UNIVERSIDADE FEDERAL DO ESTADO DO RIO DE JANEIRO
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
ESCOLA DE INFORMÁTICA APLICADA

Eleições Presidenciais 2018: Um estudo de caso de busca e avaliação de Informações na Web

Ricardo Augusto Blei Sant´Anna das Neves

Orientadora
Morganna Carmem Diniz

RIO DE JANEIRO, RJ – BRASIL
NOVEMBRO DE 2018

Catalogação informatizada pelo(a) autor(a)

N511

Neves, Ricardo
Eleições Presidenciais 2018: Um estudo de caso de busca e avaliação de Informações na Web / Ricardo Neves. -- Rio de Janeiro, 2018.
77

Orientador: Morganna Carmem Diniz .
Trabalho de Conclusão de Curso (Graduação) - Universidade Federal do Estado do Rio de Janeiro, Graduação em Sistemas de Informação, 2018.

1. Análise de Sentimentos. 2. Busca na Web. 3. Eleições 2018. I. Carmem Diniz , Morganna , orient. II. Título.

Eleições Presidenciais 2018: Um estudo de caso de busca e avaliação de Informações na Web

Ricardo Augusto Blei Sant´Anna das Neves

Projeto de Graduação apresentado à Escola de Informática Aplicada da Universidade Federal do Estado do Rio de Janeiro (UNIRIO) para obtenção do título de Bacharel em Sistemas de Informação.

Aprovado por:

Morganna Carmem Diniz, D. Sc. (UNIRIO)

Simone Bacellar Leal Ferreira, D. Sc. (UNIRIO)

RIO DE JANEIRO, RJ – BRASIL.
NOVEMBRO DE 2018

"People think dreams aren't real just because they aren't made of matter, of particles. Dreams are real, but they are made of viewpoints, of images, of memories and puns and lost hopes."

Neil Gaiman

AGRADECIMENTOS

Agradeço a Deus, que sempre se fez presente na minha vida e me deu forças nos momentos mais difíceis.

Agradeço aos meus Pais Rogerio e Silvana pelo amor dedicados a mim durante todos os anos.

Um agradecimento especial à minha esposa Sabrina, pela paciência e apoio durante todos esses anos.

À minha Irmã Vânia e Cunhado Bruno pelo apoio moral e incentivo.

À todos os professores, em especial a professora Flávia Maria Santoro pelo conhecimento, acompanhamento e apoio recebidos nos últimos anos.

À minha orientadora, professora Morganna, que aceitou me orientar de última hora, sendo importante no fim desse capítulo da minha trajetória.

RESUMO

No mundo atual, onde quintilhões de bytes de dados são gerados diariamente, fica cada vez mais importante buscar formas e ferramentas para extrair informação e conhecimento dos mesmos e que poderão ser usados das mais variadas formas possíveis. Este trabalho tem como objetivo analisar e demonstrar algumas dessas formas usando como estudo as eleições presidenciais no Brasil de 2018.

Para analisar os sentimentos dos eleitores durante o período eleitoral (setembro e outubro de 2018) foi criada uma base de *tweets* (pequenas mensagens) escritos em português brasileiro. Essa base foi pré-processada para criar um corpus (conjunto) de mensagens com menos ruídos. Esse corpus foi analisado para a extração do sentimento presente nas mensagens.

Além dos *tweets*, foi usada uma ferramenta complementar chamada *Trends* da empresa Google, que complementou a pesquisa e ampliou o entendimento dos resultados obtidos.

Os dados foram analisados e o resultado final demonstrou uma polaridade positiva e negativa das mensagens bem dividida entre os dois candidatos mais cotados para a disputa do segundo turno (segundo pesquisa DataFolha do dia 03/10/18): Jair Bolsonaro e Fernando Haddad.

Palavras-chave: Sistemas de informação, Análise de Sentimentos, Web, Busca na Web, Eleições 2018.

ABSTRACT

In today's world, where quintiles of data bytes are generated on a daily basis, it is increasingly important to seek ways and tools to extract information and knowledge from them that can be used in as many different ways as possible. This paper aims to analyze and demonstrate some of these forms using as a study the presidential elections in Brazil in 2018.

In order to analyze voters' feelings during the electoral period (September and October 2018) a database of tweets written in Brazilian Portuguese was created. This database was preprocessed to create a corpus of messages with less noise. This corpus was analyzed for the extraction of the feeling present in the messages.

In addition to the tweets, a complementary tool called Google Enterprise Trends was used, which complemented the research and broadened the understanding of the results obtained.

The data were analyzed and the final result showed a positive and negative polarity of the messages well divided between the two candidates most quoted for the second round match (according to research DataFolha of 10/03/18): Jair Bolsonaro and Fernando Haddad.

Keywords: Information Systems, Sentiment Analysis, Web, Web Search, 2018 Elections.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABELA

1. Introdução

A Internet é uma grande rede de computadores interligados e dentre seus vários objetivos, o mais importante é o acesso à informação. Com o seu surgimento, houve um aumento exponencial na quantidade de informação disponibilizada para população e com ela, começou uma busca incessante para descobrir ferramentas que permitissem acessar de forma rápida e precisa informações armazenadas em banco de dados gigantescos (Branski, 2008).

Estima- se que a cada dia sejam criados cerca de 2,5 quintilhões de bytes de dados e a quantidade de dados existente no mundo dobre a cada dois anos (Junior, 2016).

As ferramentas de busca ou mecanismos de busca são sistemas especializados utilizados na recuperação de informações na Internet. Elas coletam e armazenam as informações dos sites em bancos de dados que estão disponíveis para consulta. Realizando uma busca, o usuário pode descobrir a localização exata das informações que deseja (Branski, 2008).

Para direcionar melhor sua pesquisa e encontrar uma determinada informação, o usuário tem que ter uma ideia do tipo de informação que gostaria de encontrar. Assim, ele precisa escolher a ferramenta mais apropriada para sua empreitada. Outro fator que deve ser observado é que quanto mais genérico o assunto, maior a chance que o usuário tem de receber múltiplas respostas durante a busca. Isto porque a procura geralmente é feita por palavras, logo quanto mais palavras o usuário informar, menos respostas o usuário irá obter e estas terão uma precisão maior (Junior, 2016).

1.1. Motivação

Um relatório intitulado "Digital in 2018" apresentado no começo deste ano, mostrou que mais de metade da população mundial está online (cerca de 4,021 bilhões) e que cerca de 250 milhões conseguiram acesso pela primeira vez em 2017. A figura 1 apresenta um resumo do relatório e demonstra as formas que as pessoas se conectam nas diversas plataformas.

