheando.')

   with DB.atomic():
      for votacao in votacoes:
         for voto in votacao['Votos']['VotoParlamentar']:
            senado_parlamentar = convert_parlamentar(voto)

            if not has_parlamentar(senado_parlamentar.id):
               senado_parlamentar.save(force_insert=True)

            senado_voto = convert_voto(votacao, voto)

            if senado_voto:
               senado_voto.save(force_insert=True)

         senado_votacao = convert_votacao(votacao)
         senado_votacao.save(force_insert=True)

   senado_parlamentares.cache()
```

E.12 utils.py

```
import requests

NO_PARTY = 'SEM PARTIDO'
NO_ENTRY = 'SEM REGISTRO'
NO_DATE = '0000-00-00'

def strip_dict_values(data):
   if isinstance(data, dict):
      keys_to_del = []

      for key, value in data.items():
         new_value = strip_dict_values(value)

         if new_value is not None:
            data[key] = new_value
         else:
            keys_to_del.append(key)
```

```
      for key in keys_to_del:
         del data[key]

   elif isinstance(data, list):
      for index in range(len(data)):
         data[index] = strip_dict_values(data[index])

   elif isinstance(data, str):
      x = data.strip().upper()

      return x if x else None

   return data

def get_json(url):
   while True:
      response = requests.get(url, headers={'Accept': 'application/json'})

      if response.status_code == 200:
         return strip_dict_values(response.json())

      else:
         print(f'Erro␣de␣código␣{response.status_code}␣ao␣tentar␣buscar␣"{url}",
             ␣tentando␣novamente...')

def get_parlamentar_id(voto):
   return voto.__data__['parlamentar']

