

**PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS**  
**NÚCLEO DE EDUCAÇÃO A DISTÂNCIA**  
**Pós-graduação *Lato Sensu* em Ciência de Dados e Big Data**

**GILLES VILLENEUVE MORAIS AMARO**

**ANÁLISE DAS ELEIÇÕES BRASILEIRAS PARA PRESIDENTE ATRAVÉS DO**  
**AGRUPAMENTO E ANÁLISE DE SENTIMENTOS**

Belo Horizonte

2022

**GILLES VILLENEUVE MORAIS AMARO**

**ANÁLISE DAS ELEIÇÕES BRASILEIRAS PARA PRESIDENTE ATRAVÉS DO  
AGRUPAMENTO E ANÁLISE DE SENTIMENTOS**

Trabalho de Conclusão de Curso apresentado  
ao Curso de Especialização em Ciência de  
Dados e Big Data como requisito parcial à  
obtenção do título de especialista.

Belo Horizonte

2022

## SUMÁRIO

<b>1. Introdução .....</b>	<b>4</b>
<b>1.1. Contextualização .....</b>	<b>4</b>
<b>1.2. O problema proposto .....</b>	<b>5</b>
<b>1.3. Objetivos .....</b>	<b>6</b>
<b>2. Coleta de dados .....</b>	<b>8</b>
<b>3. Processamento/Tratamento de Dados.....</b>	<b>11</b>
<b>4. Análise e Exploração dos Dados .....</b>	<b>15</b>
<b>5. Criação de Modelos de Machine Learning.....</b>	<b>20</b>
<b>6. Interpretação dos Resultados.....</b>	<b>31</b>
<b>7. Apresentação dos Resultados .....</b>	<b>34</b>
<b>8. Links .....</b>	<b>36</b>
<b>REFERÊNCIAS.....</b>	<b>37</b>
<b>APÊNDICE .....</b>	<b>38</b>

## 1. Introdução

Redes sociais como Twitter, Facebook, Instagram e também os antigos Orkut, Myspace e Google+ mudaram o comportamento das pessoas encorajando-as a expressarem seus sentimentos, pensamentos, opiniões e detalhes de sua rotina através de mensagens que podem ser facilmente enviadas através de *smartphones* ou via web, de acordo com Klinczak (2016).

Conforme Xiaohui (2013), essas mensagens desempenham um papel importante na vida de muitas pessoas fazendo com que as ferramentas mencionadas, bem como muitas outras que foram criadas posteriormente como o WhatsApp e Tik Tok, passassem a ter um importante papel na comunicação entre as pessoas e empresas.

### 1.1. Contextualização

A teoria de Goffman defende que o ambiente online encoraja mais ações espontâneas do que dados coletados através de pesquisas tradicionais, justamente pelos usuários estarem em um ambiente em que acreditam não estarem sendo monitorados fazendo com que as redes sociais se tornem um espaço em que pensamentos e comportamentos dos usuários sobre produtos e serviços possam ser monitorados pelas empresas trazendo informações valiosas.

O sucesso das redes sociais criou então um problema de difusão de informações em larga escala, isso porque a quantidade de informação compartilhada cresce a cada dia dificultando muitas vezes encontrar uma informação específica. Como exemplo disso podemos citar como o número de mensagens compartilhadas nas redes sociais aumentam consideravelmente quando ocorre algum grande evento, como a Copa da Mundo, eleições ou desastres naturais.

Os autores Tumasjan et al. (2010) fizeram a análise de sentimentos tendo como foco as eleições alemãs. Para isso foram coletados 104.003 mensagens do Twitter publicadas entre 13 de agosto e 19 de setembro de 2009, obtidos considerando os nomes dos 6 partidos representados no parlamento alemão. O resultado foi de 70.000 mensagens mencionando os partidos e 35.000 referindo-se aos seus candidatos. Após isso foi feita a

tradução das mensagens para o inglês e foi utilizado um software terceiro chamado de LIWC para categorizar as mensagens em positivas, negativas ou neutras. Os autores consideraram que o número de mensagens positivas sobre um partido político poderiam ser consideradas como uma intenção de voto, podendo se aproximar do resultado das pesquisas eleitorais tradicionais.

Na Copa do Mundo, jogo entre Brasil e Alemanha, foram computados 35,6 milhões de mensagens apenas na rede social Twitter, similar a isso temos a cerimônia do Oscar em 2014 com 3,3 bilhões de mensagens na mesma rede, entre outros eventos, como as eleições brasileiras, a pandemia causada pelo corona vírus e os protestos referentes as vacinas.

Alado as redes sociais, temos ainda a divulgação de notícias que podem ser encaminhadas para grupos de amigos que irão encaminhar a mesma para seus seguidores, até que essa tenham um alcance viral, fazendo com que o número de pessoas alcançadas cresça exponencialmente, oferecendo também uma oportunidade única de análise de propagação de informações.

## **1.2. O problema proposto**

As redes sociais permitem, não apenas que os usuários falem expondo suas opiniões e pensamentos sobre assuntos diversos mas, que também compartilhem notícias, o que influencia na formação e opinião de outras pessoas.

Com base nisso foram obtidos 3 conjuntos de dados da rede social Twitter, filtrados com base nas seguintes palavras: “eleições2022”, “bolsonaro”, “sergiomoro”. A escolha de termos deu-se devido a ainda não ter lançado os candidatos oficiais para as próximas eleições, e que muitos portais da mídia estarem considerando o ex juiz Sergio Moro como uma opção conhecida como “Terceira via”, sendo uma opção “neutra” para os eleitores que não se identificam com o ex-presidente Lula e o atual presidente Bolsonaro. Para cada conjunto foram coletadas no máximo 500 mensagens, totalizando 1.389 mensagens extraídas do Twitter, essa variação na quantidade de mensagens obtidas é devido a uma limitação da API na extração dos dados em sua versão gratuita.

De forma a complementar a base de dados, fizemos a extração manual de dados da rede social Facebook considerando os mesmos termos e limitados a 50 mensagens de cada um. Tal escolha de mídia complementar foi feita no intuito de, possivelmente, trazer diferentes opiniões sobre os mesmos assuntos, visto que não necessariamente um mesmo usuário que interaja em uma plataforma faça o mesmo em outra. A diferença de quantidade de dados está relacionada a limitações de hardware para rodar a aplicação de forma que procuramos ficar com cerca de mil mensagens para fazer o processamento e análises propostas.

A princípio analisamos somente mensagens em português, sendo desconsideradas mensagens em outros idiomas, e a coleta foi restrita as últimas mensagens dos últimos 3 meses, obtidas em ordem cronológica.

Exploramos as mensagens de texto através da classificação, e aos dados dessa classificação aplicamos a análise de sentimentos, desconsiderando outras informações como links, imagens, dados de usuários. Seu objetivo foi trazer uma visão prévia do sentimento da população sobre as eleições, apesar da pequena base de dados quando comparada com a quantidade de mensagens circulando na rede.

Esses dados poderão ser analisados por diversas pessoas, como profissionais de marketing de alguma empresa em específico ou o próprio governo para saber sobre a aceitação e alguma medida ou lei, entre outros. Assim, pode-se ter uma visão geral do que a comunidade está compartilhando sobre notícias e o que acha a respeito, o que pode ajudar na tomada de decisão como, por exemplo, um candidato ao ver que sua opinião frente aos eleitores não está boa, podendo trabalhar para melhorar sua imagem.

### **1.3. Objetivos**

Temos então como objetivo geral desse trabalho realizar a análise de dados extraídos do Twitter e do Facebook de acordo com os termos pré selecionados de forma a classificar as informações em conjuntos conforme sua similaridade, e então aplicar a análise de

sentimentos para verificar o sentimento geral da população com relação as eleições brasileiras de 2022. Para isso, temos os seguintes objetivos específicos:

- Realizar a coleta de mensagens da rede social Twitter e Facebook considerando-se as seguintes palavras-chave: “eleições2022”, “bolsonaro”, “sergiomoro”;
- Realizar o pré-processamento das mensagens, excluindo mensagens em branco e em outros idiomas;
- Fazer a unificação das bases;
- Aplicar algoritmos de classificação como k-means, k-medoids e DBSCAN;
- Fazer a análise de sentimentos para cada conjunto identificado;
- E por fim, gerar uma nuvem de palavras para cada um dos algoritmos obtidos, considerando o melhor resultado com base no número de termos em cada conjunto e na distância entre eles.

## 2. Coleta de Dados

Os dados foram extraídos de 2 bases de dados, a primeira foi feita através da API do Twitter e da biblioteca “twitterR” da linguagem R, considerando 3 termos de busca, limitando-se ao número máximo de 500 resultados de cada um: “eleições2022”, “bolsonaro”, “sergiomoro”. Processo feito no dia 24 de Janeiro de 2022, totalizando 1.389 dados.

Na segunda foi feita uma extração manual da rede social Facebook nessa mesma data, considerando-se os mesmos termos, porém limitado a 50 mensagens de cada, e em ordem cronológica de exibição e considerando-se apenas os dados textuais, totalizando 150 mensagens.

No geral optamos por uma quantidade de mensagens pequena de forma a otimizar o processamento em computadores domésticos e também considerando-se que alguns dados poderiam ser perdidos durante seu tratamento, sendo o objetivo ficar com cerca de mil dados após o pré-processamento.

Na Tabela 1 apresentamos todos os dados que foram obtidos ao realizar a coleta de dados do Twitter, e na Tabela 2 apresentaremos somente os termos que serão utilizados, sendo que os demais serão descartados. Na Tabela 3 constam os dados a serem extraídos manualmente do Facebook, e como essa extração será manual, serão coletadas somente as informações necessárias a pesquisa.

