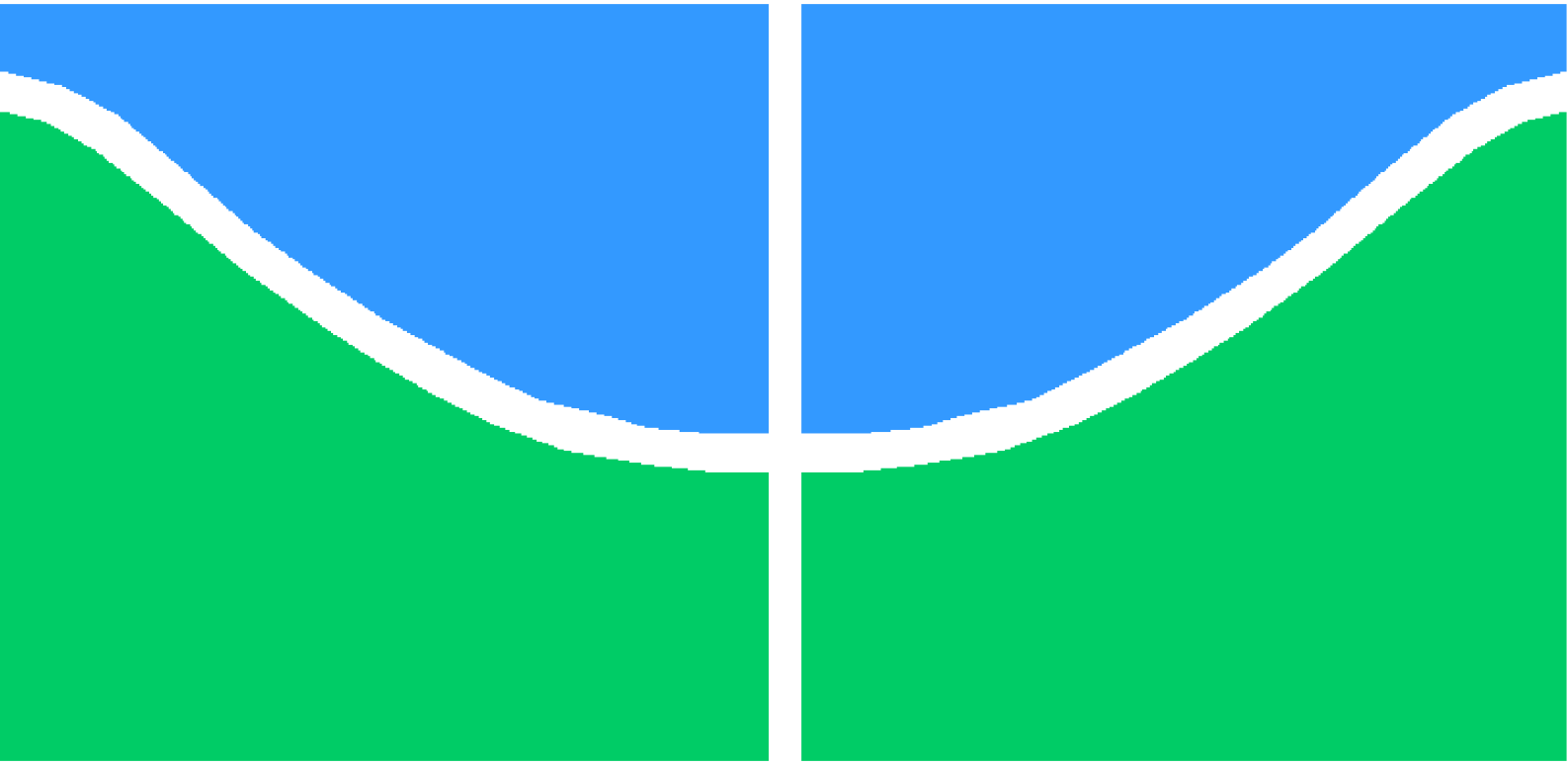


Universidade de Brasília - UnB
Faculdade UnB Gama - FGA
Engenharia de Software

Análise de textos parlamentares

Autor: Matheus Souza Fernandes
Orientador: Prof. Dr. Fábio Macedo Mendes

Brasília, DF
2016

Matheus Souza Fernandes

Análise de textos parlamentares

Monografia submetida ao curso de graduação em (Engenharia de Software) da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em (Engenharia de Software).

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Orientador: Prof. Dr. Fábio Macedo Mendes

Brasília, DF

2016

Matheus Souza Fernandes
Análise de textos parlamentares/ Matheus Souza Fernandes. – Brasília, DF, 2016-
57 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Dr. Fábio Macedo Mendes

Trabalho de Conclusão de Curso – Universidade de Brasília - UnB
Faculdade UnB Gama - FGA , 2016.

1. processamento de linguagem natural. 2. aprendizado de máquina. I. Prof. Dr. Fábio Macedo Mendes. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Análise de textos parlamentares

CDU 02:141:005.6

Matheus Souza Fernandes

Análise de textos parlamentares

Monografia submetida ao curso de graduação em (Engenharia de Software) da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em (Engenharia de Software).

Trabalho aprovado. Brasília, DF, 01 de junho de 2013:

Prof. Dr. Fábio Macedo Mendes
Orientador

Paulo Roberto Miranda Meirelles
Convidado 1

Titulação e Nome do Professor
Convidado 02
Convidado 2

Brasília, DF
2016

Resumo

O resumo deve ressaltar o objetivo, o método, os resultados e as conclusões do documento. A ordem e a extensão destes itens dependem do tipo de resumo (informativo ou indicativo) e do tratamento que cada item recebe no documento original. O resumo deve ser precedido da referência do documento, com exceção do resumo inserido no próprio documento. (. . .) As palavras-chave devem figurar logo abaixo do resumo, antecedidas da expressão Palavras-chave:, separadas entre si por ponto e finalizadas também por ponto. O texto pode conter no mínimo 150 e no máximo 500 palavras, é aconselhável que sejam utilizadas 200 palavras. E não se separa o texto do resumo em parágrafos.

Palavras-chaves: latex. abntex. editoração de texto.

Abstract

This is the english abstract.

Key-words: latex. abntex. text editoration.

Lista de ilustrações

Lista de tabelas

Listings

Sumário

1 Introdução

1.1 Contextualização

O desenvolvimento de novas ferramentas de interação entre governo e sociedade é fundamental para o avanço da democracia (MAZONI, 2011). Porém, torna-se imprecindível a aplicação do conceito de Dados Abertos por parte do Governo para que essas novas ferramentas tenham efeitos significativos para a sociedade.

O conceito para Dado Aberto considerado neste trabalho é o definido pela Open Knowledge (2016), que estabele que um dado (ou um conhecimento) é aberto quando estiver livre para uso, reuso e redistribuição. Ou seja, a informação deve estar disponível a todos, sem restrições de *copyright*, patentes ou outros mecanismos de controle. Além disso, esses dados devem ser independentes de tecnologia, baseados em formatos padronizados e desvinculados de ferramentas que os originaram. Os dados devem permitir sua manipulação por máquinas e possuir metadados que permitam identificar sua natureza, origem e qualidade (DINIZ, 2010).

No contexto governamental, os Dados Abertos fortalecem três características indispensáveis para a democracia: transparência, participação e colaboração (MAZONI, 2011). Transparência tem o papel de informar a sociedade sobre as ações que estão sendo tomadas ou que serão tomadas pelo governo. Participação permite que os cidadãos auxiliem o poder público a elaborar políticas mais eficazes. Finalmente, a colaboração entre a sociedade, diferentes níveis de governo e a iniciativa privada permitem aprimorar a eficácia do Estado.

De acordo com a Lei n°12.527/2011, também conhecida como Lei de Acesso à Informação, qualquer cidadão, sem necessidade de justificativa, pode solicitar dados ou informações à qualquer órgão ou entidade pública dos poderes Executivo, Legislativo e Judiciário, além do Ministério Público, nas esferas Federal, Estadual e Municipal (BRASIL, 2011). Para atender à lei mencionada anteriormente, a Câmara dos Deputados (2016) criou um portal que tem como objetivo disponibilizar dados brutos para a utilização em aplicações desenvolvidas pelos cidadãos e entidades da sociedade civil que permitam a percepção mais efetiva das atividades parlamentares, .

A publicação de dados pressupõe o uso de tecnologias que garantam que eles possam ser acessados e reutilizados por máquinas. Apesar de não garantir que os dados estarão disponíveis em um formato conveniente de uso imediato, possibilita o cruzamento de diferentes bases dados e a exibição destes de forma que possam ser melhor apresentados à sociedade (DINIZ, 2010).