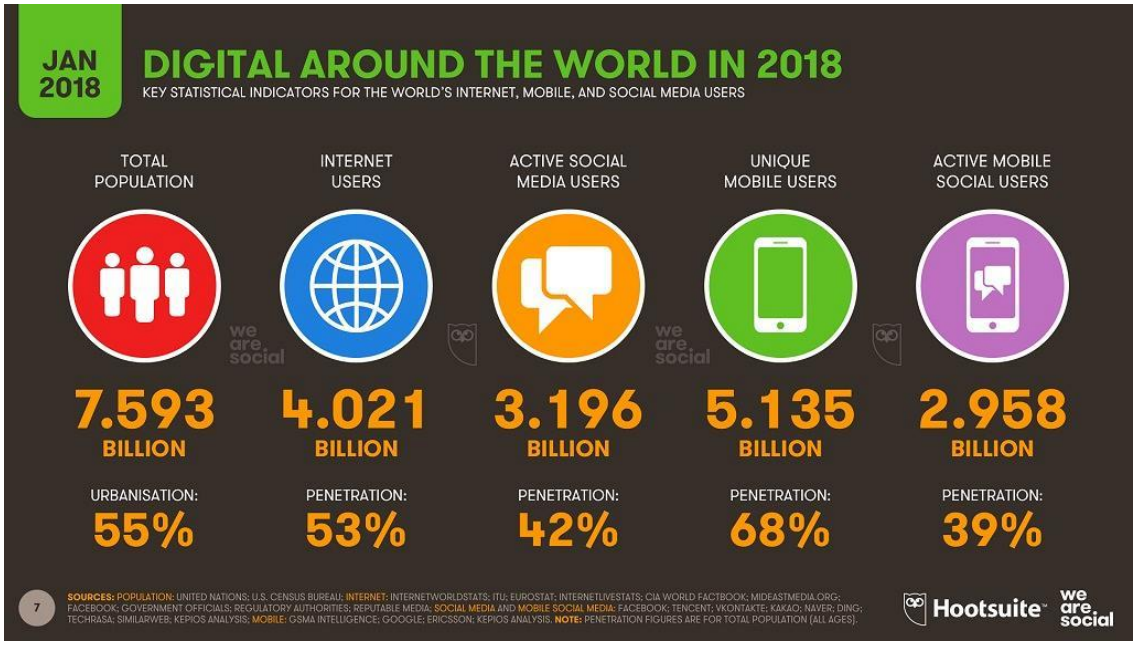

Figura 1: Pessoas conectadas em diferentes plataformas ("Digital in 2018", 2018).

Atualmente o volume de dados disponíveis na rede é muito grande e tem uma velocidade de disseminação muito alta. Diariamente são publicadas milhões de novas entradas por pessoas do mundo inteiro (Branski, 2008).

Diferentemente das bibliotecas, os documentos da Internet não estão classificados segundo um padrão determinado. Portanto, o usuário precisa localizar informações em um grande volume de páginas disponíveis, sem qualquer organização (Branski, 2008).

Encontrar a informação depende, principalmente, do uso eficiente das ferramentas de busca. O usuário precisa conhecer como é coletada e estruturada a informação em diferentes bancos de dados; suas características e limitações; todas as possíveis formas de interação e suas linguagens de busca, para conseguir ter êxito em sua empreitada.

Vale ressaltar, que um fato muito importante neste contexto (fundamental na evolução do presente trabalho) é que com a disseminação da internet houve o surgimento e a popularização dos smartphones e redes sociais. As organizações começaram a perceber que precisavam saber qual era seu público alvo e como ele se comportava, o que pensava, quais seus interesses, seus desinteresses, informações que poderiam ajudar a empresa a traçar um perfil que auxiliasse no direcionamento de qual produto deve ser desenvolvido ou qual abordagem deve ser adotada para chegar ao cliente e com isso conquistá-lo a ponto de torná-lo um cliente ativo (Santos, 2017).

As redes sociais se difundiram e acabaram sendo um ambiente onde o usuário dispõe suas informações de maneira natural, por vontade própria, sem necessariamente ter que

passar por exemplo por um questionário (Santos, 2017). No contexto de redes sociais, pode-se citar o Facebook, que possui mais de 2,13 bilhões de usuários em todo o mundo e gera a maior quantidade de dados dentre as redes sociais (Estadão, 2018).

Coletar e avaliar esses dados, geralmente demanda um certo esforço. Se for de forma manual, demanda um esforço cansativo. Automatizar o processo de coleta seria mais do que o ideal nesses casos (Branski, 2008).

Assim sendo, técnicas de Inteligência e Analise de Negócio (*Business Intelligence and Analytics* – BI & A), associadas a cenários de *Big Data*, tornaram-se cada vez mais importantes nas comunidades acadêmicas e empresariais nas últimas duas décadas (Chen, 2012).

Diante do exposto, a motivação para realização desse trabalho é estudar formas de analisar e tentar entender, como as pessoas reagem na web a algum tipo de acontecimento e como isso pode ser usado para alguma finalidade.

1.2 Objetivo Geral

Este trabalho possui os seguintes objetivos gerais:

1. Apresentar e analisar algumas formas de coletar informação dentre diversos buscadores disponíveis na Internet;
2. Avaliar e tentar extrair informação dos dados coletados;
3. Transformar a informação recebida em conhecimento para algum fim específico.

Como assunto para esse estudo, serão usadas às eleições presidenciais no Brasil de 2018.

1.3. Objetivos Específicos

Para atingir o objetivo, será necessário alcançar os seguintes objetivos específicos:

1. Estudar as formas de buscar a informação;
2. Coletar as informações;
3. Analisar e discutir os resultados avaliando suas vantagens e desvantagens.

1.4. Organização do Texto

O presente trabalho está estruturado em capítulos que, além desta introdução, inclui:

- Capitulo II: Aspectos teóricos e conceituais envolvidos no desenvolvimento do trabalho. Além disso, são apresentados as ferramentas, conceitos, tecnologias e informações importantes que possuam relevância ao trabalho.
- Capitulo III: Estudo de caso.
- Capitulo IV: Conclusão do trabalho.

1.5. Metodologia de Pesquisa

Neste tópico pretende- se esclarecer a metodologia empregada no estudo de caso.

O método de pesquisa utilizado foi o qualitativo, apoiando-se em técnicas de coleta de dados quantitativas.

A pesquisa qualitativa não busca enumerar ou medir eventos, além de normalmente ser direcionada ao longo de seu desenvolvimento. Ela serve para obter dados descritivos que expressam os sentidos dos fenômenos (Neves, 1996).

O estudo foi desenvolvido a partir de:

1. Primeiro foi utilizada uma ferramenta do Google chamada *Trends*, o intuito foi ter uma visão geral do assunto abordado. A pesquisa foi feita antes de ocorrer a votação no primeiro turno (dia 29/09/2018) justamente para ter um panorama geral das eleições (conseguindo informações não apenas dos dois eventuais candidatos que disputariam o segundo turno), além de minimizar riscos de uma possível vitória em primeiro turno (cenário que empobreceria o material para a pesquisa);

2. Na análise de sentimentos, foi utilizado a API (*Application Programming Interface*) disponibilizada pelo Twitter para coleta das mensagens feitas pelos usuários na plataforma. Foram escolhidos os dois candidatos mais cotados para vencer as eleições segundo as pesquisas eleitorais do dia 03/10/2018 (Jair Bolsonaro e Fernando Haddad). As palavras chave pesquisadas foram: “Haddad” e “Bolsonaro”. A escolha das palavras na busca por apenas os nomes, se deu para encontrar o maior número possível de mensagens.

3. Foram feitas duas coletas sequencias de uma hora cada para cada um dos candidatos. A pesquisa “Bolsonaro” retornou 109.589 (cento e nove mil quinhentos e oitenta e nove) mensagens e a pesquisa “Haddad” 108.792 (cento e oito mil setecentos e noventa e dois) mensagens.

4. Os *scripts* base foram selecionados pensando na performance, pois o volume de *tweets* coletados foi muito grande e um *script* que não fosse performático, poderia levar horas ou até mesmo dias para finalizar a análise.

5. A escolha da ferramenta ifeel da Universidade Federal de Minas Gerais para a análise complementar, se deu por algumas razões: simplicidade do uso da mesma; acurácia nos resultados interessante e para mostrar/ prestigiar os estudos que estão sendo realizados por pesquisadores brasileiros nessa área. Vale ressaltar que cabe melhorias na performance da ferramenta que ainda se encontra em desenvolvimento. Para demonstrá-la no modo de análise do arquivo enviado, foi utilizado um corpo de mensagens bem reduzido (com 152 mensagens), pois a análise nesse modo demora bastante tempo.