Nome da coluna/campo	Descrição	Tipo
text	Mensagem textual podendo conter texto, links, emoticons, hashtags, números, entre outros	string
favorited	Contém o valor verdadeiro ou falso caso a mensagem tenha sido ou não favoritada	booleano
favoriteCount	Quantidade de vezes que a mensagem foi favoritada	inteiro
replyToSN	Nome do usuário que replicou a mensagem	string



created	Data e hora em que a mensagem foi criada	datetime
truncated	Caso a mensagem tenha passado da quantidade de caracteres permitida, indo para 280 caracteres, de acordo com as novas regras do Twitter	booleano
replyToSID	Contém o valor verdadeiro ou falso caso a mensagem tenha sido ou não compartilhada	booleano
Id	Identificador da mensagem	inteiro
replyToUID	Id do usuário que fez o compartilhamento	inteiro
statusSource	Agente de usuário de origem para este <i>tweet</i>	string
screenName	Nome do usuário que fez a postagem	string
retweetCount	Quantidade de vezes que a mensagem foi replicada para os seguidores	inteiro
isRetweet	Verdade caso a mensagem tenha sido compartilhada por outro usuário	booleano
retweeted	Contém o valor verdadeiro ou falso caso a mensagem tenha sido ou não compartilhada	booleano
longitude	Dados de longitude quando disponíveis	decimal
latitude	Dados de latitude quando disponíveis	decimal

Tabela 1: Dados fornecidos pela API do Twitter.

<b>Nome da coluna/campo</b>	<b>Descrição</b>	<b>Tipo</b>
text	Mensagem textual podendo conter texto, links, emoticons, hashtags, números, entre outros	string

Tabela 2: Dados a serem consideradas na base de dados 1 (extraída do Twitter).

<b>Nome da coluna/campo</b>	<b>Descrição</b>	<b>Tipo</b>
Text	Corpo do texto da postagem no Facebook	string

Tabela 3: Dados a serem consideradas na base de dados 2 (extraída do Facebook).

Após realizar a filtragem das colunas a serem utilizadas em cada base de dados, as mesmas foram unificadas como um único conjunto sendo pré processadas e os algoritmos aplicados a elas conforme objetivo do trabalho.

### 3. Processamento/Tratamento de Dados

O processamento, ou tratamento de dados compreende a aplicação de diversas técnicas para organização, filtro, padronização e preparação dos dados, conforme Klineczak (2016). Assim, faz-se necessário primeiramente identificar com quais dados trabalharemos, de forma a fazer o processamento somente neles, e não desnecessariamente em toda a base de dados, após isso é necessário observar os dados de forma geral, visualizando o que precisa ser feito, como por exemplo:

- Campos em branco, precisam ser excluídos ou não irão afetar o resultado final?
- Campos com formatação diferente, como por exemplo datas, precisam ser padronizados? Qual formato mais usual? Qual formato prevalece na base de dados?
- Em caso de dados textuais, serão retirados artigos ou palavras com menos de 2 ou 3 caracteres?
- Em caso de dados textuais, números e *emoticons* serão considerados?
- Existem dados duplicados? Precisam ser removidos para não prejudicar o resultado final?

Dessa forma, vemos que não existe uma receita de bolo no tratamento dos dados, sendo necessário fazer essa avaliação caso a caso e de acordo com a base de dados obtida. Em nosso caso, como iremos trabalhar com dados textuais, a primeira coisa a ser feita é desconsiderar o restante da base de dados, trabalhando apenas com a coluna “text” se tratando da base de dados de *tweets*, e em seguida fazendo a união desta com o campo único obtido manualmente da base de dados do Facebook.

Um ponto de destaque é que os dados de redes sociais, difere de outras bases, podem apresentar grandes ruídos, por não serem estruturados e dinâmicos, e no caso dos dados obtidos do Twitter, existe ainda a limitação de tamanho de mensagem, o que faz com

que muitos usuários utilizem abreviações e *emoticons* para se expressar, segundo Berry (2006).

Com base nisso, fizemos o processamento com base nas técnicas utilizadas na área de Recuperação de Informações, de acordo com Baeza-Yates e Ribeiro-Neto (1999), executando primeiramente os passos clássicos de tratamento de dados textuais, como:

- Identificação da unidade textual base, no nosso caso, a coluna “text”, que será a única a ser utilizada e para ASCII;

```
> tweets <- sapply(conjuntoTotal$text,function(row) iconv(row, "latin1", "ASCII", sub
=""))
```

- Padronização de todas as letras para minúsculas

```
> myCorpus <- VCorpus(VectorSource(tweets))
> myCorpus2 <- tm_map(myCorpus, tolower)
```

- Remoção de palavras e termos específicos do Twitter, como “RT”, “rt” e “@” (o RT significa que uma mensagem foi duplicada para outros seguidores, com conteúdo textual adicional ou não, e o @ envia uma notificação para uma pessoa em específico);

```
> removeRT <- function(x) gsub("(RT|via)((?:\\b\\W*@\\w+)+)", "", x)
> myCorpus3 <- tm_map(myCorpus2, removeRT)
> removert <- function(x) gsub("(rt|via)((?:\\b\\W*@\\w+)+)", "", x)
> myCorpus4 <- tm_map(myCorpus3, removert)
> removePeople <- function(x) gsub("@\\w+", "", x)
> myCorpus5 <- tm_map(myCorpus4, removePeople)
```

- Remoção de pontuação (*emoticons*) e números;

```
> myCorpus6 <- tm_map(myCorpus5, removePunctuation)
> myCorpus7 <- tm_map(myCorpus6, removeNumbers)
```

- Remoção de links dentro da mensagem;

```
> removeURL <- function(x) gsub("http[[:alnum:]]*", "", x)
> myCorpus8 <- tm_map(myCorpus7, removeURL)
```

- Remoção das palavras chave utilizadas na obtenção da base de dados do texto, visto que a maior parte das mensagens irá conter essas palavras, o que pode influenciar no resultado final;

```
> myStopwords <- c(stopwords(kind="pt"), "eleições2022", "bolsonaro", "sergiomoro")
> myCorpus9 <- tm_map(myCorpus8, removeWords, myStopwords)
```

- Remoção de links que possam estar na mensagem via código fonte;

```
> myStopwords2 <- c(stopwords(kind="pt"), "href")
> myCorpus10 <- tm_map(myCorpus9, removeWords, myStopwords2)
```

- Remoção de espaços em branco como espaçamento duplo;

```
> myCorpus11 <- tm_map(myCorpus10, stripWhitespace)
```

- Conversão das mensagens para texto simples, ou seja, sem codificações;

```
> myCorpus12 <- tm_map(myCorpus11, PlainTextDocument)
```

- E por fim, aplicação da técnica *stemming*, um procedimento que conecta elementos textuais de semântica parecida, feito através da obtenção do radical das palavras, fazendo com que sufixos e prefixos sejam eliminados e variações como plurais sejam reduzidas a uma forma única.

```
> myCorpus13 <- tm_map(myCorpus12, stemDocument)
```

Com base nisso, iniciamos o tratamento com uma base de dados inicial de 1.389 dados obtidos do Twitter e 150 obtidos do Facebook, totalizando um conjunto de dados de 1.539 mensagens. As mensagens do Twitter foram obtidas via API, e devido a isso é necessário que sejam convertidas como data frame, já a do Facebook, como foi obtida manualmente foi armazenada em uma planilha de texto, e ao fazer sua importação, a função

“*read\_excel*” já faz sua conversão para data frame. E então as bases de dados são unidas através da função “*rbind*” e executados os passos mencionados acima.

Como resultado final, podemos observar uma mensagem como exemplo, tendo os elementos textuais descritos retirados.

```
> inspect(myCorpus13[[1]])
<<PlainTextDocument>>
Metadata: 7
Content: chars: 67

brasil preocupado mudana mundo vendo saco lixo planeta melhor compr
```

Como não obtivemos nenhuma mensagem totalmente em idioma estrangeiro nem em caracteres estranhos ou somente com pontuação (*emoticons*), não tivemos que fazer o descarte de nenhum dado, e optamos também por não excluir mensagens repetidas, pois podem representar uma opinião de maior peso sobre algum candidato ou sobre a eleição, dessa maneira nossa base de dados final continua do mesmo tamanho da original, porém com o pré processamento realizado.

```
> myCorpus13
<<VCorpus>>
Metadata: corpus specific: 0, document level (indexed): 0
Content: documents: 1539
```

E como última etapa dessa fase fizemos a conversão do conjunto de dados em uma matriz de termos x frequência através da função *TermDocumentMatrix*, onde foram excluídas palavras com menos de 3 caracteres. Como resultado obtivemos 3.018 termos encontrados nas 1.539 mensagens iniciais.

```
> myTdmReal <- TermDocumentMatrix(myCorpus13,control=list(wordLengths=c(3,Inf)))

> myTdmReal
<<TermDocumentMatrix (terms: 3018, documents: 1539)>>
Non-/sparse entries: 12973/4631729
Sparsity : 100%
Maximal term length: 32
Weighting : term frequency (tf)
```

#### 4. Análise e Exploração dos Dados

Começamos por verificar quais eram os termos mais frequentes no conjunto de dados para ter uma ideia geral do que se falava sobre as eleições para 2022 e os possíveis candidatos. Com isso, fizemos um filtro com base na frequência das palavras, sendo maior que 5 na Figura 1, maior que 15 na Figura 2 e maior que 50 na Figura 3. E no Quadro 1 apresentamos um resumo das palavras mais frequentes e sua quantidade.