1.2 Objetivo

O objetivo desse trabalho é utilizar técnicas de processamento de linguagem natural para, através da análise dos discursos e proposições dos parlamentares, determinar um perfil temático para os deputados, bem como evidenciar o termos mais utilizados em seus discursos e proposições e, assim, permitir que o cidadão veja, de forma comparativa, o que seus representantes no parlamento mais dizem em seus discursos e o que mais dizem em suas proposições.

1.2.1 Contribuições Tecnológicas

- Implementar biblioteca Python para consumo de dados abertos governamentais, com foco nos dados abertos da Câmara dos deputados.

- Implementar aplicação Django para utilização e persistência dos dados abertos governamentais, também com foco nos dados aberto da Câmara dos deputados.

- Implementar sistema web de comparação entre discursos e proposições parlamentares, a ser detalhado no decorrer deste trabalho.

1.2.2 Contribuições Científicas

- Estudo teórico sobre Processamento de Linguagem Natural.

- Estudo teórico sobre métodos estatísticos aplicados à análise de textos.

1.3 Metodologia

Devido à natureza deste trabalho, nota-se que o modelo de pesquisa adequado deve possuir características tanto da pesquisa exploratória quanto da pesquisa experimental. Outro método de pesquisa que será utilizado é a pesquisa-ação, onde existirão ciclos de coleta e análise de dados. A cada ciclo, a análise dos dados do ciclo anterior servirão de insumo para tomadas de decisão no desenvolvimento do projeto.

Durante a fase de desenvolvimento do sistema, pretende-se utilizar uma abordagem de ágil, com a aplicação de algumas práticas adaptadas do *framework* Scrum:

- Nas retrospectivas de *sprint*, serão realizadas avaliações, para identificar os pontos fortes e fracos da *sprint* que terminou, bem como propor possíveis ações que poderão ser tomadas para mitigar os pontos fracos, na próxima *sprint*.

- *Daily meetings*, onde serão feitos comentários sobre o que foi realizado no dia e dúvidas poderão ser reportadas. Acontecerão diária e remotamente, para o melhor acompanhamento do orientador.
- Será definido um *backlog* de produto e um para cada *sprint*.

1.4 Organização do Trabalho

2 Processamento de Linguagem Natural

Este capítulo contém o referencial teórico que diz respeito ao Processamento de Linguagem Natural.

2.1 Pré-processamento: modelo *bag-of-words*

A grande parte dos dados que serão utilizados nesse trabalho estão dispostos em formato de texto, ou seja, um formato não estruturado que dificulta a extração de informações. Um método comum do processamento de texto em linguagem natural é o modelo *bag-of-words*, onde o texto é representado na forma de um vetor de frequência de palavras que ocorrem no texto (MATSUBARA et al., 2003).

A representação computacional de um texto no modelo mencionado é feita através de um dicionário onde suas chaves são os termos presentes no documento e os valores são suas respectivas frequências. Tomando o trecho "fui à padaria e comprei pão" como exemplo a sua representação no modelo *bag-of-words* corresponde a:

```
bag_of_words = {
    "fui": 1,
    "à": 1,
    "padaria": 1,
    "e": 1,
    "comprei": 1,
    "pão": 1,
}
```

Listing 2.1 – Representação de um trecho no modelo bag-of-words

É fácil ver que esta representação torna a ordem das palavras irrelevante. As frases "e comprei fui padaria pão à" e "à pão comprei padaria e fui" também possuem as mesmas representações.

Um vetor também pode ser uma forma de representar um texto. Nesse caso, cada índice do vetor representa uma palavra e cada elemento corresponde à frequência da mesma.

A linguagem de programação *Python* possui, nativamente, ferramentas que facilitam a contagem de elementos de um dicionário. O `Counter` [1] é uma subclasse de `dict` e assim como a classe pai, é uma coleção não ordenada de elementos. Os elementos são armazenados como chaves de dicionário e seus contadores como valores, que podem assumir

[1] https://docs.python.org/3/library/collections.html

qualquer valor inteiro, incluindo zero e negativos. Isso torna o `Counter` uma representação ideal para o modelo *bag-of-words*, já que ele, naturalmente, conta a quantidade de vezes que um termo aparece dentro de uma coleção (uma lista, tupla ou dicionário, por exemplo).

2.1.1 Representação dos Termos

A representação de um termo a_i pode se dar de diferentes maneiras, como a quantidade de vezes que o termo aparece em um texto ou apenas se o termo aparece no texto, por exemplo. Alguns dos tipos de representação serão expostos a seguir.

2.1.1.1 *Boolean*

Essa medida usa a representação binária para os termos presentes nos documentos, onde o valor de a_i é 0 quando o termo não aparece nenhuma vez no documento ou 1 quando aparece uma ou mais vezes. Essa medida é muito simples e, geralmente, modelos estatísticos levam em consideração também a frequência com que os termos se repetem nos documentos (SALTON; BUCKLEY, 1988).

2.1.1.2 *Term Frequency*

Ao contrário da medida mencionada anteriormente, a medida *term frequency* considera a quantidade de ocorrencias do termo t dentro do documento

$$f_t = \frac{n_t}{N}, \tag{2.1}$$

onde n_t é a quantidade de vezes que o termo t aparece dentro do documento e N a quantidade total de termos do documento.

Alguns termos comuns podem aparecer na maioria dos documentos sem fornecer informações úteis em uma tarefa de mineração de textos (MATSUBARA et al., 2003).

2.1.1.3 *Term Frequency - Inverse Document Frequency*

Para diminuir a influência de termos comuns, é possível utilizar um fator de ponderação, para que os termos que aparecem na maioria dos documentos tenham valores numéricos menores do que aqueles que raramente aparecem (MATSUBARA et al., 2003). Segundo JONES (1972), a especificidade de um termo pode ser quantificada por uma função inversa do número de documentos em que ele ocorre, essa função varia entre 0 e $logN_d$, onde N_d é o número total de documentos e $d(t)$ a quantidade de documentos nos quais o termo t aparece ao menos uma vez:

$$i(t) = log\frac{N_d}{d(t)} \tag{2.2}$$

Portanto, o valor final de a_i é dado pela equação:

$$f(t) = \frac{n_t}{N} \cdot log\frac{N_d}{d(t)}, \tag{2.3}$$

onde $\frac{n_t}{N}$ é a frequência do termo dentro do texto e $log\frac{N_d}{d(t)}$ sua taxa de ponderação.

2.1.2 Modelo *N-Gram*

O modelo de representação de textos através de um conjunto de palavras pode limitar qualitativamente a análise que será realizada, já que frases não são consideradas. Por exemplo, "Santa Catarina", um estado brasileiro, possui um significado completamente diferente quando as palavaras "Santa" (mulher canonizada), e "Catarina" (nome feminino) são analisadas separadamente.

Um n-grama é uma sequência de n elementos dentro de um texto. Os elementos podem ser palavras, sílabas, letras ou qualquer outra base. Um n-grama de tamanho 1 é chamado de unigrama, de tamanho 2, bigrama, e de tamanho 3, trigrama. Sequências com 4 ou mais elementos são chamados de n-gramas. Usando a frase "Eu não gostei desse filme" como exemplo, temos os seguintes unigramas: "eu", "não", "gostei", "desse" e "filme". Os seguintes bigramas: "eu não", "não gostei", "gostei desse" e "desse filme". E os seguintes trigramas: "eu não gostei", "não gostei desse" e "gostei desse filme".

2.1.3 Dimensionalidade dos documentos

A representação vetorial de uma coleção de documentos no modelo *bag-of-words* pressupõe um espaço dimensional igual ao número de termos presentes em toda a coleção de documentos. Suponha que analisou-se 10 documentos e, em média, foram retirados 200 novos termos de cada um. Logo, a dimensionalidade média dos vetores serão de, aproximadamente, 2000. Se a maior parte dos termos aparecer em apenas um ou dois documentos, teremos a maior parte das componentes vetoriais nula. (MATSUBARA et al., 2003). Existem métodos cujo objetivo é diminuir a dimensionalidade desses vetores. Dentre eles, citamos a transformação de cada termo no radical de origem, utilizando algoritmos de *stemming*.

2.1.3.1 Stemização

A stemização (do inglês, *stemming*) é o processo de reduzir palavras flexionadas à sua raiz, essa redução não precisa, necessariamente, chegar à raiz morfológica da palavra. A raiz obtidada geralmente é o suficiente para mapear palavras relacionadas à um valor comum, mesmo se este não for uma raiz válida. O estudo de algoritmos de *stemming* é foco de pesquisas desde a década de 60 e o primeiro algoritmo foi publicado por Lovins (1968).