def get_votacao_id(voto):
   return voto.__data__['votacao']
```

Desenvolvimento de solução para visualização e análise de votações políticas

Mateus Arns Kreuch[1], **José Eduardo de Lucca**[1]

[1] Departamento de Informática e Estatística – Universidade de Santa Catarina (UFSC)

mateus.kreuch@ufsc.br, jose.lucca@ufsc.br

Abstract. *This work aims to develop an extensible solution for the visualization and analysis of roll-call data in the Brazilian Chamber of Deputies and Federal Senate, where the votes of each legislator are individually recorded and publicly disclosed. Using advanced statistical techniques, such as dimensionality reduction and clustering algorithms, the goal is to create empirical models that positions legislators on the political spectrum in a quantitative manner, facilitating the understanding of relationships between parties, the identification of coalitions, and the anticipation of political trends. By fostering a better understanding of political mechanisms, this work is expected to promote and strengthen transparency and democracy.*

Resumo. *Este trabalho visa desenvolver uma solução extensível para a visualização e análise das votações nominais da Câmara dos Deputados e do Senado Federal no Brasil, onde os votos de cada parlamentar são registrados individualmente e divulgados publicamente. Utilizando técnicas estatísticas avançadas, como redução de dimensionalidade e algoritmos de clusterização, o objetivo é criar modelos empíricos que posicionam os legisladores no espectro político de maneira quantitativa, facilitando a compreensão das relações entre partidos, identificação de coalizões e antecipação de tendências políticas. Ao facilitar uma melhor compreensão das mecânicas políticas, espera-se que este trabalho promova transparência e fortalecimento democrático.*

1. Introdução

No Brasil, a Câmara dos Deputados e o Senado Federal empregam vários tipos de votações, que são utilizadas para deliberar sobre diferentes matérias. Em especial, se destaca a votação nominal, onde, diferentemente de outros tipos de votação, os votos de cada parlamentar são registrados individualmente, e o resultado é divulgado publicamente. A votação nominal é obrigatória em matérias de maior relevância, como propostas de emenda à Constituição, vetos presidenciais, e pedidos de urgência. Além disso, os portais da Câmara dos Deputados[1] e do Senado Federal[2] disponibilizam online os dados das votações nominais em diversos formatos.

Um sumário de determinado período histórico pode ser construido ao organizar os dados das votações nominais em uma matriz de votos [Brigadir et al. 2016], onde cada célula leva o valor do voto. As linhas representam legisladores e as colunas representam

[1] https://dadosabertos.camara.leg.br

[2] https://www12.senado.leg.br/dados-abertos/

propostas em diferentes níveis de andamento. No entanto, devido à alta dimensionalidade dessa estrutura, é necessário o uso de técnicas estatísticas para extrair informações úteis para análise política.

Assim, é possível utilizar esta matriz para criar modelos empíricos, ou mapas ideológicos, que representam o posicionamento dos legisladores em relação uns aos outros, efetivamente situando-os no espectro político [Poole and Rosenthal 1985, da Silva et al. 2018]. Esses modelos são fundamentais para compreender as relações entre partidos, identificar coalizões e antecipar tendências políticas e ideológicas, ou, alternativamente, permitem reconhecer partes disfuncionais da organização governamental, possibilitando uma análise mais profunda e precisa do comportamento político.

Entre as técnicas capazes de construir mapas ideológicos está a redução de dimensionalidade [Brigadir et al. 2016], técnica aplicada no domínio da inteligência artificial que permite capturar as estruturas subjacentes dos dados originais e representá-los em um espaço de dimensão inferior, facilitando sua compreensão. De maneira complementar, algoritmos de clusterização, que visam fazer agrupamentos automáticos de dados segundo o seu grau de semelhança, permitem identificar legisladores com visões políticas similares [Lee et al. 2017].

