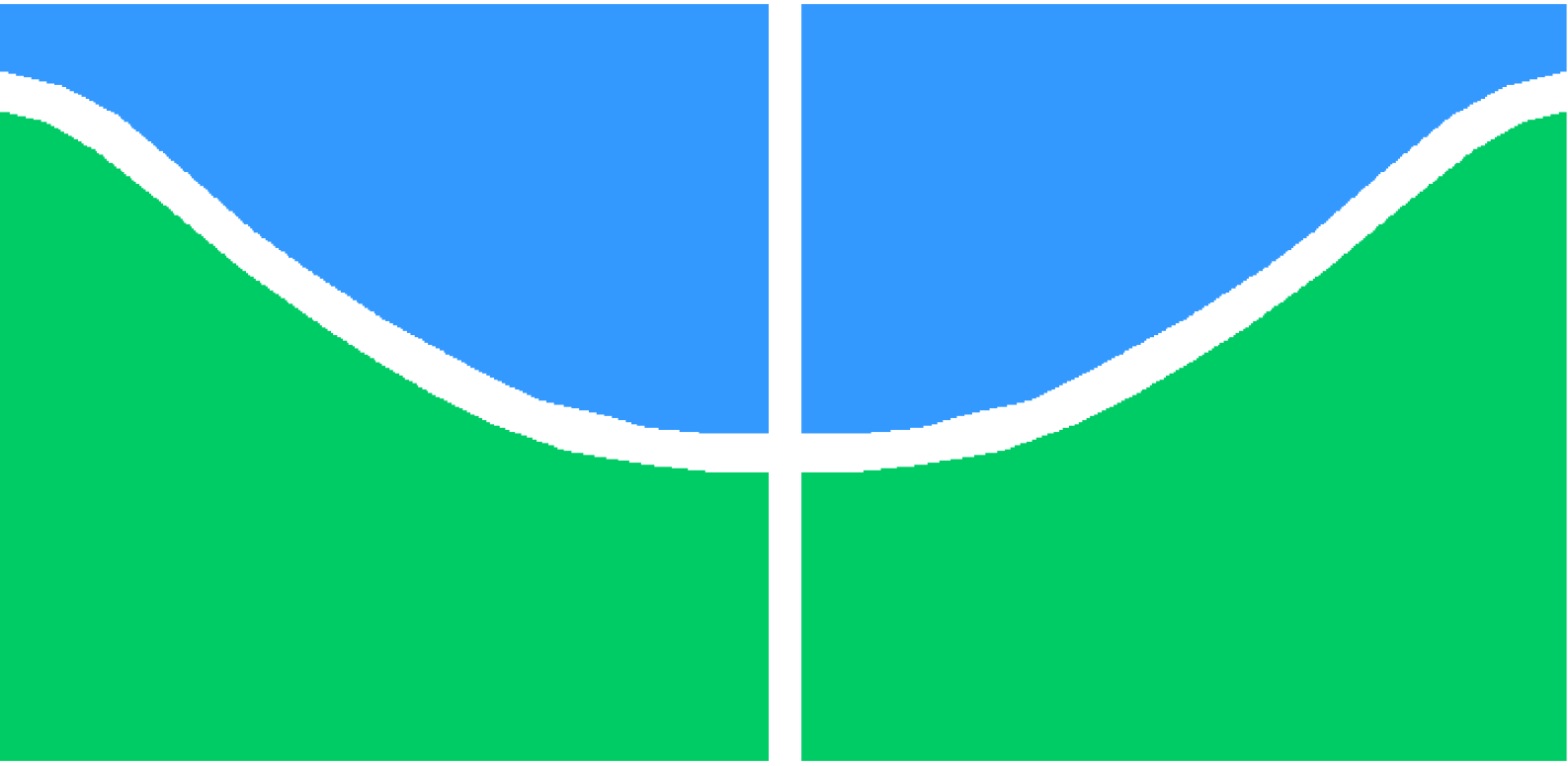


Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Engenharia de Software

Tenho Dito: uma aplicação para análise de discursos parlamentares utilizando técnicas de processamento de linguagem natural

Autor: Matheus Souza Fernandes

Orientador: Prof. Dr. Fábio Macedo Mendes

Brasília, DF

2016

Matheus Souza Fernandes

Tenho Dito: uma aplicação para análise de discursos parlamentares utilizando técnicas de processamento de linguagem natural

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

Tenho Dito: uma aplicação para análise de discursos parlamentares utilizando técnicas de processamento de linguagem natural/ Matheus Souza Fernandes. – Brasília, DF, 2016-

65 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. Tenho Dito: uma aplicação para análise de discursos parlamentares utilizando técnicas de processamento de linguagem natural

CDU 02:141:005.6

Matheus Souza Fernandes

Tenho Dito: uma aplicação para análise de discursos parlamentares utilizando técnicas de processamento de linguagem natural

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

Prof. Dr. Paulo Roberto Miranda Meirelles
Convidado 1

Prof. Dr. Teófilo de Campos
Convidado 2

Brasília, DF
2016

Resumo

O processamento de linguagem natural tem sido utilizado com sucesso na área de análise de discurso, onde é possível reconhecer padrões e classificar textos, extraindo informações de grandes volumes de dados. Este trabalho tem como objetivo extrair o perfil temático dos deputados federais, através do processamento dos textos obtidos de seus discursos e proposições, bem como desenvolver uma aplicação *web* para que os resultados dessa pesquisa sejam apresentados de forma lúdica e amigável. O texto discute as técnicas de processamento de linguagem natural utilizadas nesta análise, que incluem a remoção de *stop words*, algumas técnicas de *stemização*, representação dos textos em *bag-of-words* e algumas técnicas de aprendizagem de máquina supervisionada e não-supervisionada, como *Naive Bayes* e k-means.

Palavras-chaves: Processamento de linguagem natural, aprendizado de máquina.

Abstract

Natural language processing has been used successfully in the discourse analysis where it's possible to recognize patterns and classify texts, extracting information from large volume of data. This paper aim to extract the thematic profile of the federal deputies through the processing of texts obteined from their speeches and proposals, as well as develop a web application so that the results of this research are presented in a playful and friendly way. The text discusses the natural language processing techniques used in this analysis, which include the removal of stop words, some stemming techniques, representation of texts in bag-of-words model and some techniques of supervisioned and unsupervisioned machine learning, such as Naive Bayes and k-means.