A consequência da aplicação de algoritmos de *stemming* consiste na remoção de prefixos ou sufixos de um termo e ou da transformação de verbos para suas formas no infinitivo. Por exemplo, as palavras **ouvir**, **ouvi**, **ouviriam**, **ouve** e **ouvindo** seriam reduzidas para um mesmo *stem*: **ouv**. Esse método diminui, portanto, a dimensionalidade dos vetores e dicionários dentro de uma *bag-of-words*. Ao invés de analizar a frequência dos termos, analisamos a quantidade de vezes que um *stem* aparece em um documento.

É evidente que os algoritmos de *stemming* são dependentes do idioma analisado. O algoritmo de Porter (1980), um dos algoritmos de *stemming* mais conhecidos, remove os sufixos de termos em inglês, tem sido amplamente utilizado, referenciado e adaptado desde sua criação. É possível adaptá-lo para a língua portuguesa considerando que as línguas provenientes do latim possuem formas verbais conjugadas em sete tempos e com sete terminações distintas.

Devido ao fato de uma linguagem ter tantas regras e exceções, é pouco provavél que o algoritmo de *stemming* retorne o mesmo *stem* para todas as palavras que tenham a mesma origem ou radical morfológico. Pode-se dizer, também, que a medida que o algoritmo vai se tornando específico o suficiente para atender todas essas regras e exceções a eficiência do algoritmo também diminui (IMAMURA, 2001).

2.1.3.2 *Stop Words*

Palavras que possuem pouco ou nenhum valor semântico, como *"e"*, *"de"* e *"seus"*, são conhecidas como *Stop Words* e, por não agregarem valor à analise textual, são removidas durante o pré-processamento (RAJARAMAN; ULLMAN, 2011). Essas palavras não são exclusividade de uma linguagem específica e geralmente representam a maioria dos termos de um texto. No caso da língua inglesa, por exemplo, palavras como *"of"* e *"the"* também não possuem nenhum valor para a análise. A lista de *Stop Words* obviamente varia de acordo com a linguagem que está sendo analisada (LOPES, 2015). O quadro abaixo mostra a lista de *Stop Words* consideradas neste trabalho.

2.2 Aprendizado de Máquina

Como subárea da inteligência artificial, o aprendizado de máquina tem como objetivo criar técnicas computacionais e sistemas que automaticamente adquirem conhecimento. Existem diversos algoritmos de aprendizado de máquina, utilizados para resolver problemas específicos, portanto, é importante compreender suas limitações (REZENDE, 2003).

As tarefas de aprendizado de máquina podem ser classificadas em três categorias (RUSSEL; NORVIG, 2003):

de	os	tua	tem	estão	da	lhes	essas
e	é	foi	nossas	muito	o	se	tuas
tu	por	as	sua	aquele	entre	não	ele
delas	minhas	às	nos	pela	havia	me	como
ser	aqueles	nossa	vocês	eu	ter	tenho	suas
está	isso	pelos	estes	tinha	depois	foram	este
para	só	quem	deles	isto	um	eles	do
vos	mais	mesmo	num	dele	será	minha	a
no	teus	à	você	em	meus	esses	pelas
com	ao	dela	há	que	na	nosso	te
aos	dos	ou	aquela	era	uma	das	esta
teu	nem	já	até	seja	esse	mas	quando
aquelas	nossos	têm	também	seus	lhe	meu	seu
ela	elas	estas	nós	sem	essa	fosse	qual
		pelo	nas	numa	aquilo		

Tabela 1 – Exemplos de *Stop Words*

- **Apredizagem supervisionada:** é fornecido ao algoritmo um conjunto de entradas e suas respectivas saídas, com o objetivo de aprender uma regra geral que mapeia as entradas às saídas.
- **Aprendizagem não-supervisionada:** nenhum tipo de entrada é fornecido, com o objetivo do próprio algoritmo identificar os padrões do conjunto de dados.
- **Aprendizagem por reforço:** o algoritmo interage com o ambiente dinâmico afim de concluir determinados objetivos.

2.2.1 Aprendizagem Supervisionada

Na aprendizagem supervisionada, cada exemplo de treinamento é descrito por um conjunto de atributos que servem como dados de entrada e são associados a um valor de saída. A partir de um conjunto pré-definido de entradas e saídas, o algoritmo consegue gerar uma saída adequada para uma nova entrada. A aprendizagem supervisionada é a principal técnica utilizada para casos de classificação (MOHRI; ROSTAMIZADEH; TALWALKAR, 2012).

2.2.1.1 Aprendizado Bayesiano

O aprendizado Bayesiano é do tipo supervisionado, já que recebe como *input* dados iniciais e seus respectivos rótulos (ou classes). O algoritmo utiliza fórmulas estatísticas e cálculos de probabilidades para realizar a classificação (MITCHELL, 1997).

As principais vantagens do aprendizado bayesiano são: o fato de se poder embutir nas probabilidades calculadas o conhecimento de domínio que se tem (caso se tenha) e a

capacidade das classificações feitas pelo algoritmo se basearem em evidências fornecidas. Por outro lado, muitas probabilidades devem ser calculadas, o que pode ocasionar em um alto custo computacional. Uma das soluções para o custo do cálculo das probabilidades é o uso do classificador bayes ingênuo (PARDO; NUNES, 2002).

2.2.1.1.1 Classificador *naive bayes*

O classificador bayesiano, também chamado de bayes ingênuo (ou *naive bayes*, em inglês), admite que os atributos do elemento a ser classificado são independentes entre si, mesmo que eles possuam tal dependência (PELLUCCI et al., 2011).

Segundo Oguri, Luiz e Renteria (2007), existem basicamente dois tipos de classificadores bayesianos ingênuos: o modelo binário e o modelo multinomial. O modelo binário representa um documento como um vetor binário, ou seja, o valor 0 em uma posição k (onde k representa uma palavra do documento) representa a não ocorrência do termo e o valor 1 representa ao meno uma ocorrência desse termo. Já o modelo multinomial assume que o documento é representado por um vetor de inteiros, representando a quantidade de vezes que um termo ocorre no documento. Entretanto, quando lidamos com dados contínuos, os seus valores são distribuídos de acordo com uma distribuição Gaussiana (HAND; YU, 2001).

O classificador *naive bayes* é baseado na aplicação do Teorema de Bayes:

$$P(classe|A) = \frac{P(A|classe) \times P(classe)}{P(A)}, \tag{2.4}$$

onde:

- $P(classe)$ é a probabilidade *a priori* da classe em questão,
- $P(A)$ é a probabilidade *a priori* da nova instância a ser classificada,
- $P(A|classe)$ é a probabilidade *a posteriori* de A condicional a *classe* e
- $P(classe|A)$ é a probabilidade *a posteriori* de *classe* condicional a A.

Para calcular a classe mais provável da nova instância, calcula-se as probabilidades de todas as classes possíveis e escolhe-se a classe com maior probabilidade. Em termos estatísticos, isso é equivalente a maximizar $P(classe|A)$, ou seja, maximiza-se o valor do numerador e minimíza-se o valor do denominador. Como o denominador não depende da variável *classe*, pode ser ignorado. Temos então:

$$P(classe|A) = P(A|classe) \times P(classe) \tag{2.5}$$

Considerando que $A = (a_1, a_2, \ldots, a_n)$, onde a_n são os atributos que compõe A, a suposição "ingênua" que o classificador faz é que todos os atributos de A são independentes entre si, o que simplifica o cálculo da probabilidade de $P(A|classe)$, podendo ser reduzida a $P(a_1|classe) \times P(a_2|classe) \times \ \ldots \times P(a_n|classe)$. Logo,

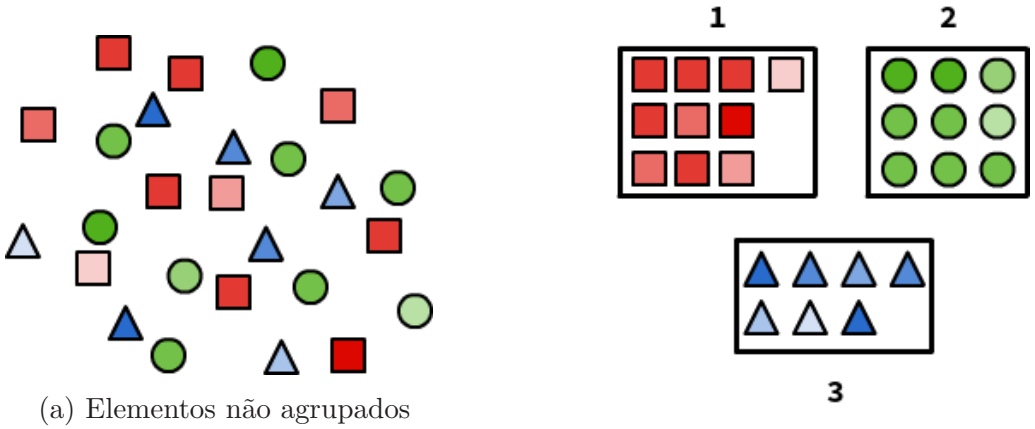
$$P(classe|A) = \prod_{i=1}^{n} P(a_i|classe) \times P(classe) \tag{2.6}$$

2.2.2 Aprendizado Não Supervisionado

Na aprendizagem não supervisionada, os dados de entrada não possuem classes (ou rótulos) e o objetivo do algoritmo é descrever estruturas dentro do conjunto de dados. Uma vez que os dados não são classificados, não existe um erro ou uma recompensa, o que distingue o aprendizado não supervisionado da aprendizagem supervisionada ou por esforço. A aprendizagem não supervisionada é bastante utilizada para resumir e explicar as principais características dos dados (JORDAN; CHRISTOPHER, 2004).