De fato, a utilização da redução de dimensionalidade na análise política não é novidade no resto do mundo, remontando a poole1985 com a publicação da família de algoritmos NOMINATE [Poole and Rosenthal 1985]. No entanto, a transparência governamental que possibilita esse tipo de pesquisa no Brasil é recente [Hagopian and Mainwaring 2005]. Consequentemente, muitos dos projetos existentes contêm lacunas, como a falta de dados sobre o Senado Federal [da Silva et al. 2018]. Além disso, a maioria dos projetos foca no público geral, e não necessariamente em pesquisadores. Isto é, possuem interfaces para visualizar informações e, até certo ponto, controlar o sistema desenvolvido, mas não possibilitam a extensão e aplicação de outros algoritmos em cima dos modelos gerados.

Assim, este trabalho desenvolverá uma biblioteca em Python para facilitar a visualização e análise das votações nominais da Câmara dos Deputados e do Senado Federal. Também serão exploradas diversas técnicas de clusterização e redução de dimensionalidade, avaliando sua aplicabilidade no cenário político brasileiro.

2. Branalysis

A biblioteca, doravante chamada de branalysis, realiza a coleta dos dados de votações nominais, disponibilizando-os através de uma API unificada, e produz as estruturas necessárias para a redução de dimensionalidade, sendo inteiramente compatível com a biblioteca scikit-learn[3], reconhecida como uma das principais bibliotecas de estatística e aprendizado de máquina no ecossistema Python. Além disso, a solução facilita a criação de gráficos a partir da biblioteca matplotlib[4], a principal biblioteca de visualização de dados no Python.

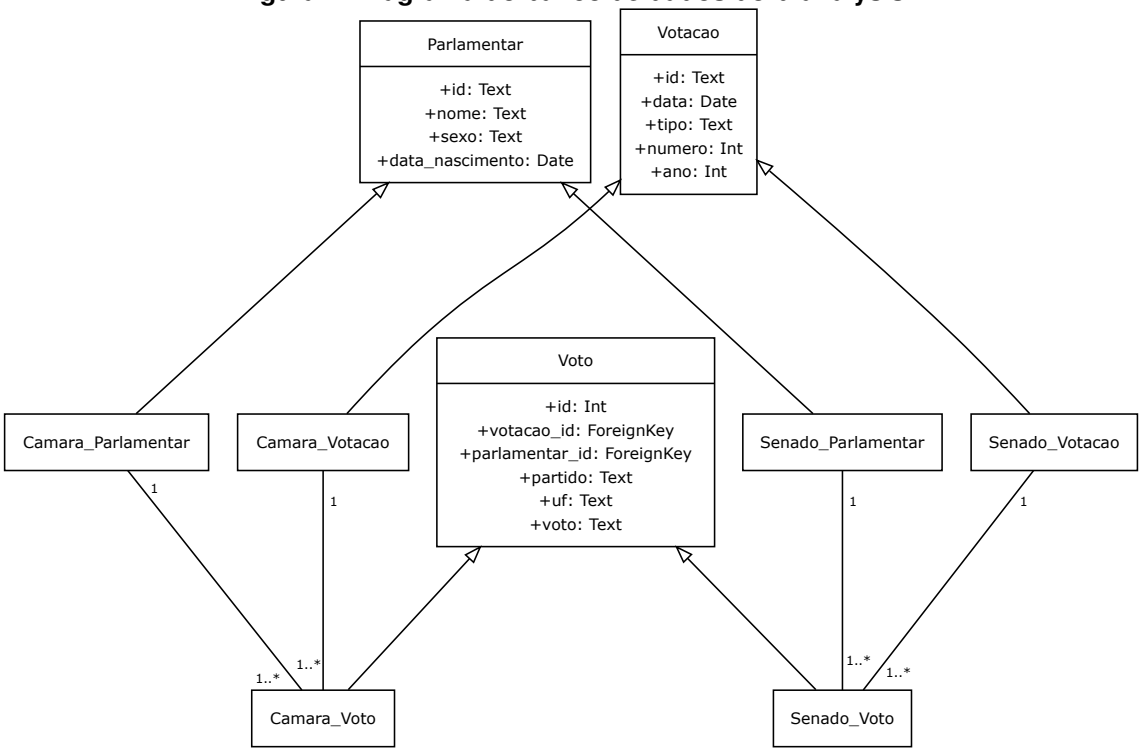
No entanto, essa compatibilidade existe sem incluí-las como dependências explícitas, operando apenas sobre estruturas comuns. Suas únicas dependências efeti-

[3]https://scikit-learn.org
[4]https://matplotlib.org

vas são as bibliotecas numpy[5], requests[6] e peewee[7]. Dessa forma, outras bibliotecas de aprendizado de máquina podem ser utilizadas, desde que compatíveis com as mesmas estruturas de dados, como é o caso da umap-learn[8], que fornece o algoritmo UMAP ausente no scikit-learn, que será estudado neste artigo.

A coleta de dados foi realizada por meio das APIs públicas disponibilizadas pela Câmara dos Deputados[9] e pelo Senado Federal[10]. O branalysis coleta os dados de votações nominais de ambas as casas legislativas, assim como informações sobre os legisladores e seus votos, os quais são armazenados em um banco de dados SQLite local, com modelagem simétrica entre as casas. O processo de coleta segue um modelo incremental: a cada execução, a biblioteca verifica se os dados referentes ao ano solicitado já foram previamente capturados. Caso positivo, a coleta é descartada para evitar duplicidade; caso contrário, os dados são extraídos e armazenados no banco de dados.

Figura 1. Diagrama do banco de dados do branalysis



A escolha de utilizar um banco de dados em vez de métodos mais primitivos de cacheamento, como arquivos CSV, foi motivada pela facilidade de uso e pela capacidade de reduzir o volume de dados, visando solucionar problemas inerentes às APIs disponíveis. Durante o desenvolvimento da biblioteca, foi constatado que diversos *endpoints* das plataformas apresentavam falhas. Requisições minimamente complexas, como a filtragem de votações por data, frequentemente resultavam em *timeout* ou não surgiam efeito.

[5]https://numpy.org
[6]https://requests.readthedocs.io
[7]http://docs.peewee-orm.com
[8]https://umap-learn.readthedocs.io
[9]https://dadosabertos.camara.leg.br
[10]https://www12.senado.leg.br/dados-abertos/

Mesmo quando bem-sucedidas, os dados eram organizados de maneira ineficiente, exigindo múltiplas requisições adicionais para completar as informações que o branalysis buscava coletar. Essa limitação evidenciou a necessidade de métodos mais sofisticados para gerenciar o que já havia sido armazenado em cache e o que ainda faltava, a fim de evitar requisições desnecessárias e assegurar a integridade dos dados. A transacionalidade proporcionada pelo SQLite revelou-se essencial nesse processo, especialmente em cenários de interrupção inesperada da coleta de dados.

De qualquer maneira, a partir desses dados, é possível gerar uma matriz de votos, que é a base para a aplicação dos algoritmos de redução de dimensionalidade. Essa matriz de votos $V(n \times m)$, onde n representa o número de parlamentares e m corresponde ao número de votações nominais, contém, em cada célula V_{ij}, o voto do parlamentar i na votação nominal j. O valor numérico associado a cada voto é determinado por um *transformador*.