Key-words: Natural language processing, machine learning.

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 e a disponibilização de dados governamentais possibilita que novas aplicações surjam, trazendo novas formas de utilização e interpretação destes dados (MAZONI, 2011). O presente trabalho utiliza dados disponíveis publicamente pela Câmara dos Deputados Federal para analisar textos de discursos e proposições parlamentares utilizando técnicas de processamento de linguagem natural e aprendizagem de máquina. O objetivo da análise é determinar eixos temáticos para cada deputado e avaliar o alinhamento entre os temas dos discursos e das propostas encaminhadas.

No contexto governamental, os Dados Abertos[1] 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

[1] 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 das 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).

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 temática 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 aprendizado de máquina aplicado 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

Este trabalho está divido em quatro capítulos. No capítulo 2 são apresentadas as revisões bibliográficas realizadas para pré-processamento de textos e aprendizado de máquina aplicado a processamento de linguagem natural. No capítulo 3 são abordados os trabalhos relacionados, ferramentas utilizadas e o planejamento das atividades. O capítulo 3 apresenta os resultados alcançados com essa pesquisa. Por fim, o capítulo 5 possui as considerações finais, perspectivas futuras e um cronograma para a continuação deste 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* (BOW)

A maior parte dos dados 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 (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 as respectivas frequências. Tomando o trecho "fui à padaria e comprei pão" como exemplo, sua representação no modelo *bag-of-words* corresponderia 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.

O texto também pode ser representado na forma de um vetor. Nesse caso, cada índice do vetor representa uma palavra e cada coordenada corresponde à frequência da mesma. Quando mais de um texto é analisado, os vetores que os representam devem possuir a mesma relação "índice-palavra". Por isso, as palavras que compõe cada texto são ordenadas alfbéticamente antes da formação do vetores.

A linguagem de programação *Python* possui, nativamente, ferramentas que facilitam a contagem de elementos de um dicionário. O módulo `collections`[1] possui a classe

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

`Counter`, que é uma subclasse de `dict` e representa um mapa entre elementos e números. 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 Valores no BOW

Apresentamos o modelo do BOW associando um valor a_i a cada palavra que representa, como a quantidade de vezes que o termo aparece em um texto. Existem, no entanto, outras formas de representação expostas a seguir.

2.1.1.1 *Boolean*

Essa medida associa valores binários para os termos presentes nos documentos, onde o valor de a_i é 0 quando o termo não aparece nenhuma vez ou 1 quando aparece uma ou mais vezes. Essa medida simples é muitas vezes utilizada para a análise de documentos pequenos, como frases e parágrafos.

2.1.1.2 *Term Frequency*

A medida *term frequency* considera a quantidade de ocorrencias do termo t dentro do documento, ao contrário da medida mencionada anteriormente (SALTON; BUCKLEY, 1988)

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

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.

2.1.1.3 *Term Frequency - Inverse Document Frequency*

Alguns termos comuns podem aparecer na maioria dos documentos sem fornecer informações úteis em uma tarefa de mineração de textos. Para diminuir a influência destes termos, 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. Uma alternativa comum é utilizar uma função que 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.

Existem outras formas de ponderação que penalizam os termos mais comuns de formas mais ou menos extremas. O formato logarítmo possui a característica de anular termos que apareçam em todos os documentos, considerando-os portanto, como totalmente não-informativos.

2.1.2 Modelo *N-Gram*

O modelo de representação de textos através de um conjunto de palavras pode limitar qualitativamente a análise, já que termos compostos não são considerados. 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. É comum usar as denominações unigrama, bigrama e trigrama para n-gramas de 1, 2 ou 3 elementos. 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 sera de, aproximadamente, 2000. Se a maior parte dos termos aparecer em apenas um ou dois documentos, teremos a maior parte das componentes vetoriais nulas. (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, mas apenas a um valor útil do ponto de vista computacional. 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, e 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, assim como a dificuldade de implementação aumenta (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, podem ser 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 parcial de *Stop Words* consideradas nesse trabalho.

O algoritmo utiliza a versão *stemizada* das *stop words*. Mostramos as palavras originais por uma questão de clareza.

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 – Lista parcial de *Stop Words* consideradas nesse 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. É importante, portanto, compreender suas limitações (REZENDE, 2003).

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

- **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.

Este trabalho utiliza alguns algoritmos supervisionados e não-supervisionados, mas não adota nenhuma técnica de aprendizagem por reforço.

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 problemas de classificação (MOHRI; ROSTAMIZADEH; TALWALKAR, 2012).

2.2.1.1 Aprendizado Bayesiano

O aprendizado Bayesiano é um conjunto de técnicas baseadas em análise estatística que utilizam a fórmula de Bayes. Normalmente são métodos supervisionados, ainda que alguns algoritmos não-supervisionados possam ser mapeados em métodos bayesianos (MITCHELL, 1997).

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

das classificações feitas pelo algoritmo se basearem em evidências fornecidas e numa análise estatística bem fundamentada. Por outro lado, frequentemente envolvem o cálculo de médias e outras medidas estatísticas que pode ocasionar em um alto custo computacional.

2.2.1.1.1 Classificador *naive bayes*

Uma forma de mitigar a dificuldade de cálculo está em considerar modelos probabilísticos simplificados que permitem um tratamento analítico para as probabilidades calculadas (PARDO; NUNES, 2002).

O classificador bayesiano ingênuo (ou *naive bayes*, em inglês), admite que os atributos do elemento a ser classificado são independentes entre si, dada a categoria da classificação (PELLUCCI et al., 2011).

Segundo Oguri, Luiz e Renteria (2007), existem dois tipos principais de classificadores bayesianos ingênuos utilizados em processamento de linguagem natural: 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 menos uma ocorrência desse termo. Este modelo simplesmente especifica a probabilidade de ocorrência de cada 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. Este modelo também especifica a probabilidade de ocorrência de um termo, mas permite ocorrências múltiplas. Cada modelo está relacionado a um tipo de BOW. Também é possível tratar a BOW TF-IDF como um modelo Gaussiano adequado para variáveis contínuas (HAND; YU, 2001).