2.2.2.1 Clusterização

Quando temos um conjunto de elementos, naturalmente tentamos estabelecer padrões entre eles. Uma forma natural de definir padrões em um conjunto é analisar a distância entre seus componentes. Dessa forma, quanto mais parecidos dois elementos são, mais próximos eles estão. A figura abaixo mostra um conjunto de elementos com duas características: forma (quadrado, círculo e triângulo) e cor (tons de vermelho, verde e azul). Ao lado temos os mesmos elementos agrupados em três conjuntos com características semelhantes. No grupo 1, por exemplo, todos os elementos possuem uma tonalidade de vermelho e o formato quadrado.



(a) Elementos não agrupados

(b) Elementos agrupados

Figura 1 – Exemplo de clusterização

A clusterização é uma técnica da mineração de dados que consiste, justamente, em realizar o procedimento descrito acima: organizar um conjunto de elementos, usualmente representados por vetores ou pontos em um espaço multidimensional, em *clusters* (ou agrupamentos), de acordo com alguma medida de similaridade. Ela representa uma das principais etapas da análise de dados, denominada análise de *clusters* (JAIN; MURTY; FLYNN, 1999).

Não existe uma técnica de clusterização universal capaz de revelar toda a variedade de estruturas que podem estar presentes em conjuntos de dados multidimensionais. Diferentes algoritmos dependem implicitamente de certas hipóteses a respeito da

forma dos clusters na definição da medida de similaridade e dos critérios de agrupamento (ESTIVILL-CASTRO, 2002).

2.2.2.1.1 Algorítmo *k-means*

O algorítmo de clusterização *k-means*, proposto por Lloyd (1957), tem o objetivo de dividir N elementos em k grupos, onde cada elemento pertence ao *cluster* mais próximo e k deve ser um valor informado a priori e menor ou igual à quantidade de elementos.

Os principais passos do algoritmo são:

1. **Gerar centróides:** neste passo os k centróides recebem valores iniciais. O valor inicial dos centróides podem ser definidos randomicamente, através de uma Gaussiana (com média e variância estimados a partir do conjunto de elementos) ou escolhendo um dos N elementos como centróides iniciais, uniformemente por meio da distribuição uniforme do próprio conjunto de elementos ou escolhendo arbitrariamente os centróides iniciais.

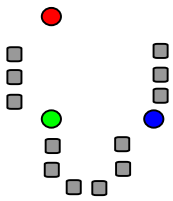


Figura 2 – k centróides (coloridos) recebem valores iniciais.

2. **Calcular distâncias:** aqui são calculadas as distâncias entre cada ponto e cada centróide. É a parte com maior peso computacional do algorítmo, já que o calculo é realizado para cada ponto.

3. **Classificar os pontos:** cada ponto deve ser classificado de acordo com a distância entre ele e o centróide de cada *cluster*. O ponto pertencerá ao *cluster* cujo centróide está mais próximo. O algorítmo converge quando, em uma iteração, nenhum ponto mudar de *cluster*.

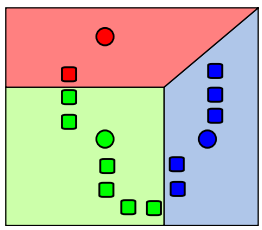


Figura 3 – Cálculo das distâncias entre os pontos e os centróides.

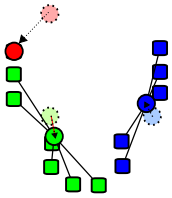


Figura 4 – Novos centróides definidos pela media dos elementos do *cluster*.

4. **Calcular novos centróides:** para cada *cluster*, um novo centróide é definido como a média desses pontos.

5. **Repetir até convergir:** retorna ao passo 2. Como o resultado do algoritmo depende da escolha dos centróides iniciais, a convergência não é garantida. Por isso, normalmente o algoritmo é executado várias vezes.

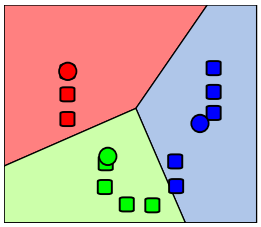


Figura 5 – O algorítmo converge quando nenhum ponto muda de *cluster*.

A unidade utilizada para a medida da distância pode afetar a análise (COLE, 1998). Sugere-se, então, que seja feito o processo de normalização (*whitening*) dos dados antes da clusterização. A normalização consiste em ajustar a escala das distâncias de forma que os valores fiquem em pequenos intervalos, como de 0 a 1, por exemplo.

2.2.2.1.2 Distância entre os pontos

Segundo Cole (1998), para clusterizar termos de acordo com sua similaridade, deve-se definir uma medida de quão próximos dois termos estão. Uma medida de distância deve ser definida de tal forma que:

- Seja sempre positiva.
- Seja simétrica: a distância de um termo a_i para um termo a_j deve ser a mesma de a_j para a_i.
- Seja reflexiva: se a distância entre a_i e a_j é zero, então $a_i = a_j$.
- Respeite a desigualdade triangular: considerando os termos (a_i, a_j e a_k), a distância $d(a_i, a_k)$ deve ser menor ou igual à soma das distâncias $d(a_i, a_j)$ e $d(a_j, a_k)$

O cálculo das distâncias pode ser realizado de diversas maneiras e estão relacionadas à uma distribuição. A distância euclidiana enter dois pontos, $A = (a_1, a_2, a_3, \ldots, a_n)$ e $B = (b_1, b_2, b_3, \ldots, b_n)$ é dada pela equação

$$dist(A, B) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + (a_3 - b_3)^2 + \ldots + (a_n - b_n)^2} \tag{2.7}$$

$$dist(A, B) = \sqrt{\sum_{i=1}^{n}(a_i - b_i)^2} \tag{2.8}$$

Já a distância de Manhattan entre dois pontos, $A = (a_1, a_2, a_3, \ldots, a_n)$ e $B = (b_1, b_2, b_3, \ldots, b_n)$, é dada pela soma das diferenças absolutas de suas coordenadas:

$$dist(A, B) = |a_1 - b_1| + |a_2 - b_2| + |a_3 - b_3| + \ldots + |a_n - b_n| \tag{2.9}$$

$$dist(A, B) = \sum_{i=1}^{n} |a_i - b_i| \tag{2.10}$$

Em ambos os casos, temos que a distância entre do ponto A ao ponto B é a mesma distância do ponto B ao ponto A.

(a) Distância Euclidiana (b) Distância de Manhattan

Figura 6 – Comparação entre distância euclidiana e de manhattan

2.2.3 Validação Cruzada

A validação cruzada é uma técnica geralmente utilizada para avaliar modelos preditivos, buscando estimar o quão preciso é o modelo avaliado. A ideia principal da validação cruzada consiste em dividir o conjunto de dados em subconjuntos mutualmente exclusivos, onde alguns desses subconjuntos serão utilizados para a estimação dos parâmetros do modelo (treinamento) e os outros subconjuntos serão utilizados para a validação do modelo (validação ou teste) (KOHAVI, 1995).

O método *holdout* é uma forma de validação de cruzada comumente utilizada e consiste em dividir o conjunto de dados em dois subconjuntos mutualmente exclusivos, onde seus tamanhos podem, ou não, ser diferentes. Um subconjunto servirá para o treinamento do modelo e o outro para a sua validação. A proporção mais comum é considerar 2/3 dos dados para treinamento e 1/3 para a validação. Essa abordagem é indicada quando está disponível uma grande quantidade de dados, quando o conjunto de dados é muito pequeno, o erro calculado na predição pode sofrer muita variação (KOHAVI, 1995).