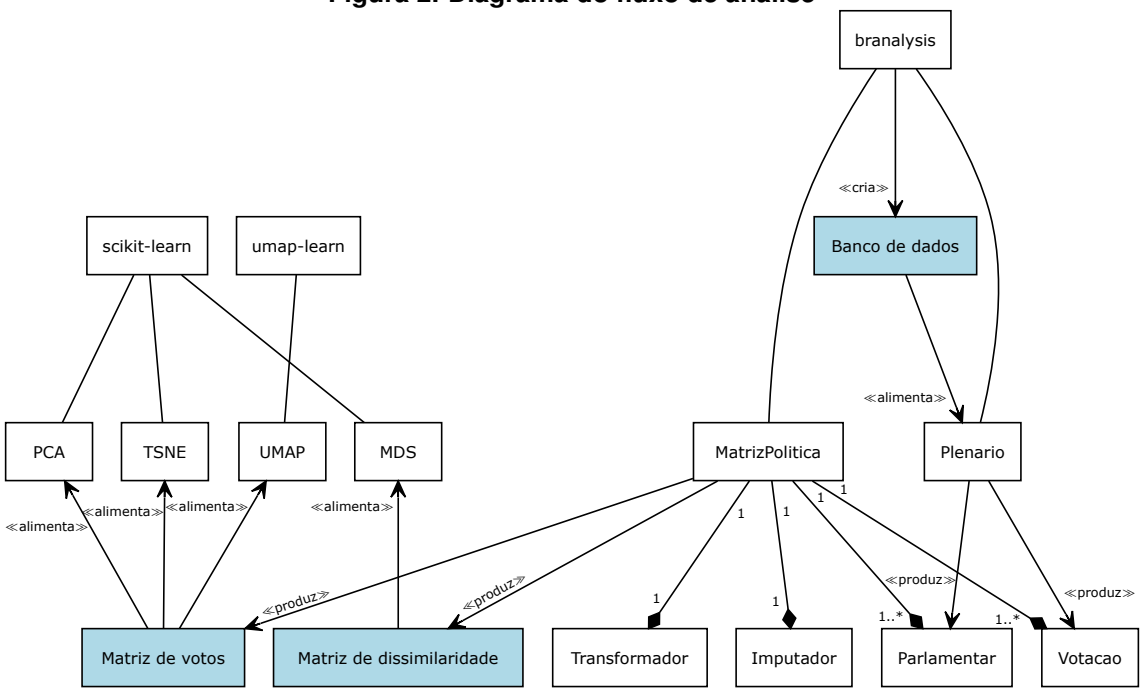
O transformador padrão converte os votos em 1 para "Sim", -1 para "Não" e 0 para "Abstenção" e "Obstrução". O usuário pode definir diferentes transformadores, conforme o aspecto dos votos que deseja explorar. Por exemplo, um transformador que atribui o valor 1 a todos os votos presentes poderia ser utilizado para identificar parlamentares que costumam participar das mesmas votações.

Há também um *imputador*, que define o que fazer com votos faltantes. O imputador padrão substitui os votos ausentes pela média do valor númerico dos votos do partido do parlamentar naquela votação específica. Assim como o transformador, o imputador também pode ser configurado pelo usuário, conforme suas necessidades analíticas.

Com esses componentes definidos, foi pensada a criação de dois tipos de matrizes de votos: uma matriz de parlamentares e uma matriz de votações, sendo uma a transposta da outra. Dessa forma, torna-se possível aplicar técnicas de redução de dimensionalidade para gerar tanto mapas ideológicos de parlamentares, como já discutido, quanto mapas ideológicos de votações, que identificam quais votações são mais semelhantes entre si, com base no perfil de voto dos parlamentares.

Além disso, foi implementada também uma matriz de dissimilaridade, exigida pelo algoritmo MDS. A matriz de dissimilaridade, denotada por $D(n \times n)$, onde n representa o número de parlamentares, contém, em cada célula D_{ij}, o valor da dissimilaridade entre os parlamentares i e j. A dissimilaridade é definida como a porcentagem de votos divergentes entre os parlamentares, sendo um voto considerado divergente caso a diferença do valor numérico de seus votos, após a aplicação do transformador e imputador, seja maior que ε, que pode ser definido pelo usuário. Essas estruturas são então utilizadas para gerar modelos de redução de dimensionalidade. De modo geral, o fluxo de análise pode ser visualizado na Figura 2.

Figura 2. Diagrama do fluxo de análise



3. Validação

Utilizando os algoritmos PCA, t-SNE, UMAP e MDS, e as técnicas de clusterização Propagação por Afinidade e HDBSCAN, foram gerados mapas ideológicos de diversos períodos legislativos, com o objetivo de verificar a corretude dos dados coletados e a precisão dos modelos gerados, assim como investigar a aplicabilidade dos algoritmos mais populares no contexto brasileiro. Os parâmetros utilizados em cada algoritmo foram:

- **PCA**: $whiten = False$.
- **t-SNE**: $perplexity = 30.0$, $early\_exaggeration = 12.0$, $learning\_rate = 50$, $max\_iter = 1000$.
- **MDS**: $metric = True$, $max\_iter = 300$, $eps = 0.001$.
- **UMAP**: $n\_neighbors = 15$, $min\_dist = 0.1$, $spread = 1.0$.
- **Propagação por Afinidade**: $damping = 0.7$, $max\_iter = 200$, $convergence\_iter = 15$.
- **HDBSCAN**: $min\_cluster\_size = 5$, $max\_cluster\_size = None$, $alpha = 1.0$, $leaf\_size = 40$.

Alguns dos períodos legislativos avaliados serão apresentados a seguir, com os respectivos mapas ideológicos. Foram utilizadas algumas métricas para medir a qualidade dos modelos gerados, entre elas:

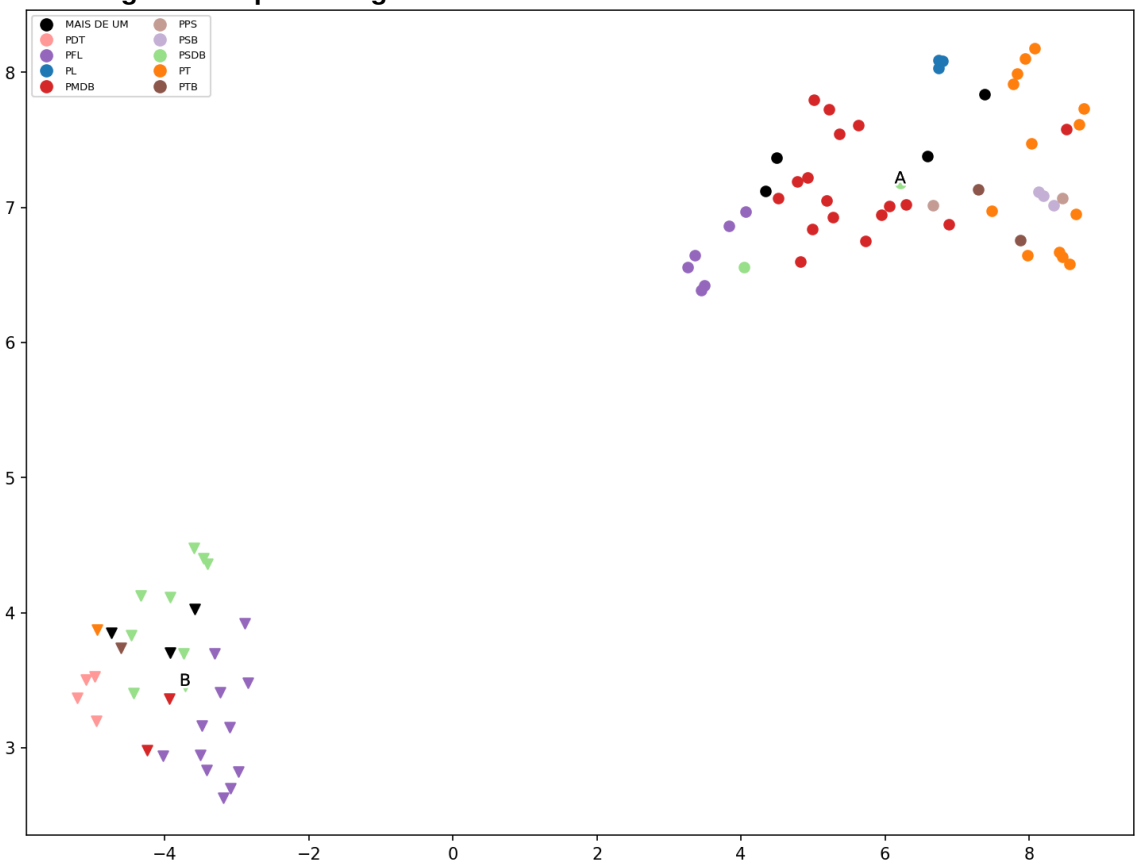
- **Variância total explicada**: A soma das variâncias explicadas por cada dimensão do modelo, nesse caso bidimensional. Um valor acima de 70% é usualmente considerado bom [Jolliffe and Cadima 2016].
- **Tensão normalizada pela escala (*Scale-normalized stress*)** [Smelser et al. 2024]: A tensão indica quão bem o modelo preserva as distâncias

originais entre os pontos. Para a tensão normalizada, um valor de 0% indica um modelo “perfeito”, 2,5% excelente, 5% bom, 10% regular e acima de 20%, ruim [Kruskal 1964].

- **Confiabilidade**: O quão bem foram preservadas as vizinhanças locais do espaço de alta dimensão. Um valor de 100% indica que todos os pares de pontos que eram vizinhos no espaço de alta dimensão continuam sendo vizinhos no espaço de baixa dimensão.

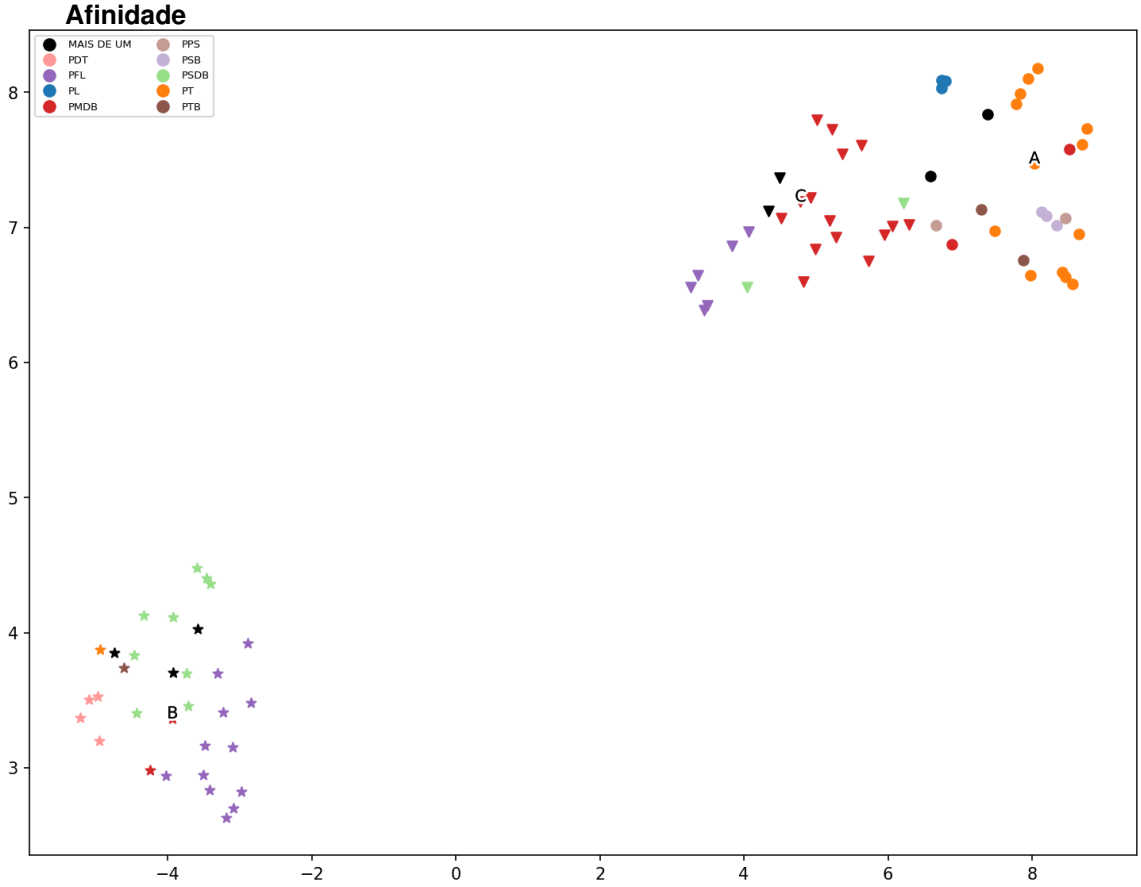
- **Homogeneidade partidária**: Quão homogêneo são os clusters em relação a afiliação partidária. Um valor de 100% indica que os clusters são compostos de membros de um único partido.