Os classificadores Bayesianos são baseados na aplicação do Teorema de Bayes:

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

onde:

- $P(classe)$ é a probabilidade da classe em questão, no contexto do teorema de Bayes, ela é comumente denominada probabilidade *a priori* (ou *prior*)
- $P(A|classe)$ é a probabilidade de obter um conjunto de dados A condicional a $classe$. Isto é conhecido como *likelihood* ou verossimilhança. O modelo Bayes ingênuo assume que $P(A|classe) = \prod_i P(a_i|classe)$
- $P(classe|A)$ é a probabilidade de um elemento pertencer a uma classe dado um conjunto de observações A. Conhecido como o *a posteriori* na literatura Bayesiana

- $P(A)$ é a probabilidade da nova instância a ser classificada. Este termo corresponde ao fator de normalização do posterior e é frequentemente ignorado. Na literatura Bayesiana é conhecido como evidência (JAYNES, 2003)

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 $P(classe|A)$. Como o *prior* é comumente pré-fixado e a evidência não depende da classe, o problema matemático principal consiste em encontrar o valor de máxima verossimilhança.

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,

$$P(classe|A) \propto P(classe) \times \prod_{i=1}^{n} P(a_i|classe) \tag{2.5}$$

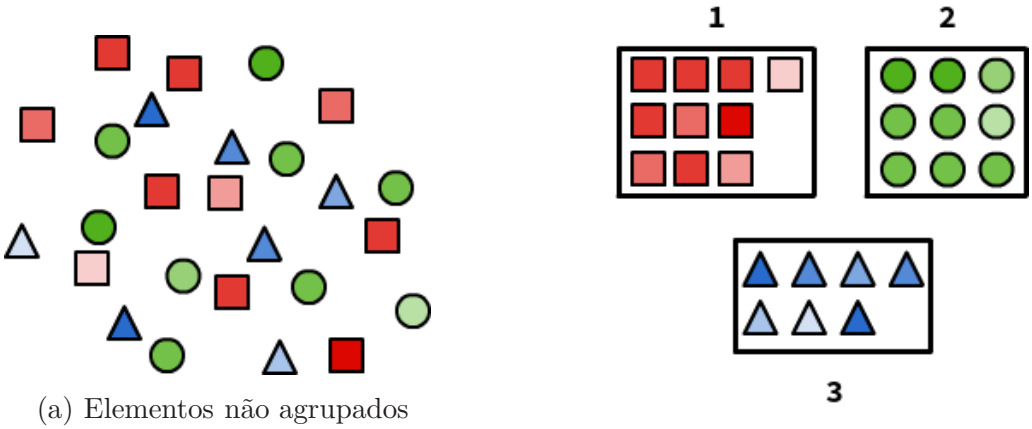
Em problemas de NLP, a_i usualmente correspondem aos valores contidos em uma BOW, ainda que seja possível adicionar outros tipos de atributos para a análise.

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 grande 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, da 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. O valor de k deve ser informado a priori, sendo menor que a quantidade de elementos.

Os 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 aleatoriamente k dos N elementos como centróides iniciais ou definindo-os como centróides de k grupos escolhidos aleatoriamente a partir dos dados iniciais.

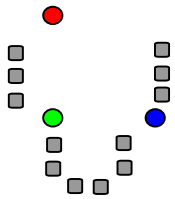


Figura 2 – k centróides (coloridos) recebem valores iniciais.

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*.

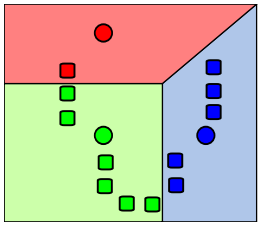


Figura 3 – Cálculo das distâncias entre os pontos e os centróides.

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

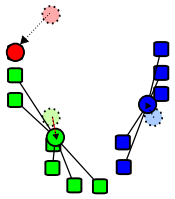


Figura 4 – Novos centróides definidos pela media dos elementos do *cluster*.

5. **Repetir até convergir:** retorna ao passo 2. Como o resultado do algoritmo depende da escolha dos centróides iniciais, a convergência não é garantida ou ele converge para uma solução sub-ótima. Por isso, normalmente o algoritmo é executado várias vezes.

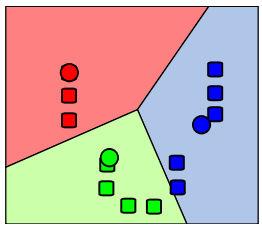


Figura 5 – O algorítmo converge quando nenhum ponto muda de *cluster*.

Mudanças de escala ou de unidade de medidas para determinadas coordenadas dos elementos podem 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 intervalos padronizados, normalmente com média nula e variância 1.

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 (métrica) 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 $d(A_j, A_k)$

Existem várias medidas de distância. Começamos pela distância euclidiana entre dois pontos, $A = (a_1, a_2, a_3, \ldots, a_n)$ e $B = (b_1, b_2, b_3, \ldots, b_n)$, dada pela equação

$$dist(A, B) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + (a_3 - b_3)^2 + \ldots + (a_n - b_n)^2} \tag{2.6}$$

$$dist(A, B) = \sqrt{\sum_{i=1}^{n}(a_i - b_i)^2} \tag{2.7}$$

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:

$$dist(A, B) = |a_1 - b_1| + |a_2 - b_2| + |a_3 - b_3| + \ldots + |a_n - b_n| \tag{2.8}$$

$$dist(A, B) = \sum_{i=1}^{n} |a_i - b_i| \tag{2.9}$$

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

Cada métrica pode gerar resultados diferentes no algoritmo *k-means* e normalmente podem ser interpretadas do ponto de vista estatístico como uma escolha do modelo que gerou cada *cluster*. A distância euclidiana pode ser associada a um modelo Gaussiano para gerar os pontos observados onde a média corresponde ao centróide de cada *cluster* e o desvio padrão é o mesmo para todos os grupos.

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. Uma proporção 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:

$$P_a = \frac{N_{vp}}{N_{it}}, \tag{2.10}$$

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 nesta análise, através de gráficos interativos que garantam que o usuário tenha uma boa experiência de usabilidade.

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.

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 rotular cada *cluster*.