6. Para a nuvem de palavras, foram removidos termos fora de contexto, verbos soltos, números, entre outros, visando maior clareza do resultado.

7. A performance dos Scripts também pode variar de acordo com a configuração do computador. Vale registrar que todas as análises foram feitas em um PC com processador Intel Core i7- 7400 2.9GHz com 16 Gb de memória RAM.

2. Aspectos Teóricos e Conceituais

Com a finalidade de conduzir teoricamente este trabalho, neste capítulo são abordados:

- Conceitos e funcionalidades do Google;
- Definição da rede social “Twitter”;
- “Mineração de dados” e “Mineração de Texto” com foco em linguagem natural e análise de intenções.

2.1. Google

O Google é a maior empresa de busca do planeta e chegou a um valor de mercado em 2018 de US$ 739 bilhões (Estadão, 2018). Apenas, a título de curiosidade, o PIB inteiro do Brasil é próximo a US$ 1,8 trilhão, ou seja, com tudo que o Brasil produz em um ano, seria possível comprar "apenas" duas empresas *Googles*.

A busca por um tema na página do Google segue três etapas básicas para gerar resultados.

A primeira etapa se chama “Rastreamento” e tem como objetivo descobrir quais páginas existem na Web. Não há um registro central de todas as páginas da Web, por isso, o Google precisa pesquisar constantemente novas páginas e adicioná-las à própria lista de páginas conhecidas (Suporte do Google, 2018).

A segunda etapa se chama “indexação”, etapa que começa após a descoberta de uma página. O Google tenta identificar, analisando o seu conteúdo, arquivos de imagens e vídeos incorporados sobre o que ela trata. Essa informação fica registrada no “índice do Google”, um grande banco de dados armazenado em uma quantidade enorme de computadores. Uma das razões para o sucesso do Google é justamente o seu sistema de *PageRank* (lista de popularidade do Google) que classifica os sites de acordo com a quantidade de links externos que o mesmo possui, como consequência, o conteúdo desse site é listado primeiro nas buscas, pois o *PageRank* entende que aquela página trata com maior relevância o assunto pesquisado, além de analisar os assuntos mais pesquisados e verificar quais sites tratam aquele tema de maneira mais significativa, checando a quantidade de vezes que o termo pesquisado aparece na página. Uma vez que a Web já foi vasculhada e indexada, é possível atender as consultas que os internautas fazem ao Google (Pereira, 2008).

A última etapa se chama “veiculação”. Após o usuário fazer uma consulta, o Google tenta encontrar a resposta mais relevante com base em vários fatores (a consulta primeiro passa pelo servidor Web do Google, depois pelos servidores de índice e em seguida pelos servidores de documentos, para só então serem entregues as respostas ao usuário). O Google tenta determinar as respostas mais adequadas e de qualidade mais alta, bem como avaliar outras considerações que fornecerão a melhor experiência ao usuário. Para isso, leva em conta aspectos como localização, idioma e dispositivo (computador ou smartphone). Por exemplo, em uma pesquisa por "lanchonetes", os resultados seriam diferentes para um usuário do Rio de Janeiro e outro de São Paulo (Suporte do Google, 2018).

Vale comentar dois fatores que ajudaram o Google a ser o mecanismo de busca mais utilizado na web que são: a simplicidade e a clareza. A combinação desses dois itens foi trabalhada desde a sua concepção. Fazendo com que seja possível acessar um site de busca leve, sem poluição visual e cujas opções se localiza facilmente. Uma das preocupações do Google é também manter a ética em todos os países que trabalha. Por exemplo, se alguém pesquisar sobre pedofilia, encontrará textos que abordam tal assunto de maneira legal, ou seja, investigações, estudos, notícias, mas não encontrará sites com conteúdo pedófilo (Pereira, 2008).

Outro fato curioso, é que há algum tempo atrás, o algoritmo utilizado nas buscas era atualizado a cada dois ou três meses. Dessa forma, sites com ranking elevado podiam ficar no topo da busca por um bom tempo. Hoje em dia, a concorrência é mais acirrada uma vez que o Google faz mais de seiscentas alterações no seu algoritmo por ano e a grande maioria das atualizações não é anunciada (Wix, 2018).

Embora ninguém saiba exatamente quais fatores o Google leva em consideração ou qual peso é dado a cada um desses fatores, é sabido que o Google olha os seguintes aspectos para dar o ranking aos sites:

- Qualidade do conteúdo dos sites;
- Novidade do conteúdo / com que frequência o site é atualizado;
- Quantidade e qualidade dos sites que têm link para este site;
- Palavras-chave usadas no site;
- Idade do site;
- Localização de quem faz a busca e da empresa;
- O tempo que as pessoas ficam no site (também conhecido como tempo no site);
- Além de duzentos outros fatores (Wix, 2018).

2.2. Twitter

Twitter é um serviço de microblog que permite que pessoas compartilhem suas opiniões e pensamentos de forma rápida e gratuita utilizando-se de apenas 280 caracteres. (Russell, 2013)

É apresentado como um site de rede social, pois permite aos seus usuários a construção de perfis públicos em um espaço da web, bem como prover uma estrutura para a conexão de tais perfis. Em 2016, o Twitter possuía mais de 600.000.000 (seiscentos milhões) de usuários, sendo destes, cerca de 342.000.000 (trezentos e quarenta e dois milhões) ativos (*Static Brain*, 2018). Atualmente são feitos em média 58.000.000 (cinquenta e oito milhões) de *tweets* novos por dia e cerca de 9.100 (nove mil e cem) por segundo (*Static Brain*, 2018)

O uso frequente desse tipo de serviço web gera uma quantidade enorme de dados, e é uma fonte útil de informações para analisar e compreender as tendências populares. Sua vantagem em relação a outras redes sociais para esse tipo de análise é que no Twitter só é permitido a postagem de texto, diferente do Facebook que pode ser incluídas fotos, vídeos, etc.

Nesse contexto, o presente trabalho, utilizou o Twitter como base de dados para a aplicação de técnicas de mineração com a finalidade de extrair conhecimento de um assunto específico, neste caso as eleições 2018. O Twitter foi escolhido por ser uma Rede Social que possui uma API para coleta de informações de fácil utilização e por ser uma Rede Social que não necessita da aprovação dos usuários para a coleta dos conteúdos postados nessa plataforma.

2.3. Mineração de dados

Aparecendo como uma das alternativas mais eficazes para extrair conhecimento a partir de grandes volumes de dados, a mineração de dados nos ajuda a descobrir padrões válidos, novos, potencialmente úteis e ultimamente compreensíveis que ajudam as instituições em suas tomadas de decisões ou a atingir um maior grau de confiança (Machado, 2010). O processo tem como finalidade levar resultados relevantes para o usuário, ou seja, caso a informação obtida não seja a esperada, é necessário que haja uma redefinição do processo até que o resultado obtido seja satisfatório (Teixeira, 2018).

O processo de mineração de dados é uma das etapas do KDD (*Knowledge Discovery Database*), caracterizado por várias etapas, que se tornam necessárias para a obtenção do resultado desejado.

A figura 2 mostra todas as etapas do processo de KDD, estes passos são: seleção; pré-processamento e limpeza; formatação; mineração de dados (*data mining*) e interpretação/avaliação. Como se pode notar, o processo compreende, todo o ciclo que o dado percorre até virar conhecimento ou informação (Fayyad, 1996).

A seguir é apresentado um breve resumo de cada etapa (Prass, 2016).

- Seleção: é escolhido o conjunto de dados.
- Pré-processamento: é feito uma "limpeza" da base para aumentar a qualidade da análise.
- Formatação: os dados são adequadamente armazenados para que os algoritmos de aprendizado possam ser aplicados.
- Mineração de dados: consiste na exploração e análise da base de dados.
- Interpretação: o conhecimento adquirido através da técnica de *data mining* deve ser interpretado e avaliado para que o objetivo final seja alcançado.