Figura 3. Mapa ideológico do Senado Federal de 2003 - UMAP e HDBSCAN



As diferentes formas representam diferentes clusters e pontos menores representam outliers. Letras representam os centros dos diferentes clusters, onde (A) Eduardo Siqueira Campos, (B) Arthur Virgílio. Quanto mais próximos dois parlamentares, mais parecidos foram seus votos em plenário no período. Fonte: autoria própria.

Figura 4. Mapa ideológico do Senado Federal de 2003 - UMAP e Propagação por Afinidade



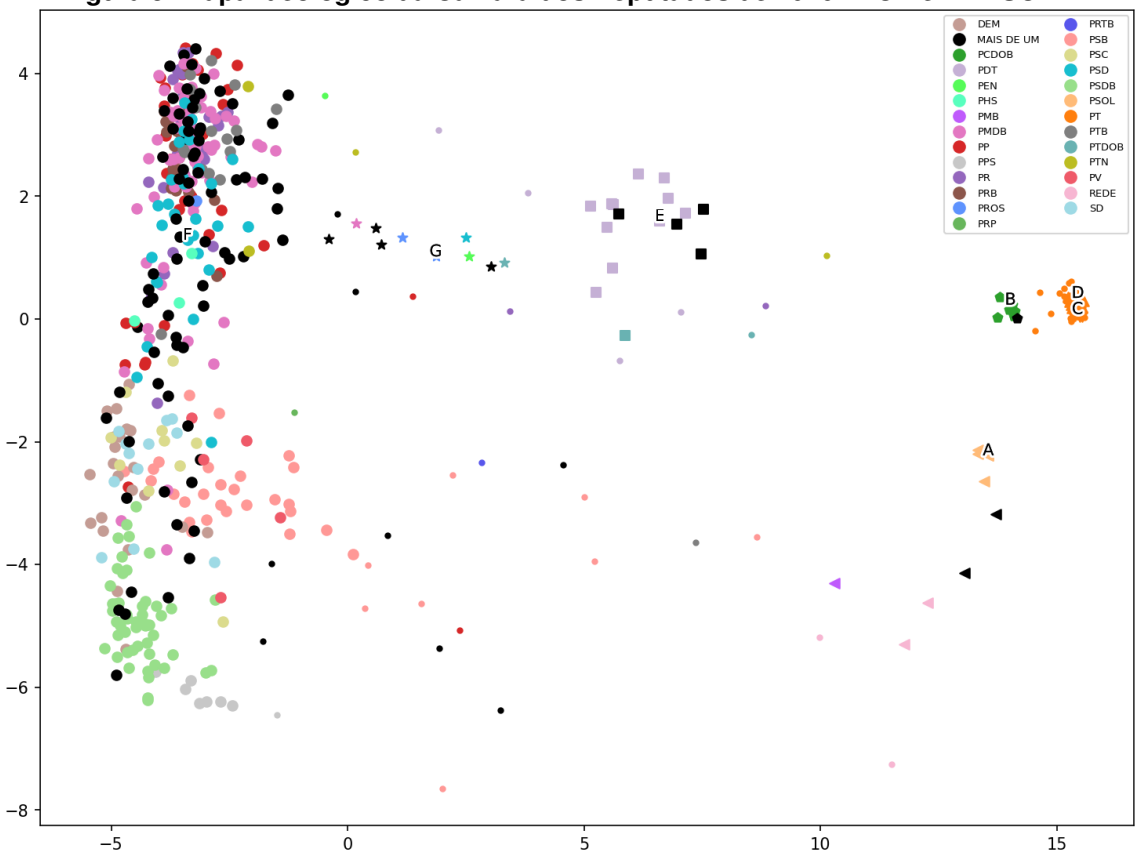
As diferentes formas representam diferentes clusters e pontos menores representam outliers. Letras representam os centros dos diferentes clusters, onde (A) Ana Júlia Carepa, (B) Mão Santa, (C) Ramez Tebet. Quanto mais próximos dois parlamentares, mais parecidos foram seus votos em plenário no período. Fonte: autoria própria.

Tabela 1. Métricas do modelo UMAP do Senado Federal de 2003

Métrica	Valor
Confiabilidade	94.4%
Outliers (HDBSCAN)	0%
Homogeneidade partidária (HDBSCAN)	13.2%
Homogeneidade partidária (Propagação por Afinidade)	29.1%

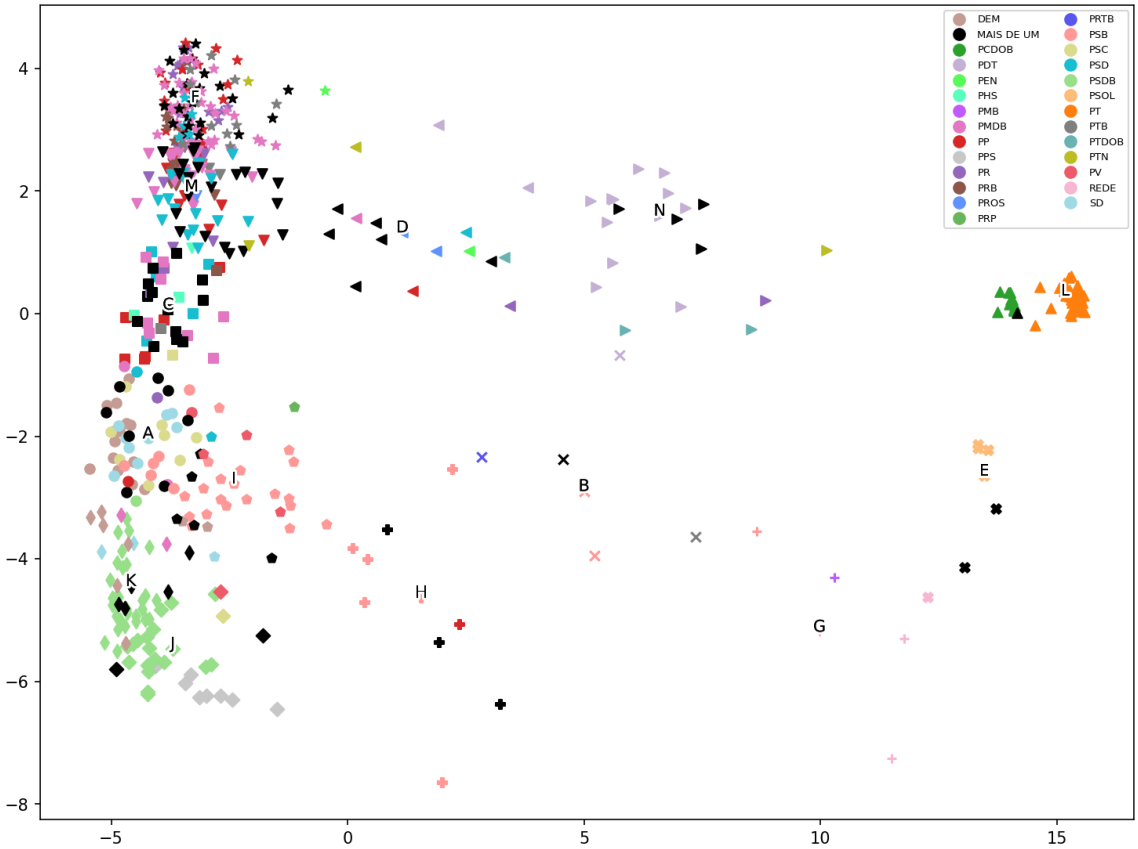
Em 2003, o acordo entre o PT e PMDB para dividir as presidências da Câmara e do Senado gerou uma reação imediata de exclusão por parte do PSDB, PFL e PDT, que, sem acesso ao poder no Congresso, articularam uma oposição ao governo eleito [Ulhôa and Costa 2002]. Simultaneamente, o PFL sofria uma divisão interna, com parte do partido apoiando o governo e parte a oposição [Ribeiro 2014]. É possível observar esses eventos no mapa ideológico gerado (Figura 3 e 4), indicando que o algoritmo UMAP foi capaz de capturar a dinâmica política do período.

Figura 5. Mapa ideológico da Câmara dos Deputados de 2016 - PCA e HDBSCAN



As diferentes formas representam diferentes clusters e pontos menores representam outliers. Letras representam os centros dos diferentes clusters, onde (A) Edmilson Rodrigues, (B) Wadson Ribeiro, (C) Adelmo Carneiro Leão, (D) Valmir Prascidelli, (E) Wolney Queiroz, (F) Heuler Cruvinel, (G) Bosco Costa. Quanto mais próximos dois parlamentares, mais parecidos foram seus votos em plenário no período. Fonte: autoria própria.

Figura 6. Mapa ideológico da Câmara dos Deputados de 2016 - PCA e Propagação por Afinidade



As diferentes formas representam diferentes clusters e pontos menores representam outliers. Letras representam os centros dos diferentes clusters, onde (A) Benjamin Maranhão, (B) César Messias, (C) Felipe Bornier, (D) George Hilton, (E) Jean Wyllys, (F) Jorge Côrte Real, (G) João Derly, (H) Júlio Delgado, (I) Keiko Ota, (J) Max Filho, (K) Nelson Padovani, (L) Nelson Pellegrino, (M) Roberto Teixeira, (N) Wolney Queiroz. Quanto mais próximos dois parlamentares, mais parecidos foram seus votos em plenário no período. Fonte: autoria própria.

Tabela 2. Métricas do modelo PCA da Câmara dos Deputados de 2016

Métrica	**Valor**
Variância explicada pelo eixo X	51.9%
Variância explicada pelo eixo Y	10%
Variância total explicada	61.9%
Outliers (HDBSCAN)	8.4%
Homogeneidade partidária (HDBSCAN)	23.9%
Homogeneidade partidária (Propagação por Afinidade)	45.1%

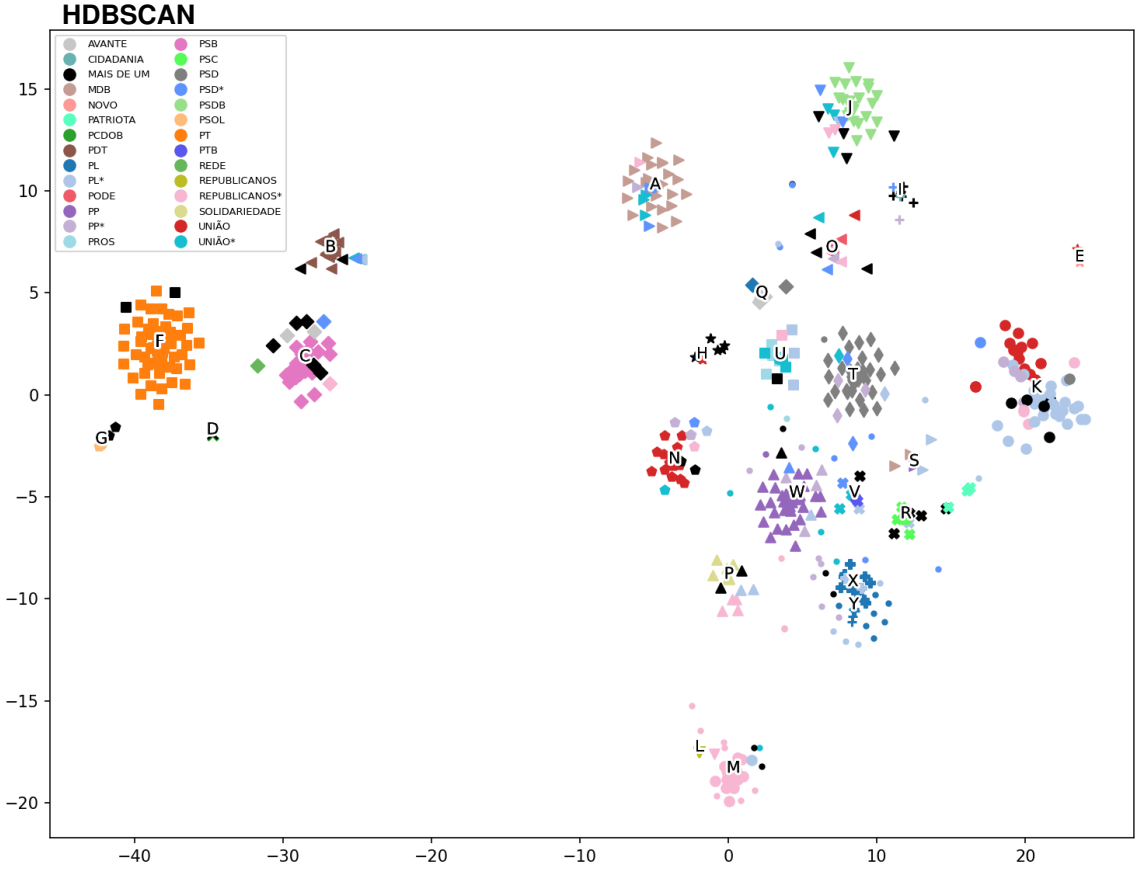
Em 2016, o processo de impeachment de Dilma Rousseff expôs o isolamento do PT e seus aliados no Congresso, especialmente diante da fragmentação partidária e da perda de apoio de antigos parceiros de coalizão, como o PMDB, PP e PTB, cujos

parlamentares votaram majoritariamente pelo impeachment de Dilma [Pitombo 2016]. Os partidos PDT, PSOL, PC do B e Rede, por sua vez, votaram majoritariamente contra o impeachment, indicando maior proximidade ao governo.

Houve também uma grande troca partidária neste ano, impulsionada pela Emenda Constitucional nº 91, que abriu uma janela partidária (momento em que um legislador pode mudar de partido sem sofrer sanções) excepcional. Durante esse período, 92 dos 513 deputados trocaram de legenda [Haje 2016].

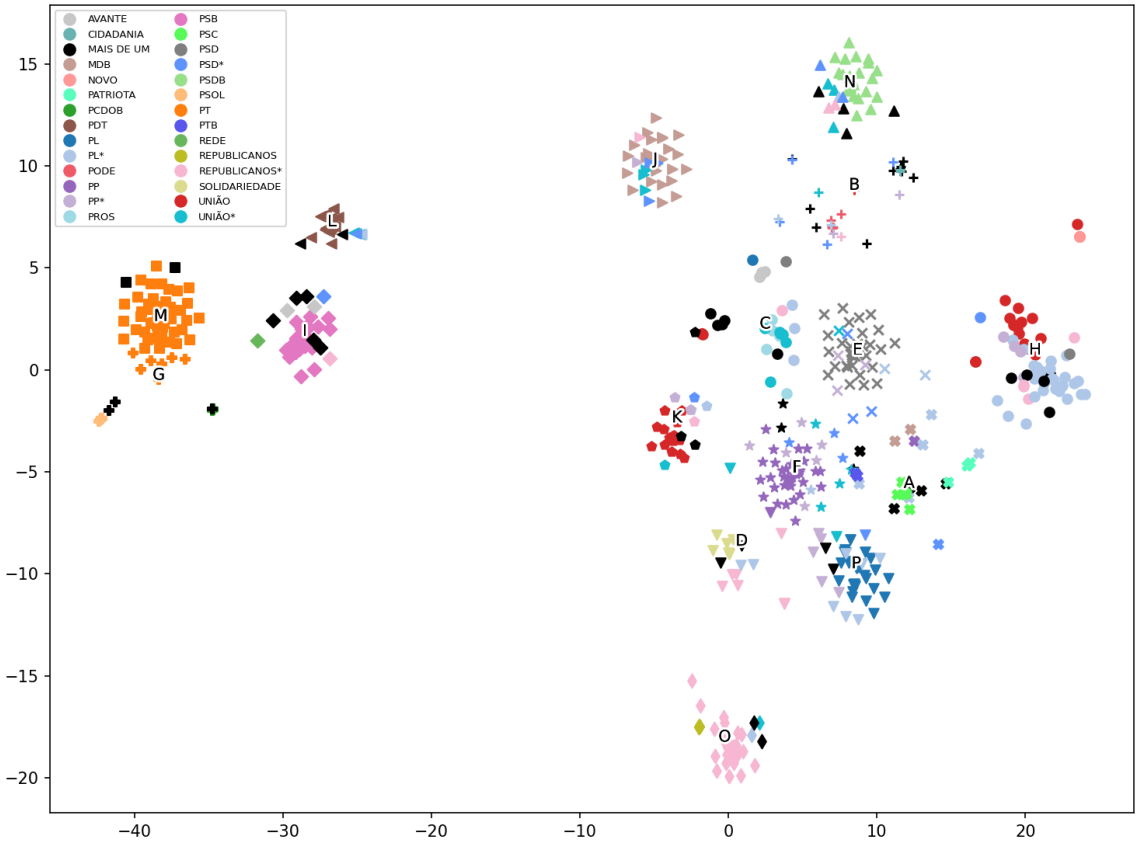
Com esse contexto, é possível observar a fragmentação partidária e a polarização política do período, assim como as mudanças na composição partidária, no mapa ideológico gerado pelo algoritmo PCA (Figura 5 e 6).

Figura 7. Mapa ideológico da Câmara dos Deputados de 2019 a 2022 - t-SNE e HDBSCAN