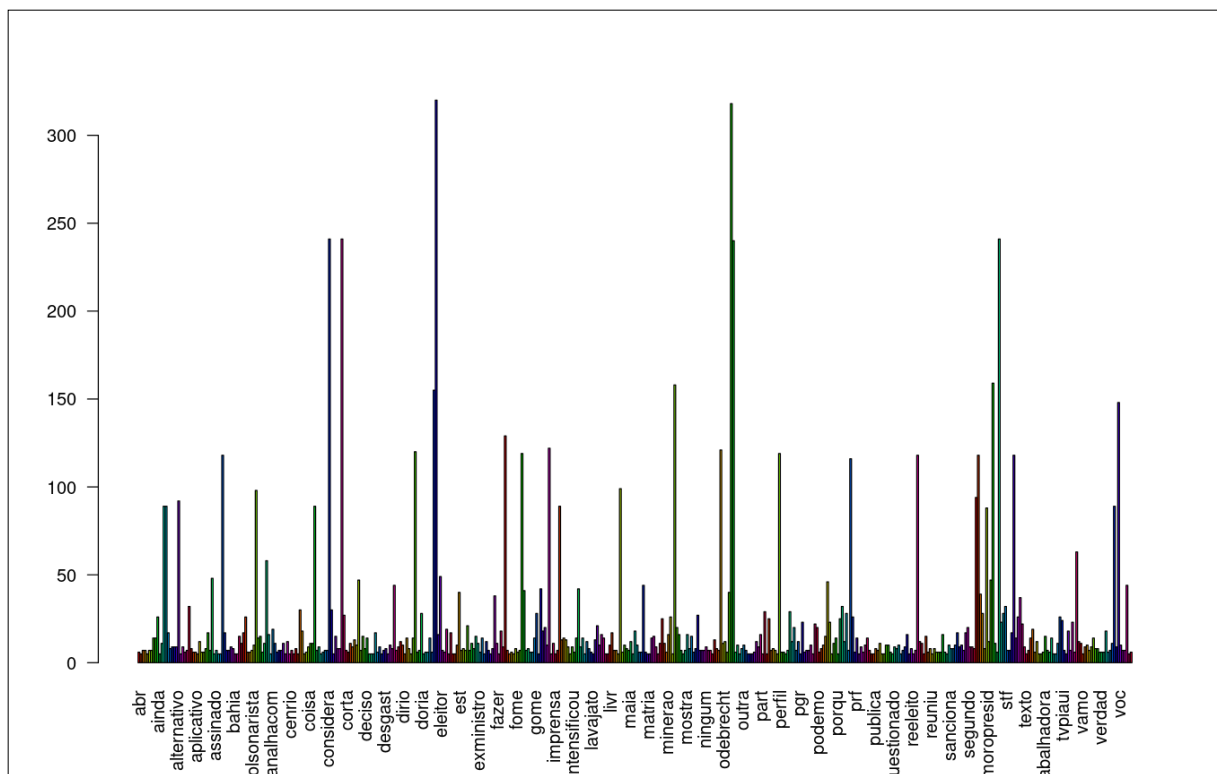


Figura 1: Termos mais encontrados no conjunto com frequência maior que 5.

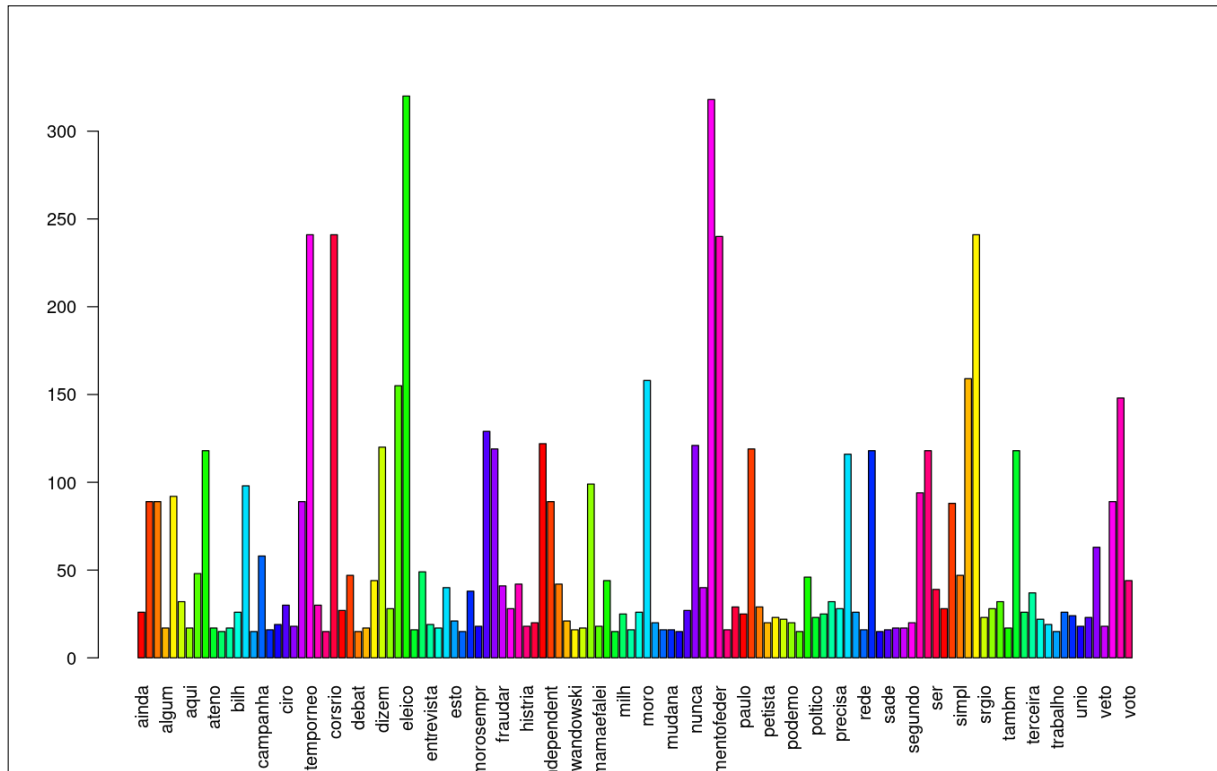


Figura 2: Termos mais encontrados no conjunto com frequência maior que 15.

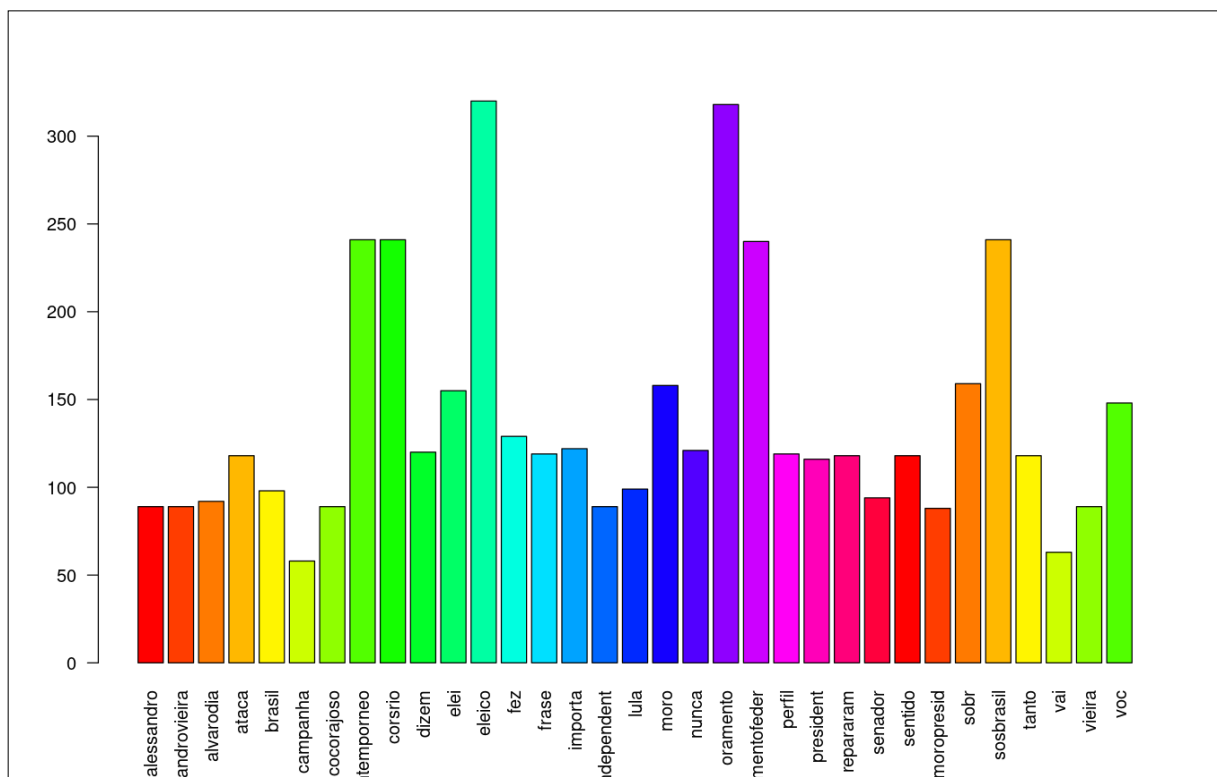


Figura 3: Termos mais encontrados no conjunto com frequência maior que 50.



```
> freq
```

ataca	contemporaneo	corsrio	dizem	elei	eleico
118	241	241	120	155	320
fez	frase	importa	moro	nunca	oramento
129	119	122	158	121	318
oramentofeder	perfil	president	repararam	sentido	sobr
240	119	116	118	118	159
sosbrasil	tanto	voc			
241	118	148			

Quadro 1: Resumo dos termos mais frequentes.

Com relação a nomes de possíveis candidatos, temos a frequência apresentada na Tabela 4.