O processo de KDD possui duas características relevantes: é interativo (o usuário pode intervir e controlar o curso das atividades) e iterativo (sequência finita de operações) (Prass, 2016).

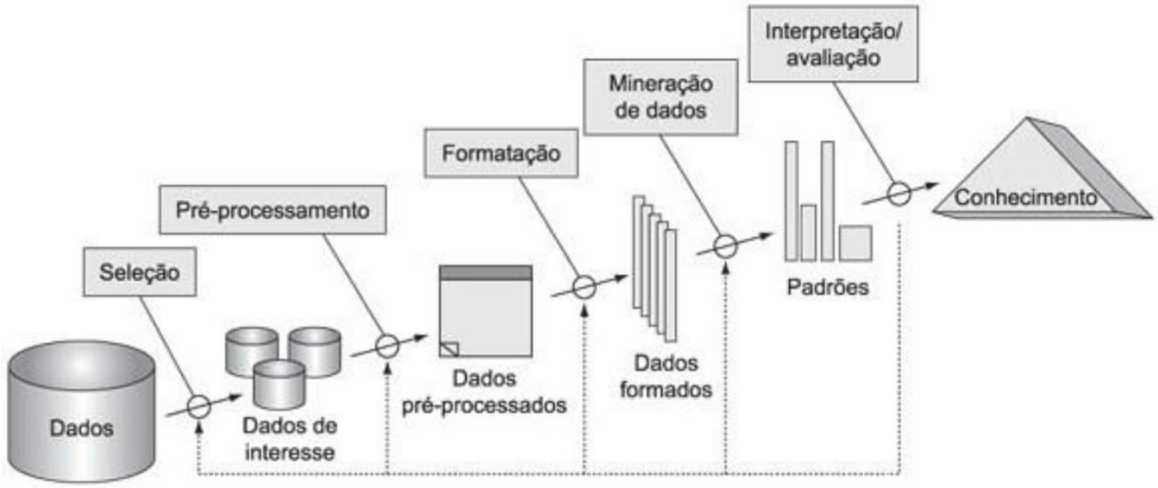

Figura 2: Processo de KDD (Piatetsky, 1996).

Dependendo do objetivo a ser alcançado através da mineração de dados, pode ser necessário aplicar técnicas para a detecção de pontos fora da curva, aprendizagem de regras de associação, agrupamento dos dados (*clustering*), classificação dos dados, regressão de dados ou sumarização. O processo de mineração de dados geralmente ocorre se utilizando de dados contidos em um depósito digital para dados (Turban, 2010).

2.4. Mineração de Textos

Enquanto a mineração de dados é aplicada a dados estruturados, a mineração de textos é aplicada na análise de textos não estruturados. Com o advento da Web e mídia sociais, este se tornou um dos conceitos mais importantes no cenário da mineração (Teixeira, 2018).

Mineração de textos é uma extensão da mineração de dados, e pode ser definida como um processo de extração de informações desconhecidas e úteis de documentos textuais escritos em linguagem natural. Como a maioria da informação é armazenada em forma de texto, a mineração de textos possui alto valor comercial, e pode ser aplicada nas mais distintas áreas de atuação (Pezzini, 2017).

Esse processo é composto por quatro passos bem definidos: seleção, pré-processamento, mineração, análise / assimilação (Morais, 2007). Essas etapas podem ser vistas no processo ilustrado na figura 3.

Figura 3: Processo de mineração de textos (Brito, 2016).

Segundo Wattenberg em 2006, a análise de redes sociais envolve três tarefas fundamentais:

- Identificar comunidades: os atores devem ser agrupados em comunidades, de acordo com os seus atributos;

- Identificar atores centrais: é necessária a identificação dos atores que possuem o maior número de conexões;

- Analisar papel e posição de relacionamentos e indivíduos: essa tarefa requer interpretação da estrutura da rede e depende dos atributos de atores e relacionamentos.

Resumidamente, a área de estudo da mineração de textos compreende cinco tarefas triviais ao processo: Seleção (Coleta de dados), Pré-Processamento de Textos, Mineração e Análise dos dados (Brito, 2016).

2.4.1. Seleção (Coleta de dados)

Esta etapa consiste na procura e no armazenamento dos dados que servirão de fonte para a pesquisa.

Diversos sites e aplicativos na web hoje em dia permitem efetuar essa coleta, como por exemplo o Twitter.

2.4.2. Pré-processamento de Textos

A etapa de pré-processamento inclui operações básicas de manipulação de dados, tais como remover ruídos ou subcamadas, coletando informação necessária para as tarefas de descoberta de conhecimento. Além disso, nessa etapa são decididas estratégias para manusear os campos que não são necessários no processo posterior de mineração de dados. É nesta etapa, que se dá início logo após o término da coleta dos dados, que é efetuada uma análise eficiente dos dados que logicamente precisam ter uma boa qualidade (Cervi, 2008).

Entre as técnicas utilizadas no pré-processamento estão presentes: limpeza de dados, *tokenização* e remoção de *stopwords*. Essas técnicas são do Processamento de Linguagem Natural (PLN) (Teixeira, 2018).

Caso não existisse um pré-processamento, a análise poderia ser afetada negativamente, apresentando resultados errados ou sem precisão.

2.4.2.1. *Tokenização*

A identificação de *tokens*, ou *tokenização* é uma importante etapa do Pré-Processamento para extrair unidades mínimas de textos. Cada unidade é chamada de *token* e esta passa a ser tratada como uma "lista". Normalmente corresponde à uma palavra do texto, podendo estar relacionado também a símbolos e caracteres de pontuação, como por exemplo vírgula (,); ponto (.) e exclamação (!) (Manning *et al*., 2008). Exemplificando, a frase: "Hoje vai fazer sol!", poderá ser dividida em cinco *tokens*. Conforme a seguir.

"Hoje vai fazer sol!"

[Hoje] [vai] [fazer] [sol] [!]

Na geração de *tokens* o "espaço" é sempre descartado, como pode ser visto acima.

2.4.2.2. Remoção de Stopwords

Ao manipular uma base textual, encontra-se muitos *tokens* que não possuem valor para o contexto, sendo úteis apenas para a compreensão geral do texto. A estas palavras, se dá o nome de *stopwords*. Todas essas *stopwords* devem ser relacionadas em uma lista, chamada de *stoplist*. A *stoplist* geralmente é composta por palavras que geralmente são vistas em qualquer pedaço de texto como, por exemplo, artigos, pontuações, pronomes e conjunções. Essas palavras normalmente não são necessárias para realizar a mineração então todas elas são retiradas do texto (Brito, 2016).

A tabela 1 mostra a identificação e remoção de *stopwords*, sendo que os *tokens* sublinhados serão descartados.

Tabela 1: Identificação e remoção de *stopwords* (os t*oken*s descartados estão sublinhados) (Brito, 2016).

StopList	Texto
de, da, do, uma para, um, tem ? ! : ; e, o, com	[eu] [acho] [que] [tem] [de] [diminuir] [a] [maioridade] [é] [pra] [14] [Eu] [sou] [contra] [a] [redução] [da] [maioridade] [penal]

2.4.2.3. Stemming e Lematização

Um dos problemas relacionado ao processamento de linguagem natural é o grande número de *tokens* que não possuem valor para análise. Seguindo esta ideia, a redução de dimensionalidade torna-se muito importante em processos de classificação automática, para determinar os melhores atributos para modelagem, bem como pelos aspectos de escalabilidade dos modelos resultantes (Kim, 2000).