Uma das métricas mais simples para avaliar o modelo é a comparação item a item dos resultados esperados pelos resultados obtidos:

$$P_a = \frac{N_{vp}}{N_{it}}, \tag{2.11}$$

onde:

- P_a é a porcentagem de acertos do modelo,
- N_{vp} é o número de itens classificados corretamente pelo modelo e
- N_{it} é o número total de itens que fazem parte do conjunto de testes.

3 Metodologia

Este capítulo aborda o planejamento e execução do projeto, contendo os procedimentos e técnicas utilizadas, possibilitando a sua replicação.

Tendo em vista os conceitos descritos nos capítulos anteriores, o presente trabalho tem como objetivo responder à seguinte questão problema:

É possível extrair o perfil temático dos deputados através da análise dos seus discursos e proposições utilizando técnicas de aprendizado de máquina e processamento de linguagem natural?

Além disso, será construído um *website* com o intuito de fornecer uma forma melhor de visualização dos dados obtidos na análise dos discursos e proposições, através de gráficos interativos que garantam que o usuário tenha uma boa experiência ao utilizá-lo.

3.1 Trabalhos Relacionados

Durante a pesquisa bibliográfica realizada neste trabalho, encontrou-se alguns trabalhos que também fizeram análise de textos parlamentares utilizando aprendizado bayesiano. O Retórica Parlamentar[1], idealizado por Davi Moreira[2], Manoel Galdino[3] e Luis Carli[4], utiliza os discursos proferidos pelos parlamentares no Pequeno Expediente e no Grande Expediente da Câmara dos Deputados para promover a transparência do mandato e fornecer subsídios para o controle social com a divulgação dos temas mais debatidos em Plenário.

Entretanto, a técnica utilizada pelo Retórica para a classificação dos discursos é um modelo bayesiano hierárquico, descrito por Grimmer (2009), onde através de aprendizado não supervisionado são gerados k *clusters*, sendo k um valor escolhido ao executar o algoritmo. O resultado é exportado para o formato *csv* e contém os termos mais frequentes de cada cluster. Um especialista deve ler e definir o rótulo de cada *cluster*.

A visualização dos dados é feita através de um gráfico de bolhas, em que cada bolha representa a relevância (medido pela frequência) de cada tema dentre todos os deputados analisados. Dentro de cada bolha são colocados os deputados que enfatizam aquele tema nos seus discursos. Um deputado está associado a um único tema, que é o tema mais enfatizado por ele nos seus discursos.

[1] http://retorica.labhackercd.net/about.html
[2] https://github.com/davi-moreira
[3] https://github.com/mgaldino
[4] https://github.com/luiscarli

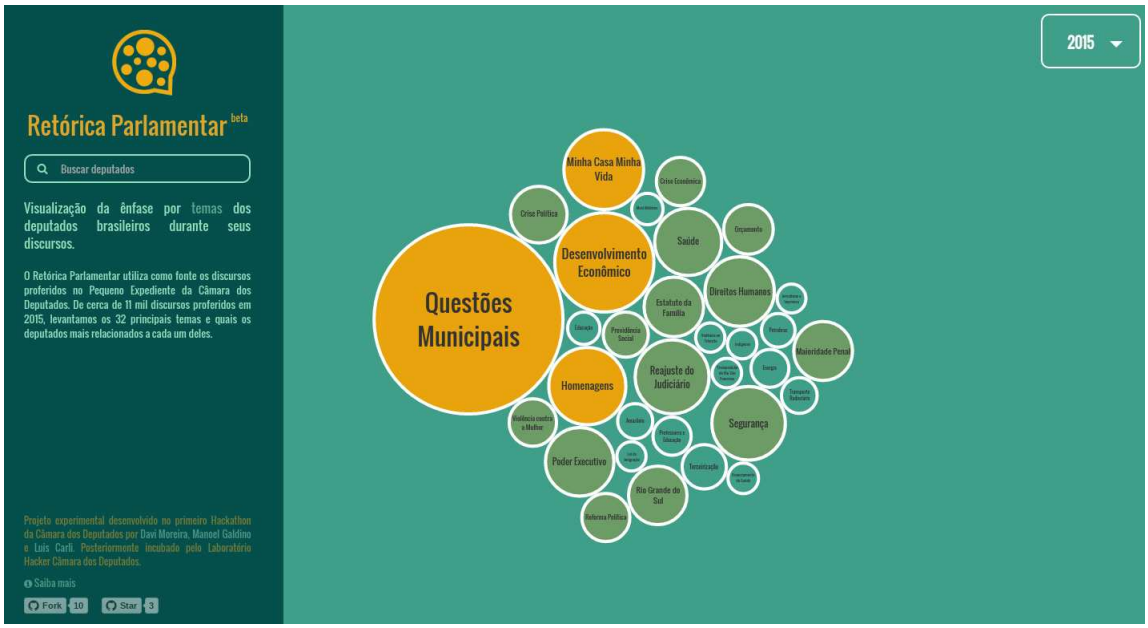


Figura 7 – Retórica Parlamentar

3.2 Obtenção dos Dados

Todos os dados utilizados para análise foram obtidos através do portal de dados abertos da Câmara dos Deputados[5], que é dividido em duas partes: dados legislativos e dados referentes à cota parlamentar, que não foram utilizados nesse trabalho. Os dados legislativos estão relacionados à informações sobre deputados, órgãos legislativos, proposições, sessões plenárias e reuniões de comissões.

3.2.1 *Webservice* da Câmara dos Deputados

Atualmente, o *Webservice* da Câmara dos Deputados é estruturado de acordo com os padrões *SOAP* (*Simple Object Access Protocol*, em português Protocolo Simples de Acesso a Objetos), que se baseiam na linguagem de marcação *XML* e utilizam, principalmente, chamada de procedimento remoto (*RPC*) e protocolo de transferência de hipertexto (*HTTP*) para a transmissão das mensagens (GUDGIN et al., 2007).

No entanto, o *webservice* possui alguns aspectos que podem ser melhorados: o uso de mais de um identificador para os deputados, como por exemplo no *endpoint* **ObterDeputados**, que entre os atributos que são retornados, se encontram "ideCadastro" e "idParlamentar". Outro aspecto que pode ser melhorado é a forma de acesso a certos recursos, como obter todos os discursos de um deputado, por exemplo. Para isso, é necessário fazer uma requisição que retorna todas as sessões que tiveram discursos em um determinado período. Para cada "sessão", temos várias fases de sessão e para cada "fase de sessão", temos um conjunto de discursos. Uma vez com os dados do discurso ("código

[5] http://www.camara.leg.br/transparencia/dados-abertos

da sessão", "ordem", "quarto" e "numero de inserção"), podemos fazer uma requisição para outro *endpoint* e ter acesso ao inteiro teor do discurso, no formato *RTF* e codificado em *base64*.

No momento de escrita desse trabalho, se encontra em desenvolvimento uma nova versão do portal de dados abertos da Câmara dos Deputados, que está em fase de testes e já pode ser acessada pela sociedade[6], porém ainda não possui os mesmos dados disponíveis na versão anterior. O novo *webservice* segue os padrões REST e possibilita a escolha do formato de retorno, podendo ser em *XML* ou em *JSON*. Além disso, visa corrigir os problemas encontrados na versão anterior (alguns mencionados no parágrafo acima), bem como aumentar a quantidade de dados disponíveis. Uma das promessas é disponibilizar o texto completo das proposições, que hoje só é disponível via *PDF* e alguns são apenas imagens *scaneadas* dos documentos físicos.

3.2.1.1 Estrutura do *webservice*

O *webservice* atual (*SOAP*) possui um total de 28 *endpoints*, onde 5 são relacionados aos deputados, 9 aos órgãos, 9 às proposições e 5 às sessões e reuniões. A seguir são descritos cada *endpoint*.

Os *endpoints* que fornecem dados de deputados são:

- **ObterDeputados:** retorna os deputados em exercício na Câmara dos Deputados
- **ObterDetalhesDeputado:** retorna detalhes dos deputados com histórico de participação em comissões, períodos de exercício, filiações partidárias e lideranças.
- **ObterLideresBancadas:** retorna os deputados líderes e vice-líderes em exercício das bancadas dos partidos
- **ObterPartidosCD:** retorna os partidos com representação na Câmara dos Deputados
- **ObterPartidosBlocoCD:** retorna os blocos parlamentares na Câmara dos Deputados.