As diferentes formas representam diferentes clusters e pontos menores representam outliers. Legendas com asterisco (*) denotam parlamentares que entraram no partido após o início do período. Letras representam os centros dos diferentes clusters, onde (A) Leonardo Picciani, (B) Mauro Benevides Filho, (C) João H. Campos, (D) Márcio Jerry, (E) Alexis Fonteyne, (F) Rachel Marques, (G) Áurea Carolina, (H) Zé Augusto Nalin, (I) Joceval Rodrigues, (J) Rafafá, (K) Márcio Labre, (L) Henrique do Paraíso, (M) Maria Rosas, (N) Juninho do Pneu, (O) Sargento Alexandre, (P) Ottaci Nascimento, (Q) Fernando Borja, (R) Pedro Dalua, (S) Evair Vieira de Melo, (T) Fábio Faria, (U) Dr. Agripino Magalhães, (V) Santini, (W) Marco Brasil, (X) Junior Lourenço, (Y) Gorete Pereira. Quanto mais próximos dois parlamentares, mais parecidos foram seus votos em plenário no período. Fonte: autoria própria.

Figura 8. Mapa ideológico da Câmara dos Deputados de 2019 a 2022 - t-SNE e Propagação por Afinidade



As diferentes formas representam diferentes clusters e pontos menores representam outliers. Legendas com asterisco (*) denotam parlamentares que entraram no partido após o início do período. Letras representam os centros dos diferentes clusters, onde (A) Aluisio Mendes, (B) Bozzella, (C) Clarissa Garotinho, (D) Dra. Vanda Milani, (E) Edilázio Júnior, (F) Eliza Virgínia, (G) Erika Kokay, (H) General Peternelli, (I) João H. Campos, (J) Leonardo Picciani, (K) Marcos Soares, (L) Mário Heringer, (M) Nelson Pellegrino, (N) Roberto Pessoa, (O) Vinicius Carvalho, (P) Vinicius Gurgel. Quanto mais próximos dois parlamentares, mais parecidos foram seus votos em plenário no período. Fonte: autoria própria.

Tabela 3. Métricas do modelo t-SNE da Câmara dos Deputados de 2019 a 2022

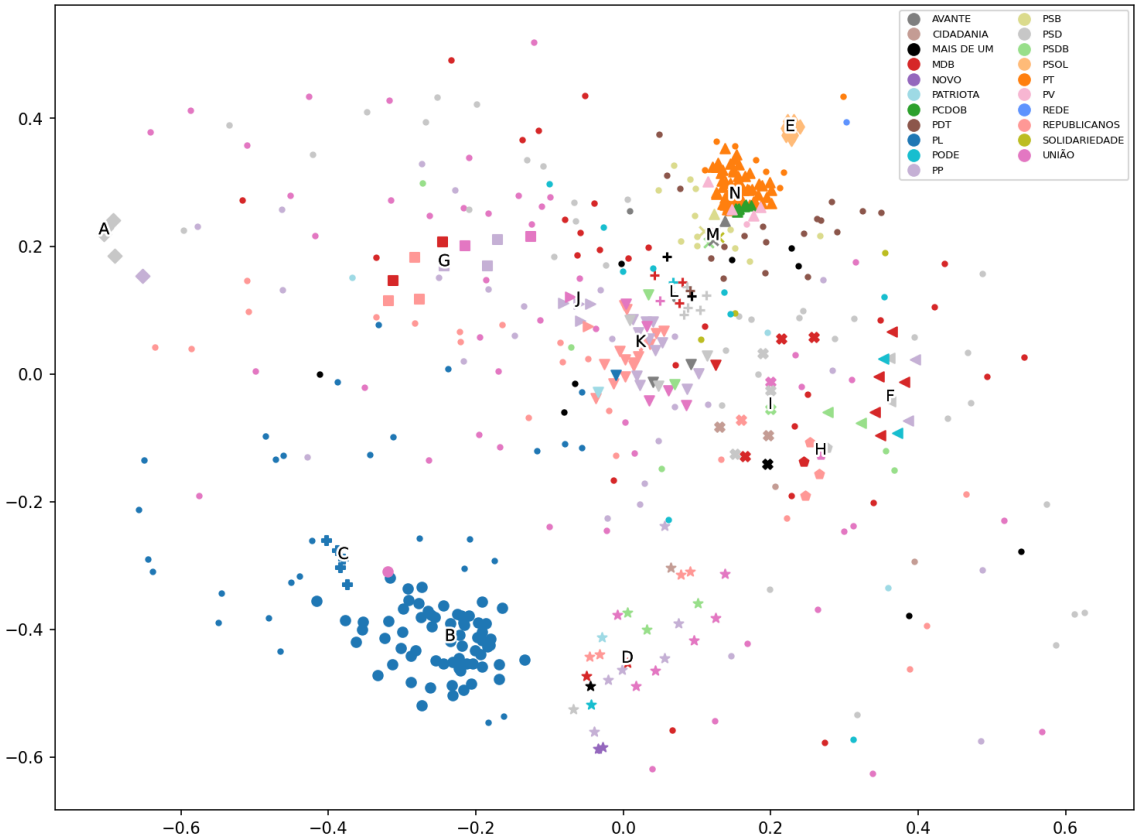
Métrica	**Valor**
Confiabilidade	94.9%
Outliers (HDBSCAN)	8.7%
Homogeneidade partidária (HDBSCAN)	80.6%
Homogeneidade partidária (Propagação por Afinidade)	70.5%

Em 2019, a composição partidária do Congresso sofreu mudanças significativas. O PSL, partido associado à candidatura de Jair Bolsonaro, que eventualmente se uniria ao DEM para formar o partido União, emergiu como uma força expressiva na Câmara dos

Deputados, juntamente com um aumento no número de cadeiras dos partidos PP, PSD e Republicanos [Caesar 2018]. Mais tarde, ao romper com o PSL, legisladores alinhados ao presidente migraram para o PL e partidos aliados [Porto et al. 2022].

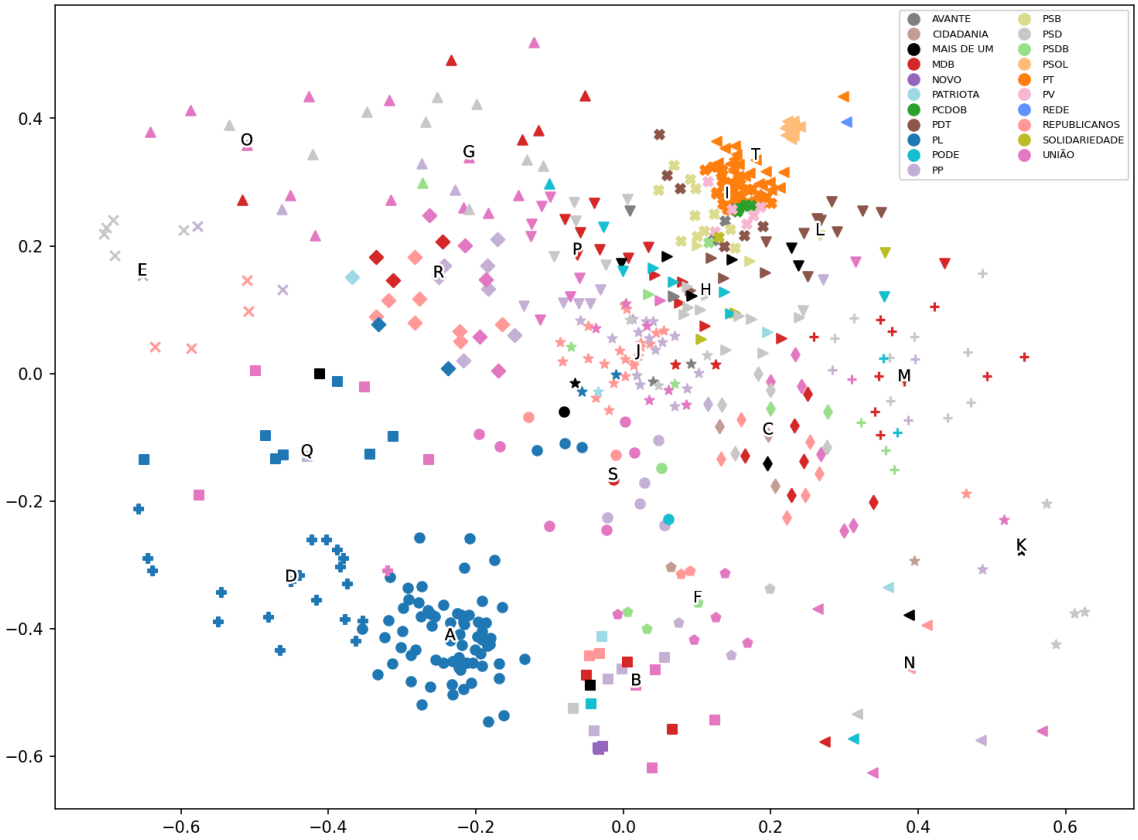
A oposição ao governo Bolsonaro foi liderada pelos partidos PT, PSB, PDT, PSOL, PC do B e REDE, que votaram contra o governo em mais de 50% das ocasiões [Estadão 2019]. O mapa ideológico gerado pelo algoritmo t-SNE (Figura 7 e 8) exibe uma clara divisão entre os parlamentares alinhados ao governo e à oposição, e as mudanças na composição partidária são evidenciadas.

Figura 9. Mapa ideológico da Câmara dos Deputados de 2023 - MDS e HDBSCAN



As diferentes formas representam diferentes clusters e pontos menores representam outliers. Letras representam os centros dos diferentes clusters, onde (A) Leandre, (B) Alberto Fraga, (C) Vermelho, (D) Pezenti, (E) Tarcísio Motta, (F) Diego Andrade, (G) Mauricio Neves, (H) Lebrão, (I) Vitor Lippi, (J) Eduardo da Fonte, (K) Gustinho Ribeiro, (L) Waldemar Oliveira, (M) Pastor Sargento Isidório, (N) Padre João. Quanto mais próximos dois parlamentares, mais parecidos foram seus votos em plenário no período. Fonte: autoria própria.

Figura 10. Mapa ideológico da Câmara dos Deputados de 2023 - MDS e Propagação por Afinidade



As diferentes formas representam diferentes clusters e pontos menores representam outliers. Letras representam os centros dos diferentes clusters, onde (A) Alberto Fraga, (B) Alfredo Gaspar, (C) Amom Mandel, (D) André Ferreira, (E) Arthur Lira, (F) Daniel Trzeciak, (G) Danilo Forte, (H) Diego Coronel, (I) Dimas Gadelha, (J) Gilberto Abramo, (K) Gilberto Nascimento, (L) Heitor Schuch, (M) Hercílio Coelho Diniz, (N) Julio Cesar Ribeiro, (O) Luciano Bivar, (P) Olival Marques, (Q) Pedro Lupion, (R) Pinheirinho, (S) Thiago Flores, (T) Waldenor Pereira. Quanto mais próximos dois parlamentares, mais parecidos foram seus votos em plenário no período. Fonte: autoria própria.

Tabela 4. Métricas do modelo MDS da Câmara dos Deputados de 2023