No *stemming* a preocupação está em reduzir para o seu radical, já para a lematização a preocupação é o lema, onde o gênero e o número não interessam (Teixeira, 2018).

Tabela 2: Demonstração do algoritmo de stemming (Brito, 2016).

ID	Frase Normalizada	Stemming
1	Ideia genial	Ide gen
2	belo dia dirigir	bel dia dirig
3	ganhei desconto carro	ganh descont carr
4	ganhando aposta	ganh apost
5	perdi novamente aposta	perd nov apost
6	perdemos jogo seremos eliminados	perd jog ser elimin
7	valor novo carro subiu	val nov carr sub
8	perdi novamente aposta	perd nov apost
9	chove perigoso dirigindo	chov perig dirig

Ao analisar a tabela 2, se percebe por exemplo que as palavras “perdi” e “perdemos” das frases 5 e 6, foram atribuídas ao radical "perd", ou seja ambas as palavras possuem o mesmo sentido e a aplicação do algoritmo reduz consideravelmente a quantidade de palavras a serem processadas posteriormente.

2.4.3. Mineração e Análise dos dados

Na Mineração são aplicadas técnicas direcionadas ao aprendizado de máquina (*Machine Learning* - ML) para obtenção de novos conhecimentos (Teixeira, 2018).

Já na etapa de assimilação dos dados, será validada a eficiência do processo como um todo, analisando os dados obtidos após aplicação dos algoritmos na etapa anterior. Ou seja, é nesta etapa que é avaliado se o objetivo de descobrir um novo conhecimento foi adquirido, a partir de uma base de dados (Brito, 2016).

2.5. Análise de Sentimentos

A análise de Sentimentos, também conhecida como Mineração de Opinião, corresponde ao problema de identificar emoções (opiniões em textos) (Brito, 2016).

A análise de sentimentos trata de problemas de classificação e é utilizada para classificar textos de acordo com a sua polaridade, mesmo que uma frase não denote explicitamente um sentimento. A frase "Jovem com suspeita de Dengue morre em hospital" apenas descreve um fato, no entanto, poderá ser classificada como positiva ou negativa para a área da saúde (Brito, 2016).

Nem só de textos é possível extrair a polaridade, talvez o jeito mais simples de identificar a polaridade de uma mensagem seja baseado na análise de *emoticons* (ícones ou sequência de caracteres que transmitem o estado emotivo da mensagem) (Araújo, 2013).

A Tabela 3 mostra alguns *emoticons* disponíveis e sua polaridade sejam eles positivos, negativos ou neutros.

Tabela 3: *Emoticons* e suas variações (Araújo, 2013).

Emoticon	Polaridade	Símbolos
	Positivo	:) :] :} :o) :o] :o} :-] :-) :-} =) =] =} =^] =^) =^} :B :-D :-B :^D :^B :^D :^B =B =^B =^D :') :'] :'} =') ='] ='} <3 ^.^ ^-^ ^_^ ^^ :* =* :-* ;) ;] ;} :-p :-P :-b :^p :^P :^b =P =p \o\ /o/ :P :p :b =b =^p =^P =^b \o/

	Negativo	D: D= D-: D^: D^= :(:[:{ :o(
		:o[:^(:^[:^{ =^(=^{ >=(>=[
		>={ >=(>:-{ >:-[>:-(>=^[>:-(
		
		:-[:-(=(=[={ =^[>:-=(>=[
		:'(:'[:'{ ='{ ='(='[=\ :\
		=/ :/ =$ o.O O_o Oo :$:-{ >:-{
		>=^(>=^{ :o{

	Neutro	:| =| :-| >.< >< >_< :o :0 =O :@ =@ :^o :^@ -.- -.-' -_- -_-' :x =X =# :-x :-@ :-# :^x :^# :#

2.5.1. *Naive* Bayes

Naive Bayes é um método de aprendizagem probabilística baseado no teorema de Bayes (França, 2014). Ele utiliza dados de treino para formar um modelo probabilístico baseado nas características (*features*) do dado. O *Naive* Bayes supõe que há uma independência entre as *features* do modelo. Ou seja, o classificador assume que a presença das *features* não tem nenhuma relação entre elas (França, 2014).

A figura 4 demonstra a fórmula do *Naive* Bayes, onde "*c"* representa o evento observado, "*x"* a evidencia, P[*c*] é a probabilidade do evento antes da evidencia ser vista e P[*c* | *x*] é a probabilidade de "*c"* dado o evento "*x"* (ou seja, a probabilidade de "*c"* após o evento "*x"* ser visto). P[*x*] representa o somatório da iteração de ambas as classes com a estimação da distribuição dos termos sobre as classes.

$$P(c \mid x) = \frac{P(x \mid c)P(c)}{P(x)}$$

Figura 4: *Naive* Bayes (França, 2014).

O algoritmo *Naive* Bayes tem se mostrado um método eficaz de análise. Para utilizá-lo é necessário apenas um pequeno grupo de treino durante a fase de aprendizado (Weiand, 2016).

Naive Bayes será utilizado neste trabalho para realizar a classificação de textos baseada em sentimentos presentes nas mensagens da base coletada.

2.5.2. Natural *Language* Toolkit (NLTK)

NLTK é uma biblioteca (desenvolvida na linguagem de programação Python) para a construção de softwares que visam trabalhar com linguagem humana. Ela é uma das ferramentas para este fim mais utilizada (Weiand, 2016).

A utilização desta biblioteca teve como principal objetivo o auxílio ao desenvolvimento e implementação do algoritmo *Naive* Bayes.

3. Estudo de Caso

Neste capítulo, são apresentados os passos realizados nesta pesquisa.

Na primeira parte, foi utilizado o Google *trends* (ferramenta do Google para análise de buscas em https://trends.google.com.br/trends/?geo=BR).

O objetivo da segunda parte, foi exemplificar meios de se coletar, analisar e extrair sentimentos do Twitter. Para isso, foi utilizado *scripts* em Python para coleta e análise dos Tweets, um software online chamado *iFeel* (www.ifeel.dcc.ufmg.br) e *o Iramuteq* (Ferramenta de análise de texto).

3.1. Google *Trends*

Google *Trends* é uma ferramenta do Google criada em 2015, que mostra dentre outras coisas os termos mais populares buscados em um passado recente. A ferramenta apresenta gráficos com a frequência em que um termo particular é procurado (apresentando possíveis motivos para um aumento ou diminuição do volume de buscas) (Google Trends, 2018).

A ferramenta está disponível em 28 países, incluindo o Brasil e segundo dados da própria empresa, são realizadas 100 bilhões de consultas por mês (Google Trends, 2018).

As figuras 5 e 6 mostram o interesse de busca dentre os presidenciáveis nas eleições 2018, nota-se por exemplo que os candidatos Marina da Silva e Geraldo Alckmin (candidatos com mais intenções de voto na pesquisa do DataFolha do dia 28/09/2018) não aparecem na lista dos mais buscados enquanto o candidato Cabo Daciolo aparece.

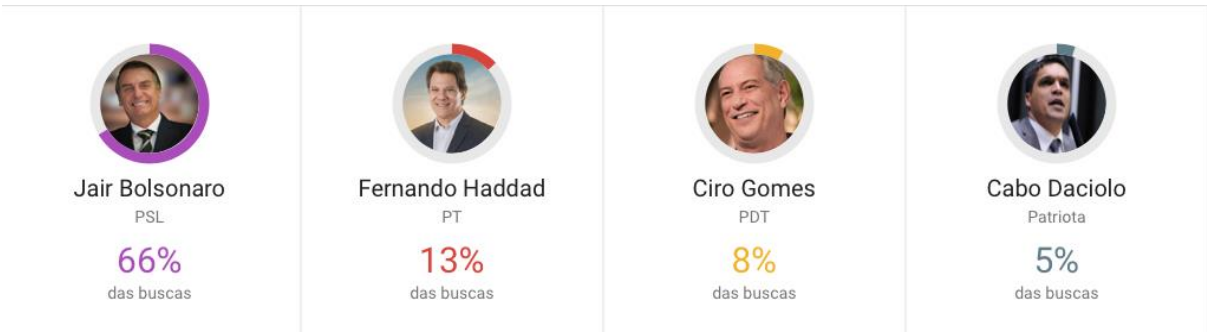

Figura 5: Interesse de busca nos candidatos à presidência no Brasil. Período (22/09/2018 - 29/09/2018).