Possível candidato	Termo	Quantidade de menções
Bolsonaro	Jairbolsonaro, euvotobolsonaro, jair	63
Lula	Lulapresident, lula	105
Sergio Moro	sergiomorodobrasil, sergiomoropresid, sergiomoropresident, srgiomoro, moropresident, morovencebolsolula, moro, fechadocomsergiomorosempr	355
Doria	doria	28

Tabela 4: Frequência de menção de termos envolvendo os possíveis candidatos a presidência.

Outra análise que fizemos foi a de identificação de termos como positivos ou negativos considerando todo o conjunto, utilizando as bibliotecas *dplyr* e *tidytext*. Usamos a função *tidy* e enviando a matriz termo x frequência obtido temos um conjunto onde cada linha contém informações sobre componentes importantes do modelo. Depois disso verificamos a classificação de cada termo.

```
> ap_sentiments
# A tibble: 119 × 4
  term      document count sentiment
<chr>    <chr>    <dbl> <chr>
1 grand   character(0)     1 positive
2 crime   character(0)     1 negative
3 favor   character(0)     1 positive
4 popular character(0)     1 positive
5 like    character(0)     1 positive
6 like    character(0)     1 positive
7 like    character(0)     1 positive
8 like    character(0)     1 positive
9 like    character(0)     1 positive
10 propaganda character(0) 1 negative
# ... with 109 more rows
```

Uma forma de aprofundar essa análise é encontrar os termos mais negativos ou positivos, através do uso de funções da biblioteca *tidyr*, conforme demonstrado no Quadro 2.

term	negative	positive	sentiment	term	negative	positive	sentiment
<chr>	<dbl>	<dbl>	<dbl>	<chr>	<dbl>	<dbl>	<dbl>
crime	47	0	-47	valor	0	12	-12
diss	14	0	-14	favor	0	8	-8
fake	3	0	-3	like	0	5	-5
vice	3	0	-3	grand	0	4	-4
insignificant	2	0	-2	popular	0	3	-3
limit	2	0	-2	excelent	0	2	-2
pretend	2	0	-2	important	0	2	-2
urgent	2	0	-2	super	0	2	-2
cruel	1	0	-1	convenient	0	1	-1
deficient	1	0	-1	decent	0	1	-1
. with 24 more rows				. with 24 more rows			

Quadro 2: Na esquerda os termos encontrados com maior peso negativo, e na direita os com maior peso positivo.

Um resumo desses termos é apresentado na Figura 4, onde podemos visualizar que a quantidade de termos considerados negativos é maior que os considerados positivos sendo, os termos negativos menores que 0, e os positivos os maiores que 0. Quando a pouca frequência com que cada termo aparece, acreditamos ser devido ao tamanho limitado do conjunto de dados utilizado.

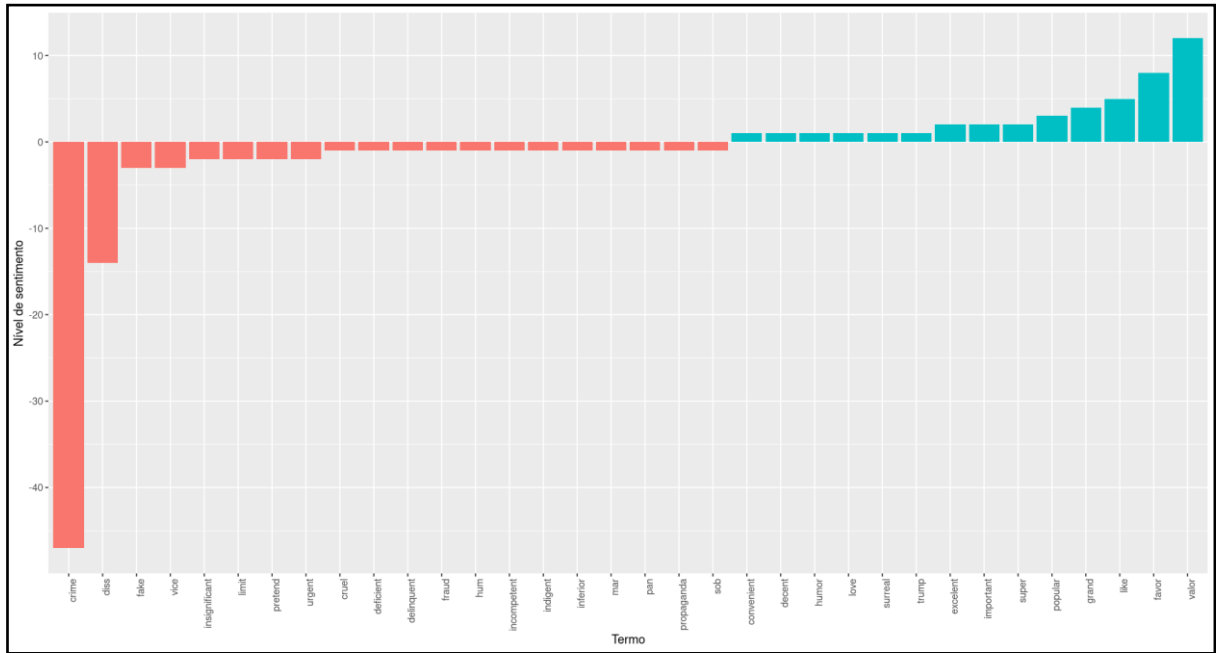


Figura 4: Gráfico comparativo entre os termos com maior peso positivo e negativo.

Finalmente, realizamos uma nuvem de palavras para os conjunto de termos negativos e positivos, conforme demonstrado no Quadro 3.



Quadro 3: Conjunto de dados de termos negativos na esquerda, e conjunto de termos positivos na direita.

## 5. Criação de Modelos de Machine Learning

Para criação dos modelos de *machine learning* utilizamos 3 conhecidos algoritmos de agrupamento, o k-means, k-medoids e DBSCAN. Os 3 algoritmos precisam que seja informado um número de conjuntos em que eles serão divididos.

De acordo com Han e Kamber (2001), o k-means é a opção mais conhecida de agrupamento, sendo que ele funciona da seguinte forma:

1. uma série de k pontos iniciais são gerados randomicamente;
2. esses pontos são considerados como centros dos agrupamentos (ou suas médias);
3. cada instância textual é usada como entrada e será associada ao agrupamento com o centro mais próximo;
4. o valor da média do agrupamento mais próximo é atualizada para ser considerada este novo elemento do agrupamento;
5. os passos (3) e (4) são repetidos até que não ocorram mais mudanças de rotulação nas instâncias.

Quanto ao k-medoids, ele é similar ao k-means, porém sua média de grupo é substituída por seu “medóide”, isto é, o ponto pertencente ao conjunto de dados existente mais central. Enquanto no centro de um agrupamento para o k-means do agrupamento está a média dos seus pontos, não necessariamente um valor pertencente ao conjunto de dados, no algoritmo k-medoids esses valores são associados a pontos de dados existentes, segundo Han e Kamber (2001).

Segundo Seif (2018), o DBSCAN é um algoritmo de agrupamento baseado em densidade, ele começa com um ponto de dados inicial randômico que ainda não tenha sido percorrido e sua vizinhança é extraída por uma distância *epsilon*  $\epsilon$  havendo um número mínimo de pontos (conforme o parâmetro minPoints) dentro da vizinhança começando o processo de agrupamento se tornando o primeiro ponto do novo conjunto. Caso contrário esse ponto é marcado como ruído e mais tarde poderá se tornar parte de algum conjunto. Nos dois casos o ponto é marcado como “visitado”. Esse processo é repetido até que todos os conjuntos sejam criados, ou seja, todos os pontos sejam percorridos e rotulados.

Algumas vantagens desse algoritmo de acordo com Seif (2018) é a identificação de pontos fora da curva como ruídos agregando-os posteriormente ao conjunto mais próximo

trabalhando bem com conjuntos de diversos tamanhos. Porém, sua principal desvantagem é que ele trabalha baseado em densidade, o que não se adequa a todos os problemas e conjuntos de dados, principalmente quando a densidade é variável.

Para a análise de sentimentos trabalhamos com os dicionários Oplexicon e o SentiLex. Já com relação a métrica, costuma-se usar a distância entre os elementos internos e externos do conjunto, o ideal é que as distâncias internas sejam próximas representando a forma de ligação entre os elementos e as distâncias externas sejam maiores, demonstrando que os conjuntos estão longe entre si. Caso sejam muito próximas, significa que o elemento poderia pertencer a outro grupo caso o número de conjuntos fosse menor.

Quanto a validação cruzada, ela trabalha com algoritmos rotulados como pertencentes a algum determinado conjunto, coisa que não temos, dessa forma não a utilizamos.

Para determinar o número de conjuntos fizemos alguns testes com cada algoritmo, conforme demonstrado em suas respectivas tabelas, considerando o k como 3, 5, 7 e 9.

A Tabela 5 apresenta o resumo dos resultados obtidos para o k-means, onde verificamos que em todos os valores de k existe um grande conjunto e então conjuntos menores que costumam ser relacionados a assuntos paralelos em volta do tema principal.

<b>Número conjuntos</b>	<b>Tamanho dos conjuntos</b>	<b>Medidas intra-cluster</b>	<b>Distância média entre os conjuntos</b>
3	2969, 23, 26	9562, 1340, 2669	1428.84
5	8, 12, 2984, 8, 6	9.62, 91.42, 10601.86, 199.62, 173.83	3923
7	18, 8, 2830, 76, 66, 12, 8	296.9293, 199.6250, 7511.8950, 9.6250, 850.4500, 1722.5556, 1441.0610	4178
9	2564, 122, 75, 18, 18, 8, 8, 57, 148	6803.1463, 976.8852, 296.4533, 1722.5556, 444.6111, 199.6250, 9.6250, 511.5439, 859.9932	3175

Tabela 5: Resumo dos resultados do algoritmo k-means.

Escolhendo o  $k=7$  como melhor conjunto devido a suas medidas intra e entre conjuntos, apresentamos sua nuvem de palavras para os termos mais frequentes na Figura 5.