Os *endpoints* que fornecem dados de órgãos legislativos são:

- **ListarCargosOrgaosLegislativosCD:** retorna a lista dos tipos de cargo para os órgãos legislativos da Câmara dos Deputados (ex: presidente, primeiro-secretário, etc)

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

- **ListarTiposOrgaos:** retorna a lista dos tipos de órgãos que participam do processo legislativo na Câmara dos Deputados
- **ObterAndamento:** retorna o andamento de uma proposição pelos órgãos internos da Câmara a partir de uma data específica
- **ObterEmendasSubstitutivoRedacaoFinal:** retorna as emendas, substitutivos e redações finais de uma determinada proposição
- **ObterIntegraComissoesRelator:** retorna os dados de relatores e pareces, e o link para a íntegra de uma determinada proposição
- **ObterMembrosOrgao:** retorna os parlamentares membros de uma determinada comissão
- **ObterOrgaos:** retorna a lista de órgãos legislativos da Câmara dos Deputados (comissões, Mesa Diretora, conselhos, etc.)
- **ObterPauta:** retorna as pautas das reuniões de comissões e das sessões plenárias realizadas em um determinado período
- **ObterRegimeTramitacaoDespacho:** retorna os dados do último despacho da proposição

Os *endpoints* que fornecem dados de proposições são:

- **ListarProposicoes:** retorna a lista de proposições que satisfaçam os critérios estabelecidos
- **ListarSiglasTipoProposicao:** retorna a lista de siglas de proposições
- **ListarSituacoesProposicao:** retorna a lista de situações para proposições
- **ListarTiposAutores:** retorna a lista de tipos de autores das proposições
- **ObterProposicao:** retorna os dados de uma determinada proposição a partir do tipo, número e ano
- **ObterProposicaoPorID:** retorna os dados de uma determinada proposição a partir do seu ID
- **ObterVotacaoProposicao:** retorna os votos dos deputados a uma determinada proposição em votações ocorridas no Plenário da Câmara dos Deputados
- **ListarProposicoesVotadasEmPlenario:** retorna todas as proposições votadas em plenário num determinado período

- **listarProposicoesTramitadasNoPeriodo:** retorna uma lista de proposições movimentadas em determinado período.

Os *endpoints* que fornecem dados de sessões e reuniões são:

- **ListarDiscursosPlenario:** retorna a lista dos deputados que proferiam discurso no Plenário da Cãmara dos Deputados em um determinado período.
- **ListarPresencasDia:** retorna a lista de presença de deputado em um determinado dia.
- **ListarPresencasParlamentar:** retorna as presenças de um deputado em um determinado período.
- **ListarSituacoesReuniaoSessao:** retorna a lista de situações para as reuniões de comissão e sessões plenárias da Câmara dos Deputados
- **ObterInteiroTeorDiscursosPlenario:** retorna o inteiro teor do discurso proferido no Plenário.

3.3 Proposta de Desenvolvimento

3.3.1 Ferramentas e Tecnologias

Para auxiliar no desenvolvimento, foram escolhidas as seguintes tecnologias:

- **Linguagem de Programação:** devido ao tamanho da comunidade, grande utilização na área de aprendizado de máquina, possibilidade de desenvolvimento *web* e conhecimento prévio, a linguagem a ser utilizada para o desenvolvimento das aplicações do presente trabalho será *Python*.
- ***Frameworks*:** para o desenvolvimento da aplicação *web*, será utilizado o *Django*. Para trabalhar com o processamento de linguagem natural e aprendizado de máquina aplicado, serão utilizadas as bibliotecas *plagiarism* e *texblob*, que encapsula a biblioteca *NLTK*.
- **Gerenciador de Repositórios de Código:** para gerenciar as diversas versões dos códigos das aplicações desenvolvidas nesse trabalho, a ferramenta utilizada será o sistema de controle de versões *Git* e o serviço de *web hosting* compartilhado *GitHub*.
- **Documentação do Projeto:** para realizar a documentção do projeto, será utilizada a biblioteca *python sphinx*, que possibilita a extração de documentação por meio de *docstrings* no código, e o próprio *GitHub*, através de *readmes* no repositório.

- **Gerenciamento de Tarefas:** para o controle de tarefas que serão executadas, estão em execução ou fazem parte do projeto, será utilizado o sistema de *issues* e quadro de projetos do *GitHub*.

3.3.2 *Pygov-br*

Pygov-br é uma biblioteca *python* desenvolvida no contexto desse trabalho cujo objetivo é centralizar o consumo de *APIs* e *webservices* governamentais brasileiros. Além dos dados, a biblioteca também irá fornecer um conjunto de *plugins* para os principais *frameworks* para desenvovimento *web*, para facilitar a utilização dos dados abertos, bem como o cruzamento de dados provenientes de diferentes órgãos governamentais.

Atualmente, a biblioteca oferece suporte somente ao *webservice* da Câmara dos Deputados, tanto para o consumo dos dados quanto para o uso em aplicações *Django*, já que para o desenvolvimento do presente trabalho apenas esses dados seriam utilizados.

A estrutura para o consumo dos *webservices* não é fixa, pois cada *webservice* possui suas características. A implementação para consumir os dados da Câmara dos Deputados segue a estrutura do *webservice* (figura 8), com alterações em alguns *endpoints*, conforme mostra a tabela 2. Além disso, todos o código implementado foi escrito em inglês, apesar dos dados estarem em português.

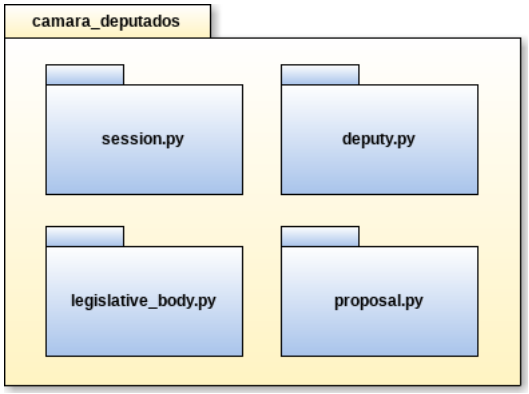


Figura 8 – Estrutura do módulo de consumo de dados da Câmara dos Deputados

Endpoint	**Módulo Final**	**Estrutura do *Webservice***
ListarPresencasParlamentar	deputy.py	SessaoReuniao
ObterAndamento	proposal.py	Orgao
ObterEmendasSubstitutivoRedacaoFinal	proposal.py	Orgao
ObterIntegraComissoesRelator	proposal.py	Orgao

Tabela 2 – Alterações na Estrutura do *Webservice* da Câmara dos Deputados

Como dito anteriormente, a *pygov-br* também possui módulos para utilização em conjunto com os *frameworks* de desenvolvimento *web* mais utilizados na comunidade.

Porém, como a solução *web* desenvolvida nesse trabalho utilizará o *framework Django*, a atual implementação da *pygov-br* possui suporte somente a esse *framework*.

O módulo **django_apps** contém os *plugins* para utilização em projetos *Django*, o que implica na uso da arquitetura *MVT* (*Model View Template*). Similar ao *MVC*, no *MVT* o ciclo começa por uma ação do usuário, a View notifica a Model, para que seu estado seja atualizado, a Model efetua as modificações necessárias e alerta as suas dependências que foi alterada, assim a Template consulta o novo estado da Model, e atualiza a sua visualização.

Entretanto, os *apps Django* da *pygov-br* possuem apenas as *models*, já que o objetivo é somente facilitar a permanência das informações obtidas dos *webservices* governamentais em um banco de dados. No caso da Câmara dos Deputados, os dados utilizados nesse trabalho ficam disponíveis seguindo o modelo entidade-relacionamento na figura 9. Podemos notar que todas as colunas de todas as tabelas se encontram em em inglês, por motivos de padronização do código.