A visualização dos dados é feita através de um gráfico de bolhas, em que cada bolha representa a relevância (medida 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

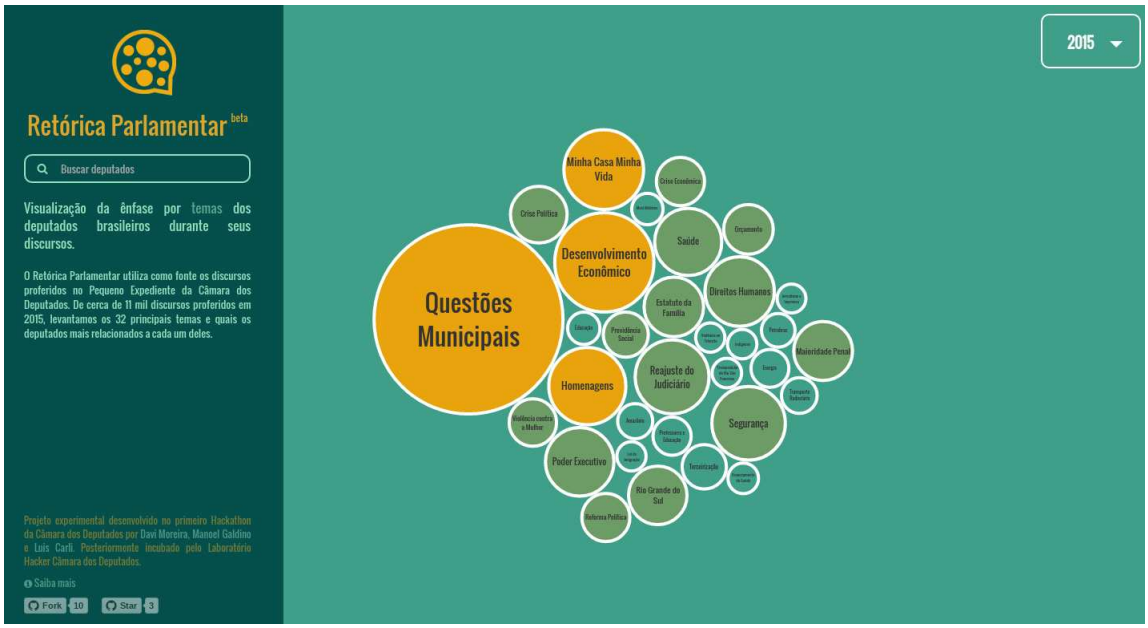


Figura 7 – Retórica Parlamentar

3.2 Planejamento das Atividades

Para a realização desse trabalho, foram identificadas algumas atividades que deveriam seguir uma ordem para ser executadas, como mostra o diagrama a seguir:

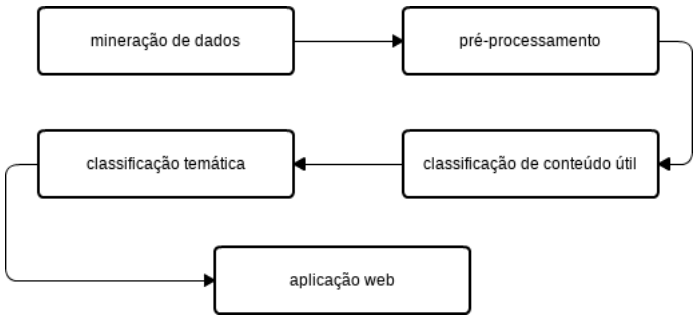


Figura 8 – Diagrama de planejamento

3.2.1 Mineração de Dados

A obtenção e persistência dos dados será realizada por uma biblioteca desenvolvida pelo autor, descrita na seção 4.1.

A obtenção dos dados consiste em realizar consultas ao *webservice* da Câmara dos Deputados, fazendo um processamento inicial, afim de padronizar as informações e transformá-las em seus tipos correspondentes na linguagem *python*. Os dados provenientes do *webservice* estão em formato *XML* e armazenam valores como *strings*.

Nesta etapa transformamos todos os campos com valores de números inteiros e decimais, datas, horários e textos em, respectivamente, valores dos tipos *int*, *float*, *date*-

time.date, *datetime.time* e *str*.

3.2.2 Pré-processamento

Apesar do "pré-processamento" não depender tanto dos dados reais, a definição das *stop words* deve ser feita levando em consideração o conteúdo que será analisado.

A etapa de pré-processamento consiste na análise dos textos obtidos na mineração de dados, afim de identificar as *stop words* presentes em textos do contexto legislativo. Além disso, deve ser realizada o processo de *stemização* para reduzir a dimensionalidade das *bag-of-words* geradas.

Estudamos diferentes estratégias para a utilização de n-gramas, mas por enquanto decidiu-se, por uma questão de simplicidade, limitar-se apenas ao uso de unigramas.

3.2.3 Classificação Conteúdo Útil

Essa classificação tem como objetivo melhorar a qualidade da análise temática dos deputados. Uma parte considerável dos textos não possuem valor semântico significativo, pois tratam de questões protocolares e de trâmite legislativo. Citamos um exemplo: *"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."*. Portanto, a classificação entre conteúdo útil/não-útil consiste em separar os parágrafos que realmente possuem valor para a análise posterior dos que não devem ser usados nestas análises.

3.2.4 Classificação Temática

Após determinar quais parágrafos serão análisados, os mesmos devem ser classificados de acordo com alguns temas. Selecionamos inicialmente: Agropecuária, Saúde, Esporte, Educação, Ciência e Tecnologia, Economia, Política, Meio Ambiente, Direitos Humanos e Segurança. Para a realização dessa tarefa, foi necessário construir um texto inicial para cada um dos temas listados, afim de fornecer um parâmetro inicial ao classificador. Os textos iniciais de cada tema têm como base textos previamente classificados em portais de notícias brasileiros e consistem em apenas uma listagem de palavras comuns relacionadas a estes temas.

3.2.5 Aplicação *Web*

Os dados obtidos da classificação temática serão utilizados para alimentar um sistema *web* para a exibição dos mesmos. As principais funcionalidades planejadas são: visualizar os temas mais abordados por deputados, organizados por região, partido e ban-

cada, visualizar todos os temas de uma determinada categoria, bem como o quanto cada tema é discutido e visualizar todos os temas abordados por um determinado deputado, mostrando separadamente os temas abordados em seus discursos e nas suas proposições.