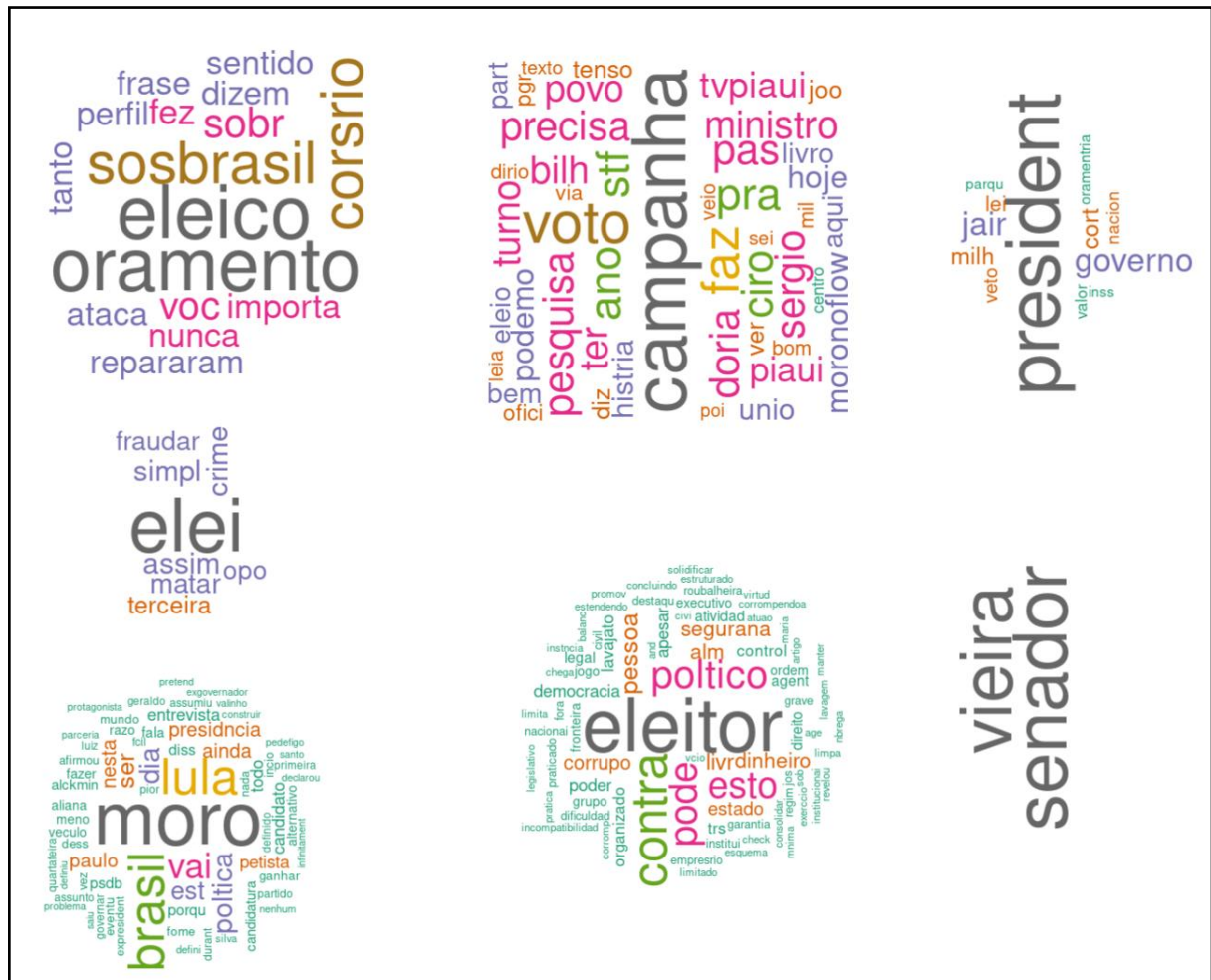
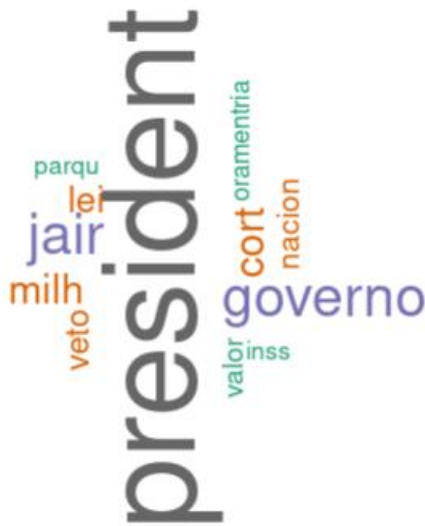



Figura 5: Nuvem de termos mais frequentes para o algoritmo k-means com  $k=7$ .

De forma comparativa apresentamos a separação dos conjuntos do algoritmo k-means com  $k=5$  na Figura 6, onde pode-se observar a distância entre os termos dentro do conjunto principal, demonstrando que eles poderiam ser separados em mais grupos, como fica visível na Figura 5. Com base nisso, para os próximos algoritmos apresentamos somente as nuvens de palavras para o melhor valor de  $k$ .

Utilizando os 3 conjuntos que apresentam nomes de candidatos (moro e lula), (doria e jair), aplicamos o algoritmo de análise de sentimentos de forma a verificar a forma como termos com relação a esses candidatos estão classificados onde obtivemos uma relação neutra para alguns termos, ou seja, não podem ser considerados nem positivos ou negativos, conforme Quadro 4.

Utilizando os 3 conjuntos que apresentam nomes de candidatos (moro e lula), (doria e jair), aplicamos o algoritmo de análise de sentimentos de forma a verificar a forma como termos com relação a esses candidatos estão classificados onde obtivemos uma relação neutra para alguns termos, ou seja, não podem ser considerados nem positivos ou negativos, conforme Quadro 4.

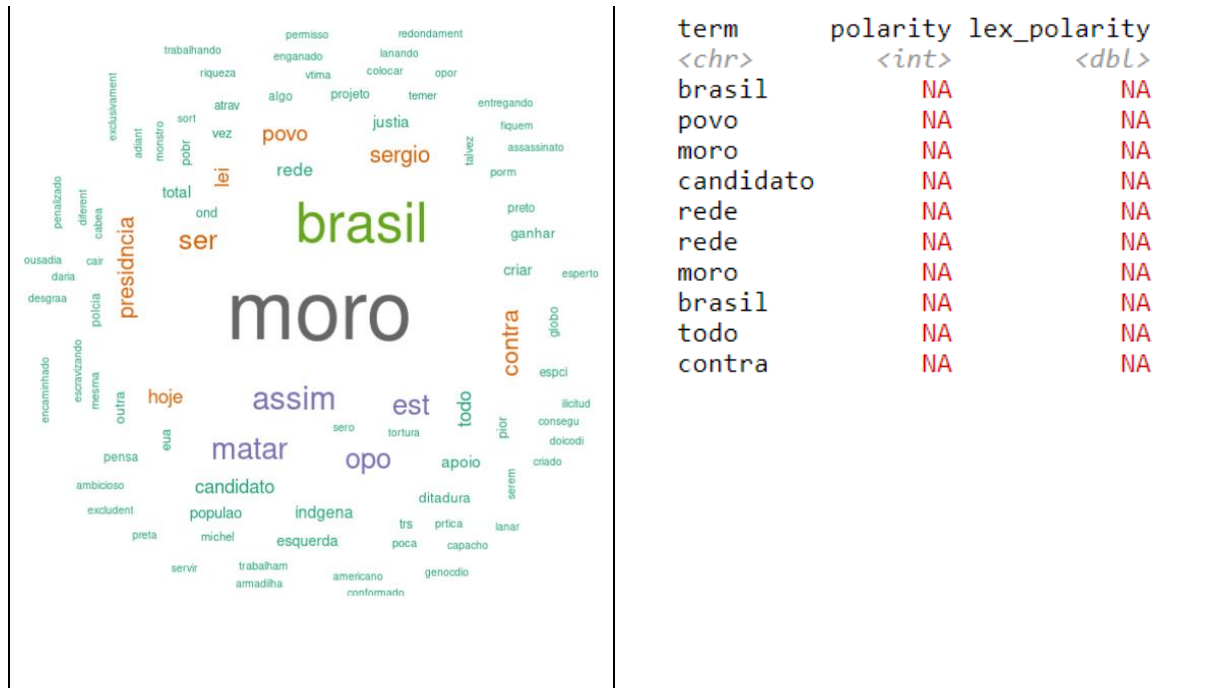
Conjunto	Análise de sentimentos																																				
	<table><tr><th>term</th><th>polarity</th><th>lex_polarity</th></tr><tr><th>&lt;chr&gt;</th><th>&lt;int&gt;</th><th>&lt;dbl&gt;</th></tr><tr><td>president</td><td>NA</td><td>NA</td></tr><tr><td>president</td><td>NA</td><td>NA</td></tr><tr><td>president</td><td>NA</td><td>NA</td></tr><tr><td>jair</td><td>NA</td><td>NA</td></tr><tr><td>president</td><td>NA</td><td>NA</td></tr><tr><td>president</td><td>NA</td><td>NA</td></tr><tr><td>governo</td><td>NA</td><td>NA</td></tr><tr><td>governo</td><td>NA</td><td>NA</td></tr><tr><td>governo</td><td>NA</td><td>NA</td></tr><tr><td>lei</td><td>NA</td><td>NA</td></tr></table>	term	polarity	lex_polarity	<chr>	<int>	<dbl>	president	NA	NA	president	NA	NA	president	NA	NA	jair	NA	NA	president	NA	NA	president	NA	NA	governo	NA	NA	governo	NA	NA	governo	NA	NA	lei	NA	NA
term	polarity	lex_polarity																																			
<chr>	<int>	<dbl>																																			
president	NA	NA																																			
president	NA	NA																																			
president	NA	NA																																			
jair	NA	NA																																			
president	NA	NA																																			
president	NA	NA																																			
governo	NA	NA																																			
governo	NA	NA																																			
governo	NA	NA																																			
lei	NA	NA																																			
	<table><tr><th>term</th><th>polarity</th><th>lex_polarity</th></tr><tr><th>&lt;chr&gt;</th><th>&lt;int&gt;</th><th>&lt;dbl&gt;</th></tr><tr><td>compr</td><td>NA</td><td>NA</td></tr><tr><td>lixo</td><td>NA</td><td>NA</td></tr><tr><td>melhor</td><td>1</td><td>0</td></tr><tr><td>mudana</td><td>NA</td><td>NA</td></tr><tr><td>planeta</td><td>NA</td><td>NA</td></tr><tr><td>preocupado</td><td>0</td><td>-1</td></tr><tr><td>saco</td><td>NA</td><td>NA</td></tr><tr><td>vendo</td><td>NA</td><td>NA</td></tr><tr><td>cujo</td><td>NA</td><td>NA</td></tr><tr><td>definio</td><td>NA</td><td>NA</td></tr></table>	term	polarity	lex_polarity	<chr>	<int>	<dbl>	compr	NA	NA	lixo	NA	NA	melhor	1	0	mudana	NA	NA	planeta	NA	NA	preocupado	0	-1	saco	NA	NA	vendo	NA	NA	cujo	NA	NA	definio	NA	NA
term	polarity	lex_polarity																																			
<chr>	<int>	<dbl>																																			
compr	NA	NA																																			
lixo	NA	NA																																			
melhor	1	0																																			
mudana	NA	NA																																			
planeta	NA	NA																																			
preocupado	0	-1																																			
saco	NA	NA																																			
vendo	NA	NA																																			
cujo	NA	NA																																			
definio	NA	NA																																			