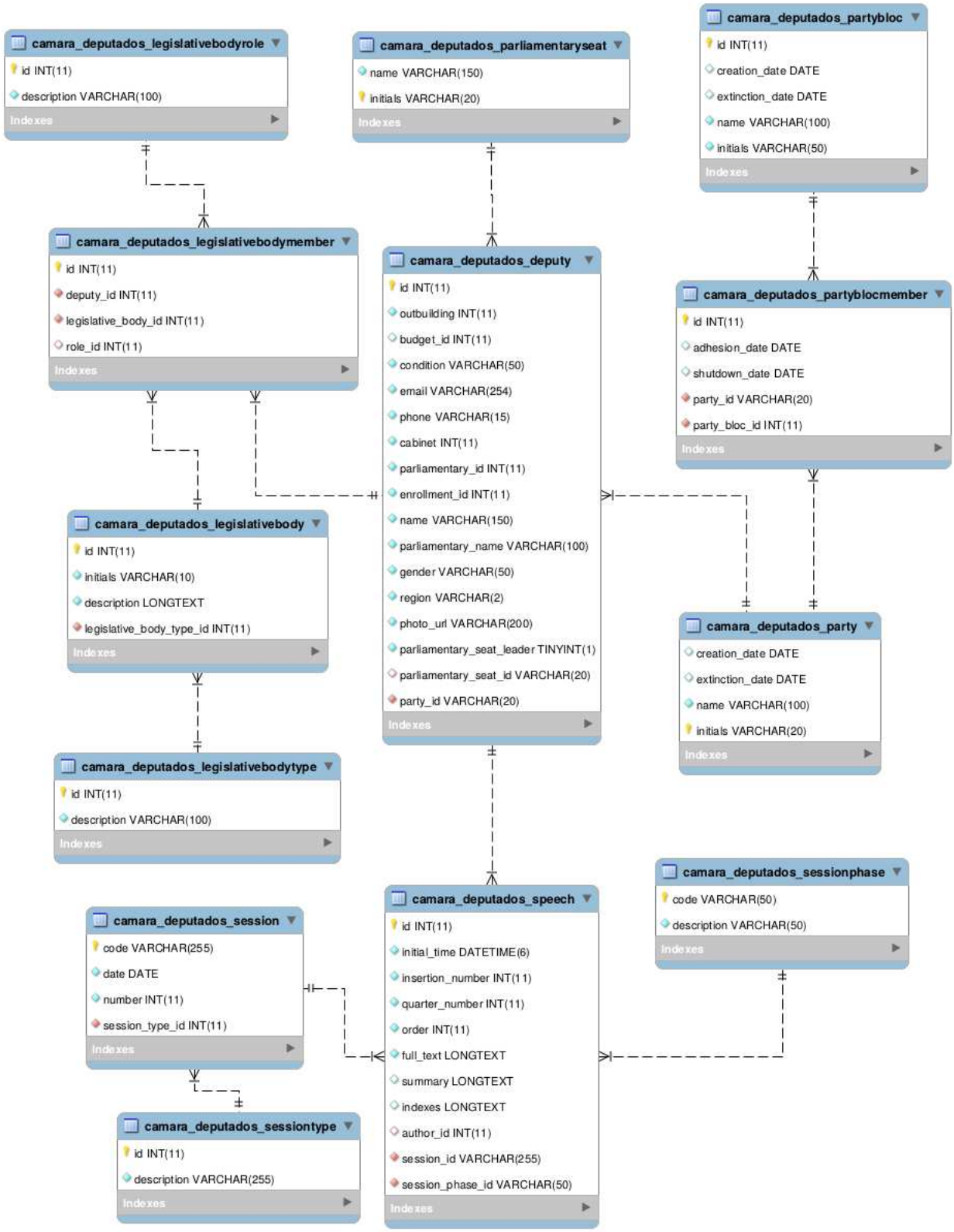


Figura 9 – Modelo entidade-relacionamento do banco de dados utilizado

3.3.3 Tenho Dito

"Tenho Dito" é uma aplicação *web*, desenvolvida utilizando a linguagem *python* com o *framework Django*, e tem como objetivo ser uma forma mais lúdica de visualização dos dados disponíveis nos *webservices* de dados abertos da Câmara dos Deputados. Utiliza métodos de processamento de linguagem natural e aprendizado de máquina para extrair o perfil temático dos parlamentares, analisando o texto de seus discursos e proposições. Além disso, também é possível traçar os temas mais discutidos (tanto em propostas quanto nos próprios discursos) pelos deputados de uma determinada região ou por partidos.

A aplicação é divida em dois grandes módulos: *nlp* e *core*. O primeiro é responsável por todas as operações relacionadas ao processamento dos textos, o que inclui aprendizado de máquina. Já o segundo módulo é responsável pela parte *web*. No momento de escrita desse trabalho, ainda não tinha sido implementado o segundo módulo. Entretanto, estão disponíveis alguns protótipos, que mostram as possíveis funcionalidades do sistema.

Conforme mencionado no item "*frameworks*", em 3.3.1, a análise dos textos será realizada com o apoio das ferramentas:

- **Plagiarism:** biblioteca desenvolvida pelo professor orientador do autor desse trabalho, Fábio Macêdo Mendes, e possui uma série de funcionalidades utilizadas no pré-processamento dos textos, como extração de *tokens*, *stemização*, remoção de *stop words*, geração de *n-gramas* e geração de *bag-of-words* (com os diferentes tipos de representação dos termos, descrito na seção 2.1.1 desse trabalho).
- **Textblob:** biblioteca *python* para processamento de dados textuais. Ela fornece uma interface simples para realizar tarefas comuns do processamento de linguagem natural, como análise de sentimento e classificação, por exemplo. Utiliza a biblioteca *LNTK* para realizar essas tarefas.

3.3.3.1 Classificação dos Discursos

A classificação dos discursos é dividida em duas etapas. A primeira etapa consiste em, inicialmente, dividir o texto em parágrafos, para que a análise seja realizada com uma quantidade menor de texto, e em seguida os parágrafos são classificados entre "protocolo parlamentar" ou "conteúdo". Por exemplo, o trecho "É preciso haver quórum de 257 Srs. Deputados para aprovação da matéria, quórum mínimo. A votação é normal. Então, acho que, quando houver uns 300 ou 320 votos, encerraremos." não representa um conteúdo significativo, da mesma forma que "O SR. ALCEU MOREIRA - Sr. Presidente, primeiro a medida provisória, logicamente." também não agregaria nenhum valor à análise. Trechos como esses devem se classificados como "protocolo parlamentar" e descartados da análise temática. A segunda etapa do processamento é a classificação temática dos parágrafos classificados como "conteúdo", na etapa anterior.

Para ambas etapas o procedimento adotado é o mesmo, com algumas alterações nos classificadores. Primeiro, um classificador *NaiveBayesClassifier*, implementado pela biblioteca *textblob*, é instanciado, utilizando dois conjuntos de palavras iniciais, um para definir "protocolo parlamentar" e outro para "conteúdo", como mostrado a seguir:

- **Protocolo Parlamentar:** "agradecimento agradeço muito obrigado v.exa. digníssimo nobre deputado amigo peço registro pela ordem pedir um aparte mérito emendas votado sessão comissão protocolo regimento pronunciamento divulgação"
- **Conteúdo:** "educação universidade estudante professor ensino escola educador saúde médicos hospitais sus remédios atendimento hospitalar tratamento leitos religião templo igreja deus bíblia fé jesus segurança polícia crime violência punição arma contrabando ditadura militar golpe 31 de março tortura censura mulher aborto feminicídio feminismo feminista maria da penha petrobras pré-sal refinamento gasolina álcool combustível petrolão corrupção ministério público agu lava-jato mensalão impeachment crime de responsabilidade agronegócio agricultura agrícolas soja lavoura rural indústria desendustrialização empregos competitividade direitos humanos minorias tortura tráfego de pessoas trabalho escravo"

Em seguida, todos os parágrafos são classificados e, dentre os que foram classificados com uma probabilidade maior que 80%, os 100 melhores colocados são utilizados para realizar o treinamento inicial do classificador. A partir disso, é realizado um treinamento supervisionado, onde o classificador sugere uma classe e um especialista diz se o trecho corresponde à classe sugerida, caso não seja ele deve fornecer a classe correta. Ao finalizar o treinamento supervisionado, todos os parágrafos são classificados novamente, agora com o classificador melhor treinado.

Com o resultado a primeira classificação, obtém-se um conjunto de parágrafos classificados como "conteúdo", que serão usados na classificação temática. De forma semelhante à primeira classificação, um classificador *NaiveBayesClassifier* é instanciado, agora com um conjunto de palavras para cada tema:

- **Agropecuária:** "agropecuária fertilizantes agronegócio abate suínos ovos cabeças bovinos frangos exportação carne animal milho ração aviária laranja safra frutos pomares laranjeiras fazenda pés produzir hectares quilos fruta produtor orgânico consumidor toneladas embrapa bezerros pecuária veterinária filhotes sementes agro produção água sol área degradação produtor café importação agrícola pescador alimento alimentação açúcar ibge fertilizante lavouras grão bovino soja etanol frutos rural"
- **Saúde:** "saúde médico doença vírus zika pesquisa paciente estudo mosquito epidemia chikungunya tratamento procedimento tremor causa gêmeos dengue transmis-

são cubano bebês cirurgia cientista risco sintomas dor ultrassom dr aegypt ovário microcefalia gravidez sistema imune imunológico drogas fertilização febre diagnóstico renal sangue insuficiente insuficiência cérebro idade nascimento hipotálamo morte dna corpo cardio muscular vacina"

- **Esporte:** "esporte jogo jogador clube time contrato treino mundial atleta surf futebol disputa penalidade compo estádio ataque atacante bola goleiro treinador seleção técnico campeonato gol pontuação futsal vitória perde perdedor lutador torcedor torcida rival diretor falta conquista prorrogação empate surfista assistência ufc"