Métrica	**Valor**
Tensão normalizada pela escala	25.3%
Outliers (HDBSCAN)	45.9%
Homogeneidade partidária (HDBSCAN)	35.1%
Homogeneidade partidária (Propagação por Afinidade)	47.7%

Por fim, em 2023, observou-se uma reconfiguração significativa das alianças políticas no Brasil. Partidos que anteriormente compunham a base de apoio ao governo de Jair Bolsonaro passaram a integrar a base governista do PT, agora no comando do governo federal. No entanto, essa nova adesão ocorreu de forma bastante fragmentada e heterogênea, refletindo um cenário de alinhamento pragmático ao invés de ideológico

[Mattos 2024]. Estas características, juntamente com a oposição por parte do PL, são refletidas no mapa ideológico gerado pelo algoritmo MDS (Figura 9 e 10).

4. Conclusão

O desenvolvimento da biblioteca explicitou a precária infraestrutura de dados disponível para análise política no Brasil, evidenciando a necessidade de soluções que contornem as limitações das APIs públicas. A utilização de um banco de dados local se mostrou eficaz para mitigar os problemas de instabilidade e ineficiência das APIs, permitindo a coleta de dados de maneira incremental e eficiente.

Quanto aos algoritmos, os modelos produzidos pelo algoritmo PCA apresentaram consistentemente uma variância total explicada abaixo do ideal (Tabela 5). Esse desempenho limitado pode ser atribuído a características do cenário político brasileiro, como a presença de um "centrão", que tende a tornar os dados mais esféricos, e a diversidade de partidos, que dificulta a separação linear dos dados.

Esses fatores podem impactar a efetividade do PCA, especialmente quando comparado ao uso desse método em contextos como o do Congresso dos Estados Unidos. Além disso, ao ser aplicado a períodos prolongados, o PCA pode introduzir artefatos devido à não linearidade dos dados, resultante de mudanças no posicionamento relativo dos parlamentares ao longo do tempo, como observado em 2023. Em contrapartida, o PCA apresentou o melhor desempenho computacional entre os métodos testados.

Tabela 5. Média das métricas dos modelos gerados pelo PCA

Métrica	**Câmara**		**Senado**	
	Média	**Desvio padrão**	**Média**	**Desvio padrão**
Variância total explicada	55%	5%	43.5%	5.8%
Outliers (HDBSCAN)	24.9%	13.2%	18.8%	13.9%
Homogeneidade partidária (HDBSCAN)	46.5%	10.6%	35.4%	9.5%
Homogeneidade partidária (Propagação por Afinidade)	54.8%	7%	47.8%	5.4%

Média das métricas de 6 modelos, dos períodos 2003-2006 (Lula 1), 2007-2010 (Lula 2), 2011-2014 (Dilma 1), 2015-2016 (Dilma 2), 2016-2018 (Temer) e 2019-2022 (Bolsonaro).

Semelhantemente, os modelos gerados pelo MDS, especificamente da Câmara dos Deputados, apresentaram uma tensão acima do ideal (Tabela 6). Além disso, uma quantidade significativa de outliers foi observada ao ser clusterizado com HDBSCAN, sugerindo que este algoritmo de clusterização teve dificuldade com a natureza esparsa dos modelos gerados pelo MDS.

Tabela 6. Média das métricas dos modelos gerados pelo MDS

Métrica	Câmara		Senado	
	Média	**Desvio padrão**	**Média**	**Desvio padrão**
Tensão normalizada pela escala	26.5%	2.5%	19.9%	2.2%
Outliers (HDBSCAN)	31.6%	15.9%	20.5%	14%
Homogeneidade partidária (HDBSCAN)	39.7%	8.6%	23.3%	10%
Homogeneidade partidária (Propagação por Afinidade)	56.4%	6.2%	55.9%	3.6%

Média das métricas de 6 modelos, dos períodos 2003-2006 (Lula 1), 2007-2010 (Lula 2), 2011-2014 (Dilma 1), 2015-2016 (Dilma 2), 2016-2018 (Temer) e 2019-2022 (Bolsonaro).

Em contraste, tanto o t-SNE quanto o UMAP geraram mapas com confiabilidade superior a 90% na maioria dos períodos analisados (Tabelas 7 e 8). Além disso, ambos os algoritmos foram mais eficazes em capturar a estrutura partidária dos dados, como evidenciado pela maior homogeneidade partidária nos clusters gerados. Entretanto, esses algoritmos apresentaram o pior desempenho computacional.

Tabela 7. Média das métricas dos modelos gerados pelo t-SNE

Métrica	Câmara		Senado	
	Média	**Desvio padrão**	**Média**	**Desvio padrão**
Confiabilidade	94.9%	2%	92.4%	2%
Outliers (HDBSCAN)	12.6%	9.6%	21.7%	9%
Homogeneidade partidária (HDBSCAN)	77.4%	7%	44.7%	11.4%
Homogeneidade partidária (Propagação por Afinidade)	70.3%	1.6%	56%	7.2%

Média das métricas de 6 modelos, dos períodos 2003-2006 (Lula 1), 2007-2010 (Lula 2), 2011-2014 (Dilma 1), 2015-2016 (Dilma 2), 2016-2018 (Temer) e 2019-2022 (Bolsonaro).