Quadro 4: Análise de sentimentos para os agrupamentos que apresentaram nomes de candidatos gerados pelo algoritmo k-medoids.

No que diz respeito ao DBSCAN, seu funcionamento difere um pouco dos demais algoritmos pois não informamos o número máximo de conjuntos e sim o eps, ou seja, a distância máxima ponto a ponto para considerar dois pontos no mesmo agrupamento. Com base nisso fizemos testes com alguns valores conforme apresentado na Tabela 6.

EPS	Número de conjuntos	Número de ruído	Tamanho dos conjuntos
1	93	1155	1155, 1134, 5, 21, 18, 9, 10, 7, 5, 9, 12, 5, 7, 6, 7, 12, 7, 5, 6, 15, 10, 6, 9, 5, 5, 24, 12, 7, 14, 5, 14, 7, 11, 6, 6, 6, 5, 11, 5, 5, 5, 7, 8, 6, 5, 7, 7, 10, 5, 6, 5, 5, 5, 8, 6, 8, 7, 28, 8, 6, 5, 17, 5, 5, 7, 13, 6, 17, 6, 8, 6, 6, 6, 5, 8, 7, 7, 5, 5, 6, 5, 5, 5, 9, 6, 8, 10, 8, 7, 6, 4, 5, 5, 5
2	15	278	278, 2652, 7, 6, 9, 9, 6, 8, 7, 9, 6, 6, 5, 5, 5
3	7	121	121, 2861, 8, 8, 9, 6, 5
5	5	37	37, 2957, 8, 6, 10
7	4	13	13, 2985, 8, 12

9	5	6	6, 2987, 8, 12, 5
---	---	---	-------------------

Tabela 6: Resumo dos resultados do algoritmo DBSCAN.

Dessa forma concluímos que, quanto maior o número EPS, menor o ruído encontrado já que os pontos tem maior espaço para se mover e tentar se encaixar em algum conjunto, e o número de conjuntos parece estabilizar entre 4 e 5, porém tendo um conjunto com muitos termos e outros bem menores.

Selecionamos então os conjunto com EPS = 5 para geração da nuvem de palavras conforme Figura 8 e similar aos demais algoritmos. O conjunto de dados que contiver nomes de possíveis candidatos faremos a análise de sentimento das palavras próximas conforme Quadro 5.

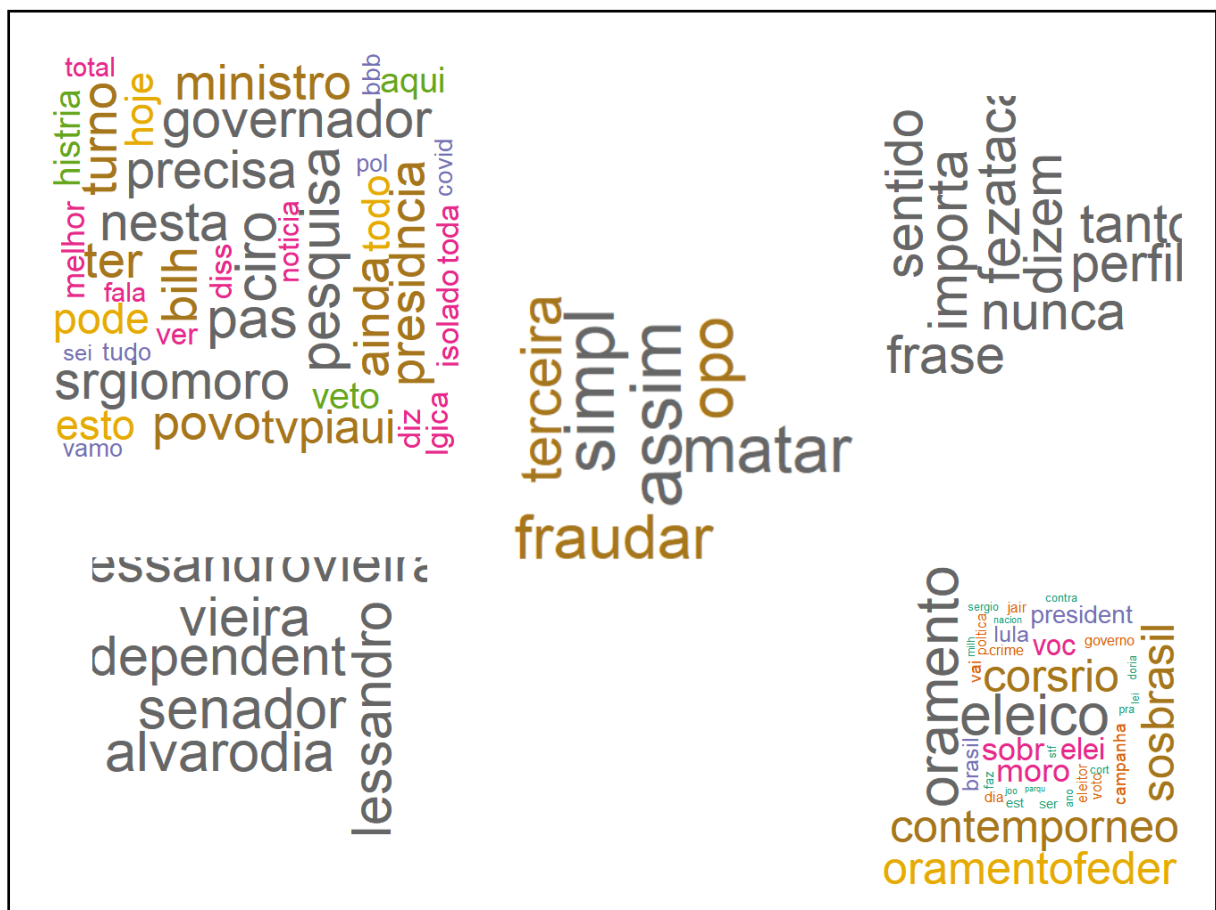


Figura 8: Nuvem de termos mais frequentes para o algoritmo DBSCAN com eps=5.

Quadro 5: Análise de sentimentos para os agrupamentos que apresentaram nomes de candidatos gerados pelo algoritmo DBSCAN.

## 6. Interpretação dos Resultados

O problema proposto consiste na identificação da opinião da população sobre as eleições para presidente que foi realizada Janeiro de 2022 com objetivo de tentar identificar algum possível candidato com mais comentários positivos podendo, supostamente, vir a ser eleito.

Para isso, obtivemos 1.539 mensagens obtidas do Twitter e Facebook filtrados com base nos termos “bolsonaro”, “sergiomoro” e “eleições2022”. Nesse conjunto de dados foram aplicadas técnicas de processamento textual e realizado o agrupamento com base em 3 diferentes algoritmos, sendo k-means, k-medoids e DBSCAN.

Também obtivemos os termos mais frequentes em todo o conjunto de forma quantitativa, considerando o termo como negativo ou positivo. De forma geral, vimos que alguns dos termos mais frequentes são: sosbrasil, orcamentofederal, eleicao, orçamento e presidente, podendo ser interpretado como uma preocupação com o futuro do Brasil com base no orçamento federal e eleições.

Com relação ao número de mensagens referentes aos candidatos, conforme Tabela 4, vimos que o ex juiz Sergio Moro é o que está sendo citado com maior frequência, seguido pelo ex presidente Lula e então atual presidente Bolsonaro. Isso poderia indicar um possível segundo turno entre Sergio Moro e Lula.

Em referência aos resultados gerados pelo algoritmo k-means, temos como melhor conjunto o com  $k=7$  devido a seus resultados intra e entre conjuntos. Na Figura 5 podemos visualizar a divisão desses conjuntos e demonstrando uma possível apreensão com o orçamento federal que vem sendo discutido junto com as eleições. Outro conjunto com grande frequência de termos como moro, lula e presidência, demonstrando um possível segundo turno entre esses candidatos. Ainda atrelando ao atual presidente Bolsonaro, presidência e ao governo, a outros dois conjuntos com termos como corrupção, campanha, eleições, entre outros.