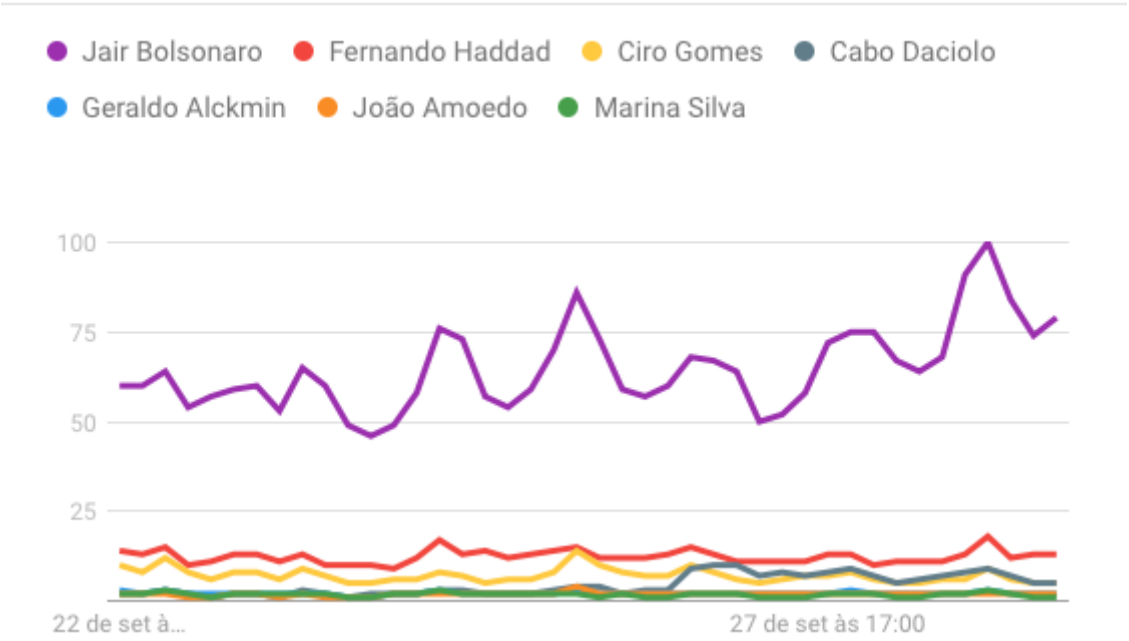

Figura 6: Interesse de busca. Período (22/09/2018 - 29/09/2018).

A seguir, as Figuras 7, 8 e 9 mostram o interesse de Busca por Estado dos três candidatos mais cotados na pesquisa DataFolha do dia 28/09/2018. Nota-se maior interesse em buscas para o Candidato Ciro Gomes no estado do Ceará; um maior interesse em buscas para o candidato Jair Bolsonaro em estados nas regiões Centro-Oeste, Sudeste e Norte e um maior interesse no candidato Fernando Haddad no estado da Bahia. A princípio há um empate nos interesses de busca nos Estados do Piauí e Pernambuco entre os candidatos Jair Bolsonaro e Fernando Haddad.

Figura 7: Interesse de busca, por região candidato Jair Bolsonaro (imagem obtida no dia 29/09/2018).

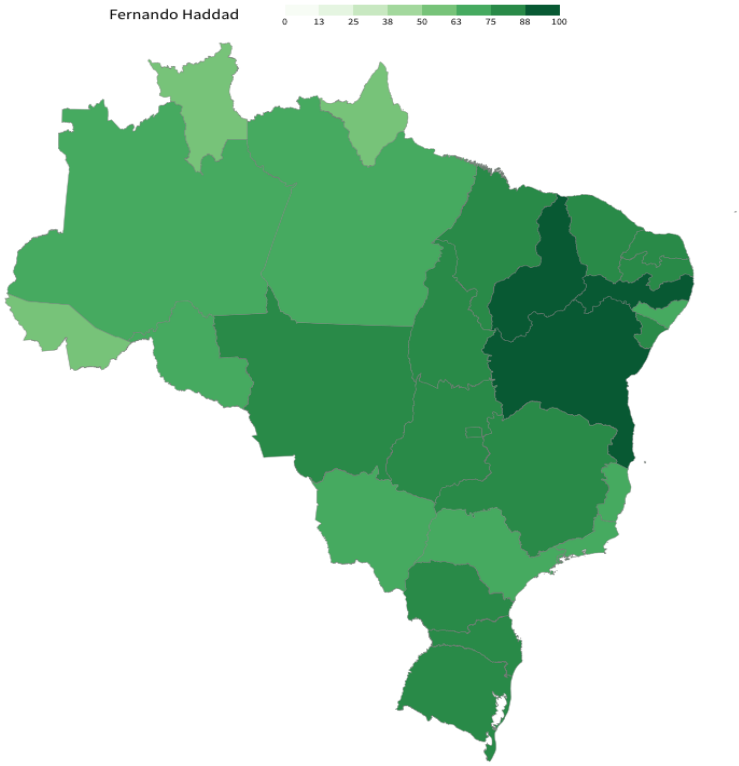

Figura 8: Interesse de busca, por região candidato Fernando Haddad (imagem obtida no dia 29/09/2018).

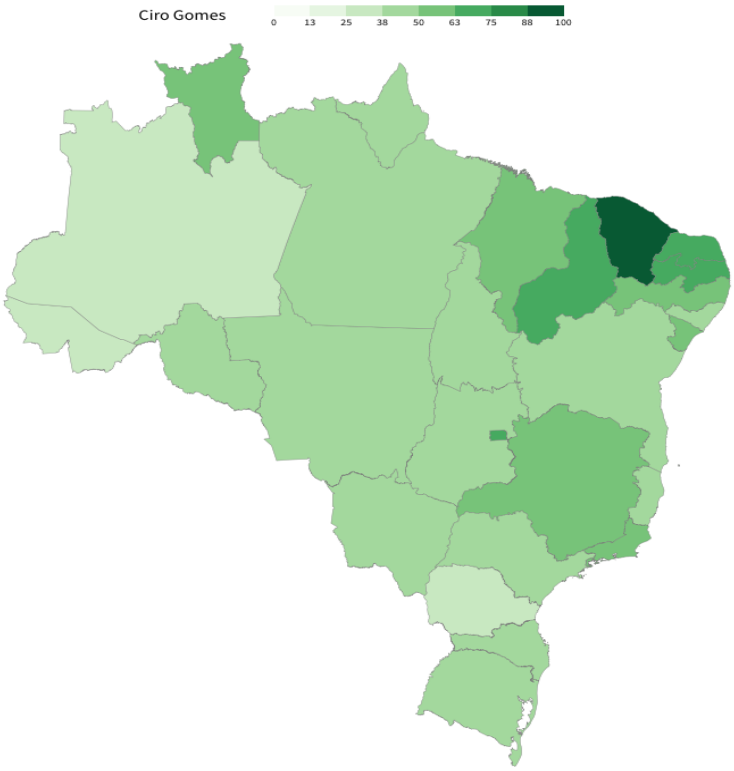

Figura 9: Interesse de busca, por região candidato Ciro Gomes (imagem obtida no dia 29/09/2018).