Esta parte do projeto está em fase inicial e não iniciou o desenvolvimento propriamente dito. As funcionalidades planejadas e protótipos iniciais estão disponíveis no apêndice A.

3.3 Ferramentas e Tecnologias

Para auxiliar no desenvolvimento, foram escolhidas as seguintes tecnologias:

3.3.1 Linguagem de Programação

Devido ao tamanho da comunidade, grande utilização na área de aprendizado de máquina, possibilidade de desenvolvimento *web* e conhecimento prévio do autor e orientador, decidiu-se utilizar a linguagem *Python*[5] para o desenvolvimento das aplicações do presente trabalho.

3.3.2 *Frameworks* e Bibliotecas

O desenvolvimento da aplicação *web*, utilizará o *framework Django*[6], 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.

Para trabalhar com o processamento de linguagem natural e aprendizado de máquina aplicado, serão utilizadas as bibliotecas *plagiarism*[7] e *texblob*[8], que encapsula a biblioteca *NLTK*[9]. Alguns cálculos são implementados utilizando o *stack* científico do *Python*, que inclui o *numpy*[10], *scipy*[11], *matplotlib*[12] e *sklearn*[13].

[5] https://www.python.org
[6] https://www.djangoproject.com
[7] https://github.com/fabiommendes/plagiarism
[8] https://textblob.readthedocs.io
[9] http://www.nltk.org
[10] http://www.numpy.org
[11] https://www.scipy.org
[12] http://matplotlib.org
[13] http://scikit-learn.org

3.3.3 Gerenciador de Repositórios de Código

O gerenciamento de versões dos códigos das aplicações desenvolvidas utiliza o *Git*[14] e o serviço de *web hosting* compartilhado *GitHub*[15]. Além disso, os pacotes *Python* são enviados para o *PyPI*[16], sendo facilmente instaláveis através do comando *pip*[17].

3.3.4 Documentação do Projeto

A documentação do projeto, realizada utilizando a biblioteca *Python sphinx*[18], que possibilita a extração de documentação por meio de *docstrings* no código e arquivos *.rst*, e o próprio *GitHub*, através de *readmes* no repositório.

3.3.5 Gerenciamento de Tarefas

O controle de tarefas executadas ou em execução é feito utilizando o sistema de *issues* e quadro de projetos do *GitHub*.

[14] https://git-scm.com
[15] https://github.com
[16] https://pypi.python.org/pypi
[17] https://pip.pypa.io
[18] http://www.sphinx-doc.org

4 Resultados Obtidos

4.1 Obtenção dos Dados

Todos os dados utilizados para análise foram obtidos através do portal de dados abertos da Câmara dos Deputados[1], 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.

4.1.1 *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. Como os dados são fornecidos utilizando o formato *XML*, eles não são "tipados", ou seja, independente do tipo (inteiro, data, texto, etc) eles são representados com *strings*. Alguns dados são ambíguos, como os referentes aos deputados, onde existem "ideCadastro" e "idParlamentar", que são utilizados como parâmetros de entrada de requisições distintas. Outro problema é que requisições comuns precisam ser feitas indiretamente pois não agrega conteúdos com *queries* relacionais, como normalmente são as *API REST*. Além disso, o inteiro teor dos discursos parlamentares estão disponíveis apenas em formato *RTF*, o que dificulta um pouco a utilização dos mesmos.

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[2], 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* sendo que alguns são apenas imagens *scaneadas* dos documentos físicos.

[1] http://www.camara.leg.br/transparencia/dados-abertos
[2] https://dadosabertos.camara.leg.br/

A estrutura do *webservice* da Câmara dos Deputados pode ser encontrada no apêndice B

4.2 Proposta de Desenvolvimento

4.2.1 *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 9), com algumas alterações. Além disso, todos o código implementado foi escrito em inglês, apesar dos dados estarem em português.

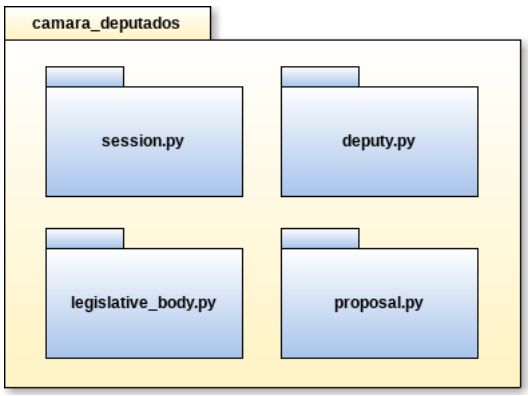


Figura 9 – Estrutura do módulo de consumo de dados da Câmara dos Deputados

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*. Esses *apps* possuem apenas as *models* (na linguagem da arquitetura *MVT* do *Django*) 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 10. Podemos notar que todas as colunas de todas as tabelas se encontram em em inglês, por motivos de padronização do código.