Quando a análise de sentimentos para os conjuntos do k-means que apresentam nomes de candidatos, na maior parte foi inconclusiva, o que pode acontecer por a palavra de

forma isolada ser neutra ou também pelo o termo não constar nos dicionários em português que foram utilizados, o Oplexicon e o SentiLex.

Similar ao algoritmo anterior, o k-medoids com  $k=7$  também apresenta melhor resultado conforme métricas adotadas contendo 4 conjuntos bem definidos dentre eles sendo, um, termos referentes ao atual presidente, outro mencionando o ex-juiz Sergio Moro, outro mencionando o ex-presidente Lula e outros com termos referentes as eleições. Com isso, é possível visualizar a existência de 3 separações de voto e opinião bem definidos pelos usuários das redes sociais Twitter e Facebook que, de uma forma geral, podem refletir a opinião geral da sociedade. Quando a análise de sentimentos dos termos relacionados aos conjuntos, a mesma também foi neutra.

Por último, quanto ao algoritmo DBSCAN, seu resultado é similar aos demais algoritmos na questão de haver 1 ou 2 grandes conjuntos de termos e outros menores. Nesse caso o destaque é para o ex juiz Sergio Moro e o ex presidente Lula que aparece em dois conjuntos através de termos diferentes. A análise de sentimentos segue a linha dos demais resultados considerando como neutra a maior parte das palavras com maior frequência.

Portanto, podemos observar que em todos os algoritmos existem alguns conjuntos de termos centrais e conjuntos menores de assuntos paralelos, sendo a divisão de conjuntos entre os algoritmos semelhantes. Resumindo, podemos encontrar nos 3 algoritmos um conjunto de dados tratando do ex juiz Sergio Moro, outro mencionando o ex presidente Lula e um deles com termos referentes ao atual presidente Bolsonaro.

De acordo com o objetivo geral do trabalho, podemos aferir que existe uma grande preocupação da população com relação as eleições, a aprovação do orçamento de gastos pelo governo e também mencionando Bolsonaro, Lula e Sergio Moro como os 3 principais candidatos a vencer as eleições presidenciais de 2022, sendo possível um segundo turno entre Lula e Moro.

Quanto aos trabalhos futuros, desejamos aprofundar a análise referente aos sentimentos, trabalhando com toda as mensagens mencionando os candidatos e não apenas



os termos de forma a criar um índice de mensagens positivas / mensagens negativas sobre cada candidato.

Uma das dificuldades encontradas foi devido as bibliotecas e dicionário de dados para se trabalhar com a análise de sentimentos pois a maior parte é voltada para a língua inglesa e os demais existentes são menos robustos e com menor número de palavras cadastradas.

## 7. Apresentação dos Resultados

De forma a sintetizar os resultados obtidos, apresentamos um resumo da pesquisa no Quadro 6 através do uso de um *dashboard*.

<b>Análise das eleições brasileiras para presidente através do agrupamento e análise de sentimentos</b>		
<p><b>Declaração do problema:</b></p> <p>Para as empresas e candidatos as eleições é essencial que se possa saber qual a opinião geral do público e/ou eleitores de seus produtos, as redes sociais tornam-se um local excelente para isso, visto que as pessoas procuram ser mais sinceras em suas opiniões em um local onde acreditam não estarem sendo monitoradas. Com base nisso, temos como problema a identificação geral da opinião da população referente as eleições presidenciais de 2022.</p>	<p><b>Resultados/ Predições:</b></p> <p>Como resultados, temos que os 3 algoritmos apresentam resultados similares, tendo em geral um grande conjunto e os demais menores, porém sempre separando em conjuntos termos referentes ao atual presidente, ao ex presidente Lula e ao juiz Sergio Moro. Com base nisso podemos prever que possa existir um segundo turno entre o ex presidente Lula e o ex juiz Sergio Moro.</p> <p>Quanto a análise de sentimentos, houve uma dificuldade em fazer a análise da maior parte dos termos, em parte pelos dicionários destinados a língua portuguesa serem menos robustos que os da língua inglesa e em parte por alguns termos serem neutros.</p>	<p><b>Aquisição de dados:</b></p> <p>Os dados foram obtidos de 2 bases de dados conforme os seguintes termos: “eleições2022”, “bolsonaro”, “sergiomoro”. A primeira extração foi feita através da API do Twitter, onde foram obtidos 1.380 mensagens, e a segunda extração foi feita de forma manual do Facebook totalizando 1.539 mensagens.</p>
<p><b>Modelagem:</b></p> <p>A modelagem foi feita com base na escolha de 3 algoritmos de agrupamento, sendo: k-means, k-medoids e DBSCAN. Para cada um deles executamos testes com diversos valores de k, e eps no caso do DBSCAN.</p> <p>Após as execuções fizemos a</p>	<p><b>Modelo de avaliação:</b></p> <p>O modelo de avaliação utilizado para os algoritmos de classificação foi basicamente as métricas intra e entre conjuntos sendo que, quando menor os valores entre os elementos de um conjunto e maior o valor da distância entre conjuntos,</p>	<p><b>Preparação dos dados:</b></p> <p>A preparação dos dados foi feita conforme pré processamento textual, onde somente a coluna “text” de ambas as bases foi selecionada e após a união das mesmas, todas as mensagens foram convertidas para letras minúsculas e ASCII retirando</p>

<p>análise conforme as métricas de avaliação definidas e escolhemos o melhor modelo de cada algoritmo para que fosse gerado a nuvem de palavras daquele conjunto, e para cada conjunto em que apareceu o nome de algum possível candidato, fizemos a análise de sentimentos dos termos do conjunto de acordo com os dicionários Oplexicon e SentiLex.</p>	<p>melhor.</p> <p>Também consideramos o tamanho dos conjuntos, ou seja, sua quantidade de elementos, e no caso do DBSCAN, o nível de ruído.</p> <p>Para a análise de sentimentos consideramos dicionários na língua portuguesa para fazer a classificação dos termos em positivo, negativo ou neutro.</p>	<p>pontuações, letras, caracteres especiais e também itens personalizados de redes sociais como RT (retweet) e @, utilizado para marcar usuários e contas nas postagens. Foi removido links, imagens e as palavras-chave utilizadas como filtro para aquisição dos dados. As mensagens foram convertidas para texto puro e aplicado <i>stemming</i> para reduzir as palavras a seus radicais. Os termos com menos de 3 caracteres também foram removidos. E por fim, o conjunto de dados foi convertido para uma matriz de termo x frequência.</p>
---	---	--

Quadro 6: Resumo da pesquisa em *dashboard*.

## 8. Links

Abaixo segue o link para apresentação e download do repositório de dados do trabalho, que contém as bases de dados utilizadas e também o arquivo com todos os comandos executados.

**Link do vídeo de apresentação :**

<https://youtu.be/j4DPYlhOvRI>

**Links do repositório:**

[https://github.com/GillesVilleneuve/TCC\\_PUCMINAS](https://github.com/GillesVilleneuve/TCC_PUCMINAS)

[https://drive.google.com/drive/folders/1aUPxuvNFsa0EKNNAuxQCNihhX9KidWJ\\_?usp=shari](https://drive.google.com/drive/folders/1aUPxuvNFsa0EKNNAuxQCNihhX9KidWJ_?usp=sharing)

[ng](#)

## REFERÊNCIAS

- BERRY, Michael W. Browne. **Algorithms and applications for approximate nonnegative matrix factorization**. Computational Statistics Data Analysis. 2006.
- HAN, J. e KAMBER, M. **Data mining concepts and techniques**. Morgan Kaufmann. 2001.
- KLINCZAK, Marjori. **Identificação e propagação de temas em redes sociais**. Dissertação de mestrado apresentada a UTFPR. UTFPR, Curitiba - PR. 2016.
- SEIF, George. **The 5 Clustering Algorithms Data Scientists Need to Know**. Disponível em <<https://towardsdatascience.com/the-5-clustering-algorithms-data-scientists-need-to-know-a36d136ef68>>. Acessado em 21 de dezembro de 2021.
- TUMASJAN, A., SPRENGER, T. O., SANDNER, P. G., & WELPE, I. M. **Predicting elections with twitter: What 140 characters reveal about political sentiment**. Proceedings of the Fourth International AAAI Conference on Weblogs and Social Media. 2010.
- XIAOHUI. Yan Jiafeng, GUO Shenghua, L. X. C. Y. W. **Learning topics in short texts by non-negative matrix factorization on term correlation matrix**. Proceedings of the 2013 SIAM International Conference on Data Mining. 2013.