Considerando os tópicos de interesse mais relevantes à população, pode- se notar que o que mais desperta interesse é a saúde, enquanto um dos menos relevantes é o meio ambiente, dados apresentados na Figura 10.

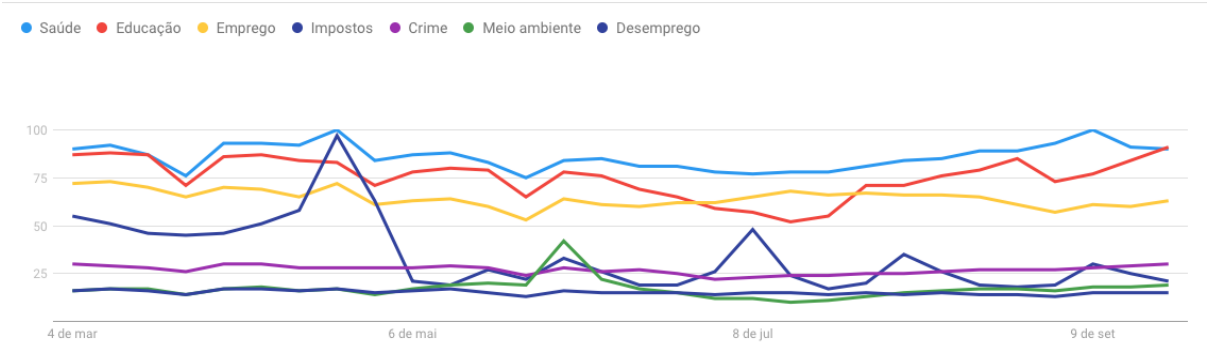

Figura 10: Interesse de busca em tópicos, em 2018.

A Figura 11 mostra as principais pesquisas feitas relacionadas às eleições 2018, nota-se um interesse muito grande em não votar e/ou como anular seu voto. Isto pode sinalizar uma descrença da população com a política no Brasil.

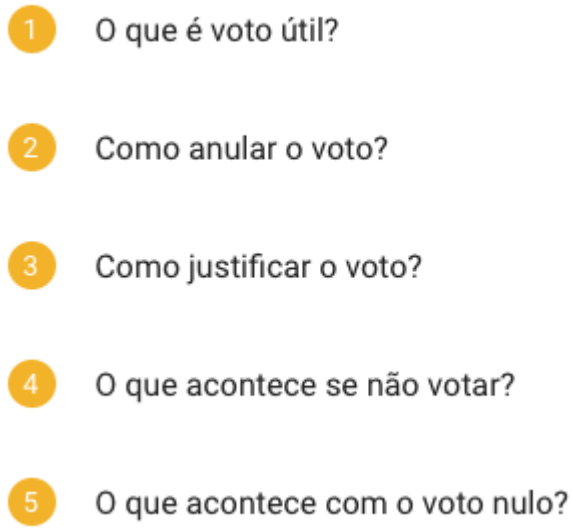

Figura 11: Principais questões sobre o ato de votar. Período (22/09/2018 - 29/09/2018) (imagem obtida no dia 29/09/2018).

A seguir nas figuras 12, 13 e 14 são apresentadas as principais consultas relacionadas aos nomes dos três candidatos mais cotados na pesquisa DataFolha do dia 28/09/2018. Nota-se por exemplo na figura 13 (nas principais consultas relacionadas ao nome do candidato Haddad) um interesse grande com o nome do candidato Bolsonaro, talvez seja porque ambos são os favoritos para vencer a disputa presidencial.

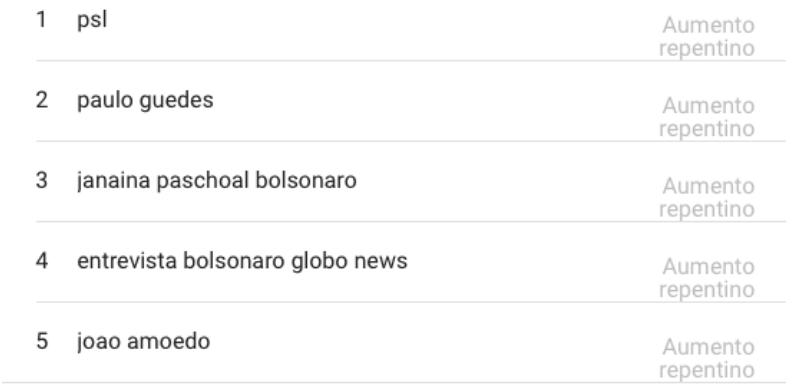

Figura 12: Principais consultas relacionadas ao nome de Jair Bolsonaro (imagem obtida no dia 29/09/2018).

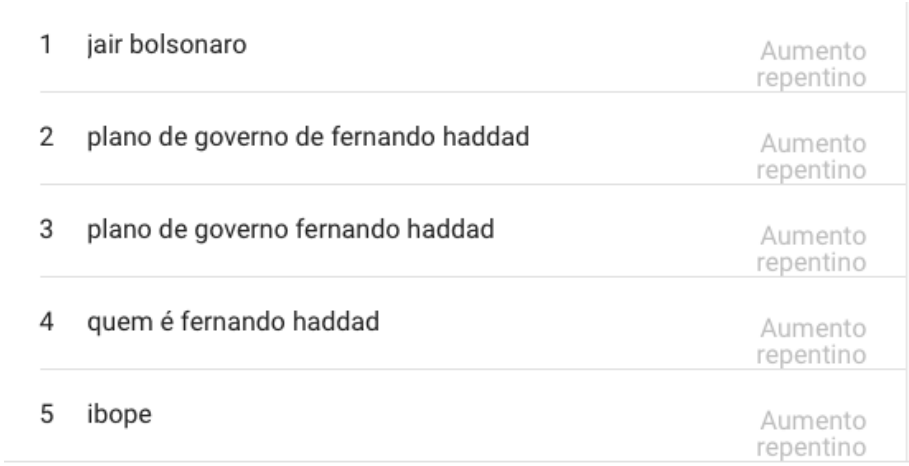

Figura 13: Principais consultas relacionadas ao nome de Fernando Haddad (imagem obtida no dia 29/09/2018).

Figura 14: Principais consultas relacionadas ao nome de Ciro Gomes (imagem obtida no dia 29/09/2018).

É possível notar na figura 15, um aumento na quantidade de consultas feitas com o nome do candidato Jair Bolsonaro no dia 06/09/2018 (dia que o candidato sofreu um atentado em Juiz de Fora/MG).

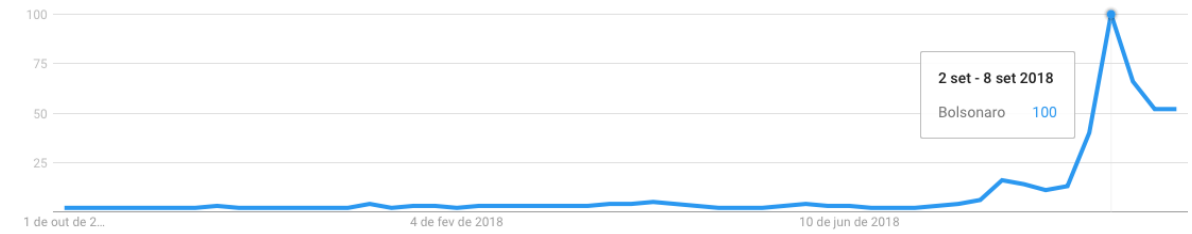

Figura 15: Evolução da quantidade de consultas relacionadas ao nome de Jair Bolsonaro ao longo do tempo (imagem obtida no dia 29/09/2018).