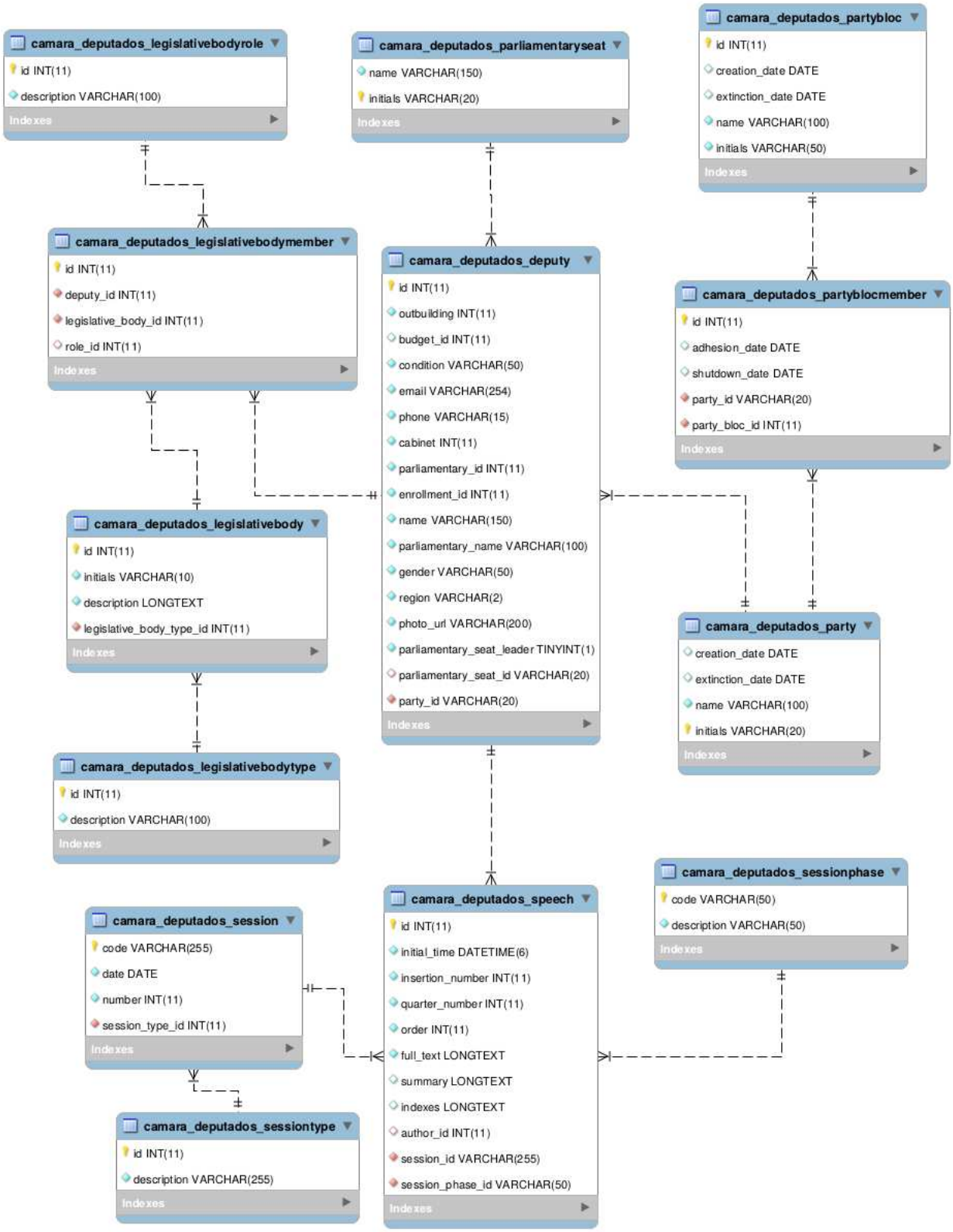


Figura 10 – Modelo entidade-relacionamento do banco de dados utilizado

4.2.2 Tenho Dito

"Tenho Dito" corresponde ao produto deste trabalho visível pelo usuário final. Trata-se de uma aplicação *web*, desenvolvida utilizando a linguagem *python* com o *framework Django*, e tem como objetivo ser uma forma mais lúdica e amigável de visualização de alguns 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, a análise dos textos será realizada com o apoio das ferramentas:

- **Plagiarism:** biblioteca em estado *alpha* 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). Apesar do foco ser a detecção de plágio em textos e códigos, as funcionalidades implementadas podem ser utilizadas em tarefas genéricas de PLN.

- **Textblob:** biblioteca *python* para processamento de dados textuais. Ela fornece uma interface simples para realizar tarefas comuns de processamento de linguagem natural, como análise de sentimento e classificação, por exemplo. Utiliza a biblioteca *NLTK*[3] para realizar essas tarefas.

4.2.2.1 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 "conteúdo útil" ou "conteúdo não-útil". Por exemplo, o trecho *"Muito obrigado, nobre Deputado. Pelo PSOL de São Paulo, o nobre Líder Ivan Valente. V. Ex^a tem cinco minutos na tri-*

[3] http://www.nltk.org

buna." 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 "conteúdo não-útil" e descartados da análise temática. A segunda etapa do processamento é a classificação temática dos parágrafos classificados como "conteúdo útil", 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 "conteúdo não-útil" e outro para "conteúdo". Os conjuntos de palavras podem ser encontrados no apêndice C.

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 a aplicação sugere uma classe mais provável e um especialista humano 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.

Vale observar que as classificações útil/não-útil sugeridas após a primeira fase normalmente correspondiam às corretas, ainda que isto não tenha sido medido explicitamente.

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 específico para cada tema. Os temas e seus respectivos conjuntos de palavras também se encontram no apêndice C.

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. Após esta etapa, acontece o treinamento supervisionado, onde um humano 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. Por isso, é utilizada uma ferramenta de *cache*, possibilitando o armazenamento dos classificadores depois de cada atualização. Toda vez que for realizado um treinamento ou uma classificação será utilizado o último classificador armazenado em

cache, com o treinamento prévio.

Esta fase mostrou que alguns discursos são difíceis de classificar ou por terem um conteúdo fragmentado (ex.: parágrafo com apenas uma ou poucas palavras, como *"-Rio de Janeiro"*) ou por conter um conteúdo que aborda mais de um tema simultaneamente (ex.: *"Eu parei de jogar há quase 17 anos e há 15 anos eu criei o Instituto Esporte & Educação - IEE, do qual sou Presidente, que trabalha com esporte e educação. E há 15 anos nós viajamos para cidades do Brasil que não têm acesso à prática motora na escola, que não têm estrutura, que não têm professores e, especialmente, que não têm a visão da educação física, do esporte, do movimento, da ação motora, da atividade motora como um fator de desenvolvimento, cuja presença é importante dentro da escola."*). Espera-se investigar modelos como o *Latent Dirichlet Allocation* que trata cada trecho como uma mistura de temas e não como pertencente a uma única categoria.

Esta fase inicial de classificação dos parágrafos do discurso será utilizada como subsídio para a classificação final de cada deputado. Cada parágrafo de discurso pode ser tratado como se fosse apenas a classe correspondete e a partir daí trataríamos a contagem das frequências temáticas como se fosse uma análise de *bag-of-words* ordinária. Esta etapa ainda não foi iniciada e será implementada no TCC2.