Tabela 8. Média das métricas dos modelos gerados pelo UMAP

Métrica	Câmara		Senado	
	Média	**Desvio padrão**	**Média**	**Desvio padrão**
Confiabilidade	92.4%	2.7%	90.3%	2.1%
Outliers (HDBSCAN)	2.4%	2.9%	5%	8.5%
Homogeneidade partidária (HDBSCAN)	78.6%	5.4%	38.5%	19.8%
Homogeneidade partidária (Propagação por Afinidade)	70%	7.8%	50.6%	10.9%

Média das métricas de 6 modelos, dos períodos 2003-2006 (Lula 1), 2007-2010 (Lula 2), 2011-2014 (Dilma 1), 2015-2016 (Dilma 2), 2016-2018 (Temer) e 2019-2022 (Bolsonaro).

Já sobre os algoritmos de clusterização, o HDBSCAN produziu clusters mais significativos, que melhor representaram as coalizões políticas presentes. No entanto, nenhum dos algoritmos foi capaz de gerar clusters satisfatórios quando parametrizados para classificar os parlamentares em grupos mais amplos, como governo e oposição, ou em blocos ideológicos. Para esses casos, pode ser necessário o uso de algoritmos onde o número de clusters seja definido a priori, permitindo uma análise mais direcionada.

Referências

Brigadir, I., Greene, D., Cross, J. P., and Cunningham, P. (2016). Dimensionality reduction and visualisation tools for voting record. In *Greene, D., Mac Namee, B. and Ross, R.(eds.). Proceedings of the 24th Irish Conference on Artificial Intelligence and Cognitive Science*. CEUR Workshop Proceedings.

Caesar, G. (2018). Saiba como eram e como ficaram as bancadas na Câmara dos Deputados, partido a partido. `https://g1.globo.com/politica/eleicoes/2018/eleicao-em-numeros/noticia/2018/10/08/pt-perde-deputados-mas-ainda-tem-maior-bancada-da-camara-psl-de-bolsonaro-ganha-52-representantes.ghtml`. Acesso em: 1 nov. 2024.

da Silva, R. N. M., Spritzer, A., and Freitas, C. D. S. (2018). Visualization of roll call data for supporting analyses of political profiles. In *2018 31st SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, pages 150–157. IEEE.

Estadão (2019). Basômetro: quanto apoio o governo tem na Câmara? `https://arte.estadao.com.br/politica/basometro/`. Acesso em: 6 mai. 2024.

Hagopian, F. and Mainwaring, S. P. (2005). *The third wave of democratization in Latin America: advances and setbacks*. Cambridge University Press.

Haje, L. (2016). Janela para troca partidária permitiu mais de 90 mudanças entre legendas. `https://www.camara.leg.br/noticias/483820-janela-para-troca-partidaria-permitiu-mais-de-90-mudancas-entre-legendas/`. Acesso em: 1 nov. 2024.

Jolliffe, I. T. and Cadima, J. (2016). Principal component analysis: a review and recent developments. *Philosophical transactions of the royal society A: Mathematical, Physical and Engineering Sciences*, 374(2065):20150202.

Kruskal, J. B. (1964). Nonmetric multidimensional scaling: a numerical method. *Psychometrika*, 29(2):115–129.

Lee, L., Zhang, S., and Yang, V. C. (2017). Do two parties represent the US? Clustering analysis of US public ideology survey. *arXiv preprint arXiv:1710.09347*.

Mattos, M. (2024). Hoje aliados, partidos que dividem o governo não garantem apoio em 2026. `https://veja.abril.com.br/politica/hoje-aliados-partidos-que-dividem-o-governo-nao-garantem-apoio-em-2026`. Acesso em: 1 nov. 2024.

Pitombo, J. P. (2016). Isolado, PT lava as mãos em 12 capitais no segundo turno. `https://www1.folha.uol.com.br/poder/eleicoes-2016/2016/`

10/1822297-isolado-pt-lava-as-maos-em-12-capitais-no-segundo-turno.shtml. Acesso em: 1 nov. 2024.

Poole, K. T. and Rosenthal, H. (1985). A spatial model for legislative roll call analysis. *American journal of political science*, pages 357–384.

Porto, D., Andrade, H., Bridi, C., and Menezes, N. (2022). Eleições 2022: veja quem mudou de legenda no fim da janela partidária. https://www.cnnbrasil.com.br/politica/eleicoes-2022-veja-quem-mudou-de-legenda-no-fim-da-janela-partidaria/. Acesso em: 1 nov. 2024.

Ribeiro, R. L. M. (2014). Decadência longe do poder: refundação e crise do PFL. *Revista de Sociologia e Política*, 22:5–37.

Smelser, K., Miller, J., and Kobourov, S. (2024). "Normalized Stress"is Not Normalized: How to Interpret Stress Correctly. *arXiv preprint arXiv:2408.07724*.

Ulhôa, R. and Costa, R. (2002). PSDB e PFL reagem ao acordo PT-PMDB. https://www1.folha.uol.com.br/folha/brasil/ult96u42226.shtml. Acesso em: 1 nov. 2024.