## APÊNDICE

### Programação/Scripts

#### #credenciais para conexao com a api do twitter

```
key <- "d7SMO6K8Ik4M6Q84vcpEwjN3J"
secret <- "NTa6aj3ktNM4qFOo6zXflntRBuvysXGKb8ZHwiz9uvinT3Dy0Z"
token <- "32443967-WMJOGbVe2Nlj0jQAAyZnFhvnrz3H4gks5pIhB0Ee4"
secretToken <- "Vjygbbhaf8o9Zx8u8TjwAXjs0EdhHvK7CS2Xa8GnkmXfw"
setup_twitter_oauth(key, secret, token, secretToken)
```

#### #coleta de 300 mensagens por palavra-chave em português

```
coleta1 <- searchTwitter('#eleições2022', n=300, lang="pt-br")
coleta2 <- searchTwitter('#bolsonaro', n=300, lang="pt-br")
coleta3 <- searchTwitter('#sergiomoro', n=300, lang="pt-br")
```

#### #conversao das mensagens para data frame

```
df1 <- do.call("rbind", lapply(coleta1, as.data.frame))
df2 <- do.call("rbind", lapply(coleta2, as.data.frame))
df3 <- do.call("rbind", lapply(coleta3, as.data.frame))
```

#### #unificação das 3 bases em um único conjunto

```
conjunto1 = rbind(df1,df3)
conjuntoTwitter = rbind(conjunto1,df2)
```

#### #gravar os dados em arquivo txt para consulta futura

```
write.table(conjuntoTwitter , file="dadosTwitter.txt")
```

#### #leitura dos dados para continuidade posterior do trabalho

```
#df <- read.table('dadosTwitter.txt', head=T, sep=',')
```

#### #leitura dos dados do Facebook, já em data frame

```
dadosFacebook <- read_excel("dadosFacebook.xls")
```

```
View(dadosFacebook)
dim(dadosFacebook)
```

### #unificação das bases de dados

```
conjuntoTotal = rbind(conjuntoTwitter[1],dadosFacebook)
```

### #gravar em excel ou txt

```
write.table(conjuntoTotal , file="conjuntoTotal.txt")
```

### #codificacao

```
tweets <- sapply(conjuntoTotal$text,function(row) iconv(row, "latin1", "ASCII", sub=""))
```

```
myCorpus <- VCorpus(VectorSource(tweets))
myCorpus2 <- tm_map(myCorpus, tolower)
```

### # remove retweet entities

```
removeRT <- function(x) gsub("(RT|via)((?:\\b\\W*@\\w+)+)", "", x)
myCorpus3 <- tm_map(myCorpus2, removeRT)
```

```
removert <- function(x) gsub("(rt|via)((?:\\b\\W*@\\w+)+)", "", x)
myCorpus4 <- tm_map(myCorpus3, removert)
```

### # remove at people

```
removePeople <- function(x) gsub("@\\w+", "", x)
myCorpus5 <- tm_map(myCorpus4, removePeople)
```

```
myCorpus6 <- tm_map(myCorpus5, removePunctuation)
myCorpus7 <- tm_map(myCorpus6, removeNumbers)
```

```
removeURL <- function(x) gsub("http[:]a-zA-Z]*", "", x)
myCorpus8 <- tm_map(myCorpus7, removeURL)
```

```
myStopwords <- c(stopwords(kind="pt"), "eleições2022", "bolsonaro", "sergiomoro")
myCorpus9 <- tm_map(myCorpus8, removeWords, myStopwords)
```

```
myStopwords2 <- c(stopwords(kind="pt"), "href")
myCorpus10 <- tm_map(myCorpus9, removeWords, myStopwords)
```

```
myCorpus11 <- tm_map(myCorpus9, stripWhitespace)
myCorpus12 <- tm_map(myCorpus11, PlainTextDocument)
myCorpus13 <- tm_map(myCorpus12, stemDocument)
inspect(myCorpus13[1])
```

#criacao da matriz termo x documento

```
myTdmReal <- TermDocumentMatrix(myCorpus13,control=list(wordLengths=c(3,Inf)))
```

#gera a matriz

```
m <- as.matrix(myTdmReal)
```

#ver a frequencia das palavras

```
freq <- rowSums(m)
```

#pego as palavras com frequencia maior ou igual a 5 caracteres

```
freq <- subset(freq, freq>=5)
```

#gera o grafico

```
barplot(freq, las=2, col=rainbow(25))
```

#criacao do conjunto do modelo

```
ap_td <- tidy(myTdmReal)
```

```
ap_td
```

#verifica a classificação de cada termo como positivo ou negativo

```
ap_sentiments <- ap_td %>%
```



```
inner_join(get_sentiments("bing"), by = c(term = "word"))
```

#### #encontra os termos mais negativos

```
ap_negativo_bing <- ap_sentiments %>%
  count(term, sentiment, wt = count) %>%
  spread(sentiment, n, fill = 0) %>%
  mutate(sentiment = positive - negative) %>%
  arrange(sentiment)
```

```
wordcloud(ap_negativo_bing$term, ap_negativo_bing$negative, min.freq=1)
```

#### #encontra os termos mais positivos

```
ap_positivo_bing <- ap_sentiments %>%
  count(term, sentiment, wt = count) %>%
  spread(sentiment, n, fill = 0) %>%
  mutate(sentiment = negative - positive) %>%
  arrange(sentiment)
```

```
wordcloud(ap_positivo_bing$term, ap_positivo_bing$positive, min.freq=1)
```

#### #gera o gráfico termos positivos x negativos

```
ap_sentiments %>%
  count(sentiment, term, wt = count) %>%
  filter(n >= 1) %>%
  mutate(n = ifelse(sentiment == "negative", -n, n)) %>%
  mutate(term = reorder(term, n)) %>%
  ggplot(aes(term, n, fill = sentiment)) +
  geom_bar(stat = "identity") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  ylab("Nível de sentimento")+
  xlab("Termo")
```

### # Léxicos da língua portuguesa para análise de sentimentos

```
oplex <- lexiconPT::oplexicon_v3.0
sentilex <- lexiconPT::sentiLex_lem_PT02
dplyr::glimpse(oplex)
dplyr::glimpse(sentilex)
```

### #K-Means = 3, 5, 7, 9

```
matrix.kmeans <- kmeans(myTdmReal, 7)
matrix.kmeans$size
str(matrix.kmeans)
```

### #rodar um para cada cluster

```
docs1 <- which(matrix.kmeans$cluster ==4)
docs1 <- myTdmReal[docs1, ]
#head(docs1); length(docs1)
matrix <- as.matrix(docs1)
words <- sort(rowSums(matrix),decreasing=TRUE)
df <- data.frame(word = names(words),freq=words)
wordcloud(words = df$word, freq = df$freq, min.freq = 1, max.words=200,
random.order=FALSE, rot.per=0.35, colors=brewer.pal(8, "Dark2"))
```

### #um para cada conjunto

```
ap_td <- tidy(docs1)
ap_td %>%
  left_join(oplex , by = "term") %>%
  left_join(sentilex %>% select(term, lex_polarity = polarity), by = "term") %>%
  select(term, polarity, lex_polarity) %>%
  head(10)
```

```
#K-medoids = 3, 5, 7, 9
```

```
matrix.kmedoids <- pam(myTdmReal, 5)
```

```
matrix.kmedoids$clusinfo
```

```
#rodar um para cada cluster
```

```
docs1 <- which(matrix.kmedoids$clustering ==7)
```

```
docs1 <- myTdmReal[docs1, ]
```

```
#head(docs1); length(docs1)
```

```
matrix <- as.matrix(docs1)
```

```
words <- sort(rowSums(matrix),decreasing=TRUE)
```

```
df <- data.frame(word = names(words),freq=words)
```

```
wordcloud(words = df$word, freq = df$freq, min.freq = 1, max.words=200,  
random.order=FALSE, rot.per=0.35, colors=brewer.pal(8, "Dark2"))
```

```
#um para cada conjunto
```

```
ap_td <- tidy(docs1)
```

```
ap_td %>%
```

```
left_join(oplex , by = "term") %>%
```

```
left_join(sentilex %>% select(term, lex_polarity = polarity), by = "term") %>%
```

```
select(term, polarity, lex_polarity) %>%
```

```
head(10)
```

```
#K-medoids = 3, 5, 7, 9
```

```
matrix.kmedoids <- gmm(myTdmReal, 3)
```

```
matrix.kmedoids$clusinfo
```

```
#rodar um para cada cluster
```

```
docs1 <- which(matrix.kmedoids$clustering ==7)
```

```
docs1 <- myTdmReal[docs1, ]
```

```
#head(docs1); length(docs1)
```

```
matrix <- as.matrix(docs1)
```

```
words <- sort(rowSums(matrix),decreasing=TRUE)
df <- data.frame(word = names(words),freq=words)
wordcloud(words = df$word, freq = df$freq, min.freq = 1, max.words=200,
random.order=FALSE, rot.per=0.35, colors=brewer.pal(8, "Dark2"))
```

#um para cada conjunto

```
ap_td <- tidy(docs1)
ap_td %>%
  left_join(oplex , by = "term") %>%
  left_join(sentilex %>% select(term, lex_polarity = polarity), by = "term") %>%
  select(term, polarity, lex_polarity) %>%
  head(10)
```

#DBscan

#gerar um para cada eps

```
dbscan.result <- dbscan(myTdmReal, 3)
```

#um para cada cluster

```
docs1 <- which(dbscan.result$cluster ==0)
docs1 <- myTdmReal[docs1, ]
#head(docs1); length(docs1)
matrix <- as.matrix(docs1)
words <- sort(rowSums(matrix),decreasing=TRUE)
df <- data.frame(word = names(words),freq=words)
wordcloud(words = df$word, freq = df$freq, min.freq = 1, max.words=200,
random.order=FALSE, rot.per=0.35, colors=brewer.pal(8, "Dark2"))
```

#um para cada conjunto

```
ap_td <- tidy(docs1)
ap_td %>%
  left_join(oplex , by = "term") %>%
  left_join(sentilex %>% select(term, lex_polarity = polarity), by = "term") %>%
```

```
select(term, polarity, lex_polarity) %>%
head(10)
```

### **Bibliotecas utilizadas**

```
library(twitterR)
library(readxl)
library(tm)
library(SnowballC)
library(dplyr)
library(tidyr)
library(tidytext)
library(ggplot2)
library(reshape2)
library(textdata)
library(cluster)
library(wordcloud)
library(lexiconPT)
library(fpc)
```

### **Versão Utilizada**

```
> version

platform      x86_64-pc-linux-gnu
arch           x86_64
os             linux-gnu
system         x86_64, linux-gnu
status
major          4
minor          1.2
year           2021
month          11
day            01
svn rev        81115
language       R
version.string  R version 4.1.2 (2021-11-01)
nickname       Bird Hippie
```