5 Considerações Finais

Durante o desenvolvimento desse trabalho, alguns resultados foram obtidos. Primeiramente, o autor estabeleceu um bom conhecimento sobre processamento de linguagem natural, incluindo técnicas de pré-processamento e aprendizado de máquina, o que possibilitou a aplicação desses conceitos no desenvolvimento das aplicações propostas nesse trabalho.

Para as contribuições tecnológicas, foi construída uma versão inicial da biblioteca *pygov-br*, que facilita o consumo do *webservice* da Câmara dos Deputados em aplicações *Python*, assim como a persistência dessas informações em banco dados, através da aplicação *Django* que faz parte da *pygov-br*. Iniciou-se o desenvolvimento da aplicação *web*, com a implementação dos algoritmos de classificação de conteúdo útil/não-útil e classificação temática. Entretanto, para a parte *web* propriamente dita, responsável pela visualização dos dados processados, foram apenas desenvolvidos os protótipos iniciais das telas do sistema.

5.1 Perspectivas Futuras

Para melhorar a classificação de parágrafos que abordam mais de um tema simultaneamente, investigaremos o modelo *Latent Dirichlet Allocation*, já que o mesmo considera que texto é gerado por uma mistura de temas, invés de pertencer a uma única categoria.

Além dos discursos, a análise temática dos deputados será realizada com os textos das proposições, que serão obtidos através da nova API de dados abertos da Câmara dos Deputados. Segundo o departamento de tecnologia da Casa, a disponibilização dos dados de proposições deverá ocorrer até março de 2017. Com isso, a biblioteca *pygov-br* também deverá ser atualizada para a utilização da nova API.

O desenvolvimento da aplicação “Tenho Dito” também deverá ser realizada e implantada, permitindo que toda a sociedade tenha acesso aos dados obtidos ao final das análises realizadas nesse trabalho, de forma lúdica e amigável.

Para fins de auditoria do processamento, será realizada a validação cruzada dos algoritmos de aprendizagem de máquina.

5.2 Cronograma

Tendo em vista as perspectivas futuras, o cronograma apresenta as atividades a serem realizadas e suas respectivas datas.

Atividade	**Março**	**Abril**	**Maio**	**Junho**
Utilizar n-gramas	X			
Integração do modelo LDA da sklearn	X	X		
Atualização da *pygov-br*		X		
Implementação do Tenho Dito		X	X	
Validação			X	
Escrita do trabalho		X	X	X

Tabela 2 – Cronograma TCC2

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 31.

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 29.

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 35.

GUDGIN, M. et al. *W3C SOAP Specifications*. 2007. Disponível em: <https://www.w3.org/TR/soap12/>. Citado na página 41.

HAND, D. J.; YU, K. Idiot's bayes — not so stupid after all? *International Statistical Review*, 2001. Citado na página 27.

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 29.

JAYNES, E. T. *PROBABILITY THEORY THE LOGIC OF SCIENCE*. [S.l.]: Cambridge University Press, 2003. 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 29.

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 2 vezes nas páginas 32 e 33.

LLOYD, S. P. Least square quantization in pcm. *IEEE Transactions on Information Theory*, p. 129–137, 1957. Citado na página 30.

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 26.

MOHRI, M.; ROSTAMIZADEH, A.; TALWALKAR, A. Foundations of machine learning. *The MIT Press*, 2012. Citado na página 26.

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 27.

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 27.

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 27.

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 26.

RUSSEL, S.; NORVIG, P. *Artificial Intelligence: A Modern Approach*. [S.l.: s.n.], 2003. Citado na página 26.

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

Apêndices

APÊNDICE A – Protótipos iniciais do Tenho Dito

Foram desenvolvidos alguns protótipos de telas do sistema Tenho Dito, a ser desenvolvido nesse trabalho. Na tela inicial (figura 11) será exibido um mapa político do Brasil e ao passar o *mouse* pelos estados, o tema mais abordado pelos parlamentares que representam o estado é mostrado. Também existe a possibilidade de alterar a forma de visualização, além de ter uma abordagem por estado, o usuário pode escolher por partido ou por tema. Entretanto, a abordagem por tema não foi prototipada.

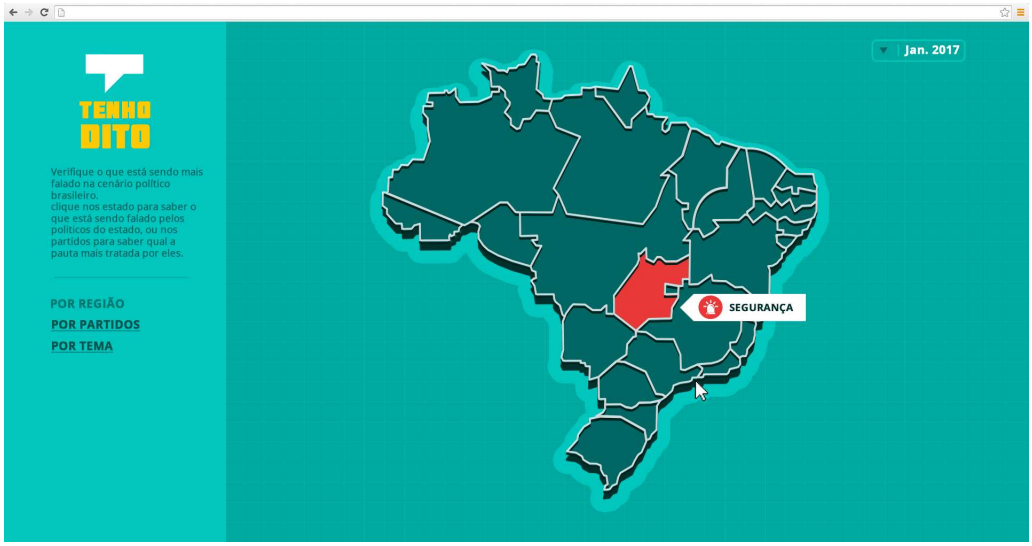


Figura 11 – Tela inicial do Tenho Dito - Visualização por região

Seguindo a abordagem por temas, ao clicar em um estado, o usuário é direcionado a outra página (figura 12), onde encontra um gráfico de bolhas, detalhando os temas abordados pelos parlamentares do estado. Quanto maior a bolha, mais o tema foi abordado. Além disso, também são listados todos os deputados que representam aquele estado, juntamente com sua foto, partido e o tema predominante em seus discursos e proposições. Ainda não foram definidas as interações com o gráfico de bolhas.