- **Educação:** "educação estudo ensino escola médio prova enem universidade faculdade matemática avaliação aluno curso pesquisa inep exame pública mec professor redação criança texto reforma currículo curricular campus leitura literatura desempenho formação qualidade disciplina fies superior analfabeto analfabetismo português física química geometria"

- **Ciência e Tecnologia:** "ciência tecnologia novidades empresa startup smart serviço smartphone consumidor produto google aparelho samsung celular internet inteligência artificial desenvolvimento dispositivo lançamento aplicativo inovar inovação sony conectar conectado comunicação 3g 4g 5g iphone sistema telecomunicações satélite design científico artigo computador tráfego eletrônico apple whatsapp televisão tv telefone avanço espacial"

- **Economia:** "economia trabalho crédito compra banco bilhões milhões vendas contas inflação consumidor juros queda crise taxa resultado econômico gasto pagamento valor financeiro investimento dinheiro índice comércio empresa desemprego fgts limite emprego cartão varejo déficite fundo recessão recuo salário lojista tesouro fiscal inadimplente recurso dólar euro moeda bolsa endividado projeções crescimento capital ações negócios"

- **Política:** "política deputado congresso pt partido estado união reforma lei legislatura legislação pec pmdb aprovar voto bancada população senado senador câmara deputado sindicato candidato candidatura mandato comissão ministério constituição eleição eleições delação judiciário votações prefeitura prefeito vereador assembleia procurador corrupção"

- **Meio Ambiente:** "ambiente área água rio empresa desastres multa seca barragem furacão desmatamento floresta tropical ibama parque preservação região terra planeta poluição ambiental espécie animais plantas platações petróleo emissão gás chuva temporal sol clima temperatura estufa aquecimento global umidade terremoto planeta biodiversidade biologia mar oceano calor energia sustentável madeira reflorestamento tempestade niño florescimento hídrico climática"

- **Direitos Humanos:** "direitos humanos mulher tortura violência morte justiça onu sexual vítima sexual adolescente presídio prevenção união negro branco segurança refugiado homens humanitario conflito sociedade racismo sexismo machismo machista feminismo feminista defensoria estupro jovens criança prostituição assassinato liberdade idoso inclusão social preconceito gay homossexual heterosexual lgbt lésbica bissexual travesti transexual transgênero impunidade imigrante"

- **Segurança:** "segurança ataque polícia suspeito morte crime terror rebelde investigação civil federal guerra onu vítima invasão preso presídio assassinato bombardeio apreensão incidente defesa exército marinha aeronáutica prisão ameaça bomba testemunha promotor policial tragédia assalto protesto"

Todos os parágrafos são classificados novamente e é gerado um conjunto com os melhores classificados, que é usado para realizar o treinamento inicial do classificador. E então acontece o treinamento supervisionado, onde um especialista diz se a classificação sugerida faz sentido e indica a classe correta quando não faz.

Também é possível realizar um treinamento não supervisionado para ambos os classificadores, de forma que as sugestões de classificação são utilizadas para o treinamento sem a análise de um especialista.

A cada iteração da fase de treinamento todas as probabilidades dos textos adicionados ao classificador são recalculadas, o que implica no aumento significativo do tempo de processamento.

Referências

BRASIL. *Lei nº 12.527, de 18 de novembro de 2011.Regula o acesso à informações*. 2011. Disponível em: <http://www.planalto.gov.br/ccivil_03/_ato2011-2014/2011/lei/l12527.htm>. Acesso em: 4 out 2016. Citado na página 17.

CÂMARA DOS DEPUTADOS. *Dados Abertos da Câmara dos Deputados*. 2016. Disponível em: <http://www2.camara.leg.br/transparencia/dados-abertos/perguntas-e-respostas>. Acesso em: 4 out 2016. Citado na página 17.

COLE, R. M. Clustering with genetic algorithms. Department of Computer Science, University of Western Australia, Australia, 1998. Citado na página 29.

DINIZ, V. *Como Conseguir Dados Governamentais Abertos*. Brasília, Brasil, 2010. Citado na página 17.

ESTIVILL-CASTRO, V. Why so many clustering algorithms — a position paper. *ACM SIGKDD Explorations Newsletter*, 2002. Citado na página 27.

GRIMMER, J. A bayesian hierarchical topic model for political texts: Measuring expressed agendas in senate press releases. *Department of Government, Harvard University, 1737 Cambridge Street, Cambridge, MA 02138*, 2009. Citado na página 33.

GUDGIN, M. et al. *W3C SOAP Specifications*. 2007. Disponível em: <https://www.w3.org/TR/soap12/>. Citado na página 34.

HAND, D. J.; YU, K. Idiot's bayes — not so stupid after all? *International Statistical Review*, 2001. Citado na página 26.

IMAMURA, C. Y. Pré-processamento para extração de conhecimento de bases textuais. 2001. Citado na página 24.

JAIN, A.; MURTY, M.; FLYNN, P. Data clustering: A review. In: *ACM Computing Surveys*. [S.l.: s.n.], 1999. v. 31, p. 264–323. Citado na página 27.

JONES, K. S. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, v. 28, p. 11–21, 1972. Citado na página 22.

JORDAN, M. I.; CHRISTOPHER, M. Neural networks. In: *Computer Science Handbook*. Second edition. Boca Raton, FL: Chapman & HallCRC Press LLC, 2004. Citado na página 27.

KOHAVI, R. A study of cross-validation and bootstrap for accuracy estimation and model selection. In: *International joint Conference on artificial intelligence*. [S.l.: s.n.], 1995. Citado na página 30.

LLOYD, S. P. Least square quantization in pcm. *IEEE Transactions on Information Theory*, p. 129–137, 1957. Citado na página 28.

LOPES, E. D. Utilização do modelo skip-gram para representação distribuída de palavras no projeto media cloud brasil. 2015. Citado na página 24.

LOVINS, J. B. Development of a stemming algorithm. In: *Mechanical Translation and Computational Linguistics*. [S.l.: s.n.], 1968. Citado na página 23.

MATSUBARA, E. T. et al. *PreTexT: uma ferramenta para pré-processamento de textos utilizando a abordagem bag-of-words*. 2003. Citado 3 vezes nas páginas 21, 22 e 23.

MAZONI, M. V. F. *Dados Abertos para a Democracia na Era Digital*. Brasília, Brasil, 2011. 80 p. Citado na página 17.

MITCHELL, T. M. *Machine Learning*. [S.l.]: McGraw Hill, 1997. Citado na página 25.

MOHRI, M.; ROSTAMIZADEH, A.; TALWALKAR, A. Foundations of machine learning. *The MIT Press*, 2012. Citado na página 25.

OGURI, P.; LUIZ, R.; RENTERIA, R. Aprendizado de máquina para o problema de sentiment classification. *Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro*, 2007. Citado na página 26.

OPEN KNOWLEDGE. About. 2016. Disponível em: <https://okfn.org/about/>. Acesso em: 4 out 2016. Citado na página 17.

PARDO, T. A. S.; NUNES, M. das G. V. Aprendizado bayesiano aplicado ao processamento de línguas naturais. *Série de Relatórios do Núcleo Interinstitucional de Linguística Computacional NILC - ICMC-USP*, 2002. Citado na página 25.

PELLUCCI, P. R. S. et al. Utilização de técnicas de aprendizado de máquina no reconhecimento de entidades nomeadas no português. *Centro Universitário de Belo Horizonte, Belo Horizonte, MG*, 2011. Citado na página 26.

PORTER, M. F. An algorithm for suffix stripping. *Program electronic library and information systems*, 1980. Citado na página 24.

RAJARAMAN, A.; ULLMAN, J. D. Data mining. In: *Mining of Massive Datasets*. [S.l.: s.n.], 2011. Citado na página 24.

REZENDE, S. O. *Sistemas inteligentes: fundamentos e aplicações*. [S.l.]: Ed. Barueri, SP, 2003. Citado na página 24.

RUSSEL, S.; NORVIG, P. *Artificial Intelligence: A Modern Approach*. [S.l.: s.n.], 2003. Citado na página 24.

SALTON, G.; BUCKLEY, C. Term-weighting approaches in automatic text retrieval. 1988. Citado na página 22.

Apêndices

APÊNDICE A – Primeiro Apêndice

Texto do primeiro apêndice.

APÊNDICE B – Segundo Apêndice

Texto do segundo apêndice.

Anexos

ANEXO A – Primeiro Anexo

Texto do primeiro anexo.

ANEXO B – Segundo Anexo

Texto do segundo anexo.