O usuário também poderá selecionar um deputado específico e visualizar o seu perfil. Na tela de perfil do deputado (figura 13), são exibidas as informações do deputado e também a quantidade de proposições e discursos analisados. Logo abaixo, será mostrado, dinâmica e randomicamente, trechos de discursos ou proposições e sua classificação. Além disso, serão listados todos os temas e a quantidade de discursos e proposições (por meio de gráfico de barras), com o objetivo de realizar uma comparação entre o que é mais dito pelo deputado e o que é mais proposto.

Figura 12 – Visualização detalhada dos temas, por região

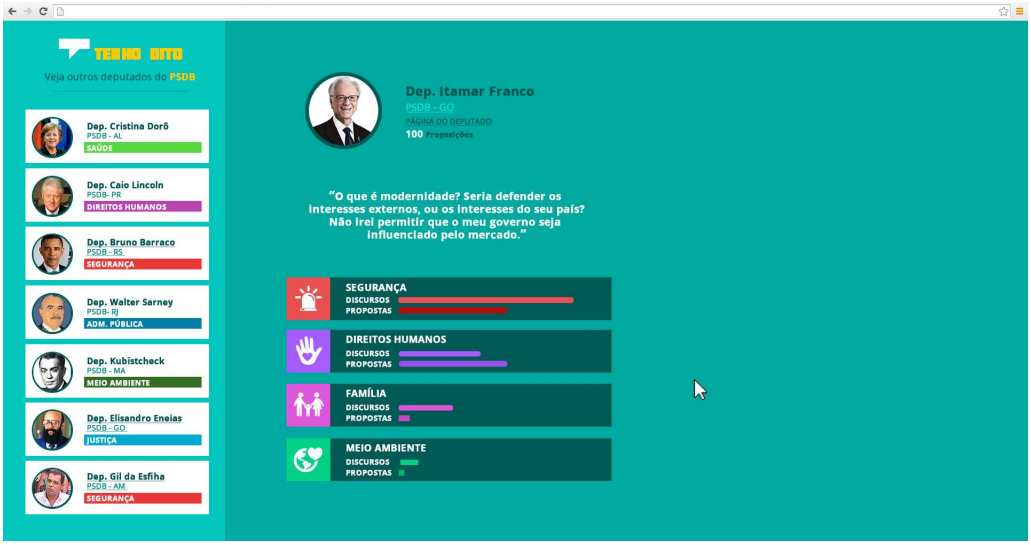


Figura 13 – Página de perfil do deputado

Quando o usuário clicar na opção de visualização por partidos, será exibida uma lista com os atuais partidos com representação na Câmara dos Deputados (figura 14). O sitema possibilitará três tipos de ordenação: por tamanho (quantidade de deputados por partido), por ordem alfabética ou por tema. Nessa tela, também serão exibidos os temas mais abordados pelos partidos, através dos seus membros. Caso o partido tenho mais deputados cujo tema mais abordado em seus discursos e proposições é "segurança", por exemplo, o tema atribuído ao partido será "segurança".

O usuário poderá, assim como na abordagem por estado, escolher um partido para detalhar os temas abordados e, da mesma forma, é exibido um gráfico de bolhas com os temas abordados pelos deputados desse partido. Ao lado são mostrados todos os deputados do partido, independente do seu estado, ao clicar em algum deles o usuário é direcionado à pagina de perfil dele.

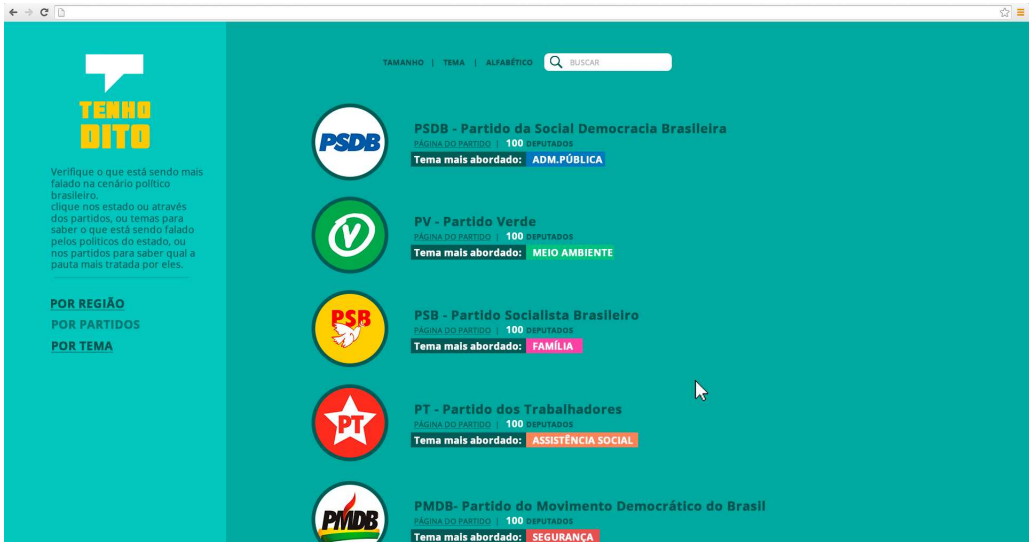


Figura 14 – Listagem dos partidos com representação na Câmara dos Deputados

Figura 15 – Visualização detalhada dos temas, por partido

APÊNDICE B – *Webservice* da Câmara dos Deputados

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 descrevemos os *endpoints* utilizados.

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)
- **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.

APÊNDICE C – Treinamento Inicial dos Classificadores

Para realizar o treinamento inicial dos classificadores *naive* Bayes, é necessário fornecer um texto inicial e a sua classificação. Esse apêndice descreve os textos usados nesse trabalho para cada classificação.

C.1 Classificação de Conteúdo Útil/Não-útil

Para a classificação de "conteúdo útil" e "conteúdo não-útil", foram utilizadas os seguintes conjuntos de palavras iniciais:

- **Conteúdo não-útil:** "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 útil:** "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"

C.2 Classificação Temática

Para a classificação temática, os temas escolhidos e seus respectivos conjuntos de palavras utilizados foram:

- **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 transmissão 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"