A figura 16 mostra um aumento expressivo no número de consultas feitas com o nome do candidato Fernando Haddad a partir da impugnação da candidatura do ex-presidente Lula à presidência pelo Tribunal Superior Eleitoral no dia 31/08/2018.

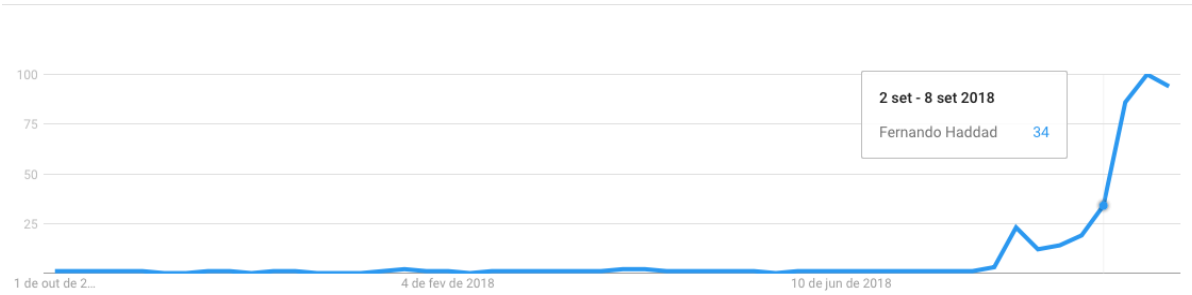

Figura 16: Evolução da quantidade de consultas relacionadas ao nome de Fernando Haddad feitas ao longo do tempo (imagem obtida no dia 29/09/2018).

A figura 17 mostra um grande declínio da quantidade de consultas feitas com o nome do candidato Ciro Gomes a partir do dia 16/09/2018, período que corresponde com o aumento do número de pesquisas relacionadas com o nome do candidato do PT Fernando Haddad. A título de curiosidade, esta movimentação corresponde também com um aumento nas intenções de voto para o candidato Fernando Haddad (e declínio das intenções de voto para o candidato Ciro Gomes) segundo pesquisa DataFolha do dia 28/09/2018 (figura 18).

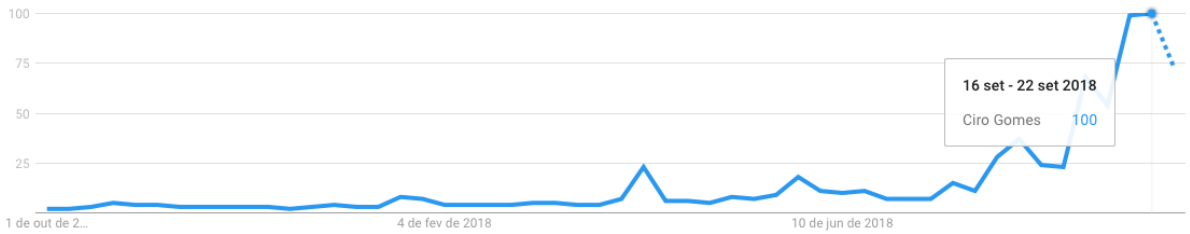

Figura 17: Evolução da quantidade de consultas relacionadas ao nome de Ciro Gomes feitas ao longo do tempo (imagem obtida no dia 29/09/2018).

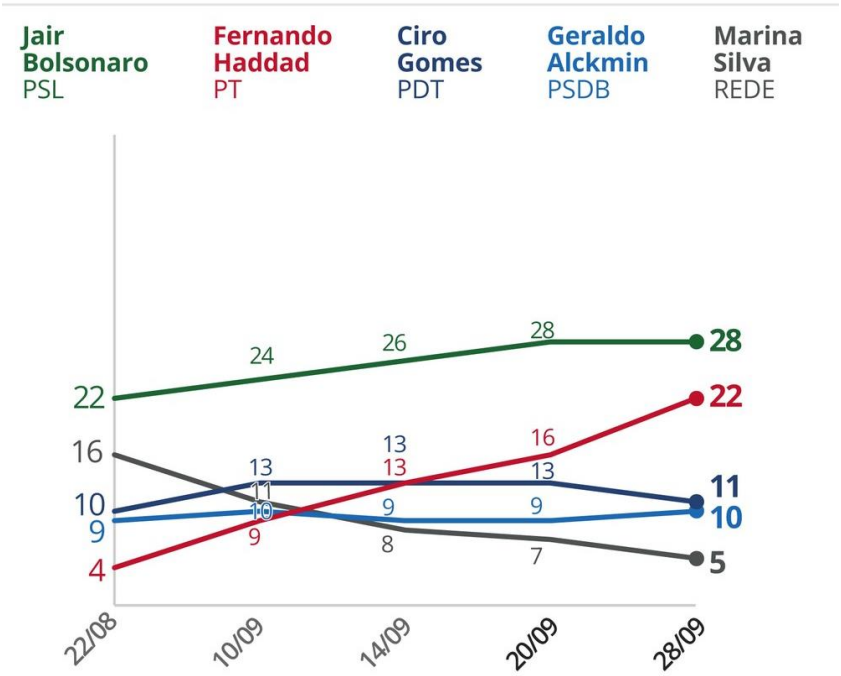

Figura 18: Pesquisa DataFolha para presidência Eleições 2018 (Pesquisa Presidencial globo.com, 28/09/2018).

3.2. Análise Sentimento Twitter

3.2.1. Tecnologias Utilizadas

A seguir são descritas as ferramentas utilizadas nesta etapa do trabalho.

3.2.1.1. Python

Os scripts foram codificados utilizando a linguagem de programação Python (versão 3.6.0).

Python possui diversos módulos e plataformas que são voltados para a mineração de textos e dados, como por exemplo o *Natural Language Toolkit* (NLTK), descrito no capítulo 2. Dessa forma é muito utilizado pelos cientistas de dados como a linguagem de programação favorita para este tipo de tarefa, pois prioriza a legibilidade do código sobre a velocidade ou expressividade.

3.2.1.2. Anaconda

Foi utilizado o Anaconda (em sua versão 4.5.4) com o objetivo de simplificar o gerenciamento e a implantação de pacotes das bibliotecas utilizadas no Python.

3.2.1.3. Atom

Atom foi usado como IDE, pois é um editor de texto de código aberto disponível para todas as plataformas de SO (Sistemas Operacionais), suporta uma variedade de linguagens de programação bem grande, além de ser leve e de fácil manuseio.

3.2.1.4. IRaMuTeQ

IRaMuTeQ é um software livre lançada em 2008, foi desenvolvido pelo *Laboratoire d'Études et de Recherches Appliquées en Sciences Sociales* (LERASS) da Universidade de Toulouse. O software é ligado ao R (software estatístico) para análise de dados textuais (IRaMuTeQ, 2018).

A ferramenta foi usada na pesquisa como auxiliar para análises gráficas dos resultados.

3.3. Desenvolvimento

Nessa seção será abordado o processo da análise dos *tweets*.

3.3.1. Extração de *Tweets*

O primeiro passo foi criar um aplicativo no Twitter Apps. Isso permitiu autenticar o Twitter e utilizar a API.

As Figuras 19 e 20 mostram a geração das chaves dentro do Twitter *Application Manager*. As credenciais geradas são: *Consumer Key* (identifica o cliente (site / serviço) que está tentando acessar os recursos de um usuário final); *Consumer Secret* (senha do cliente usada para autenticar com o servidor); *Access Token* (emitido para o cliente quando o mesmo se autentica com sucesso) e *Access Token Secret* (enviado com o *token* de acesso como se fosse uma senha). Feito isso, pode- se realizar a coleta de dados para a mineração de texto.

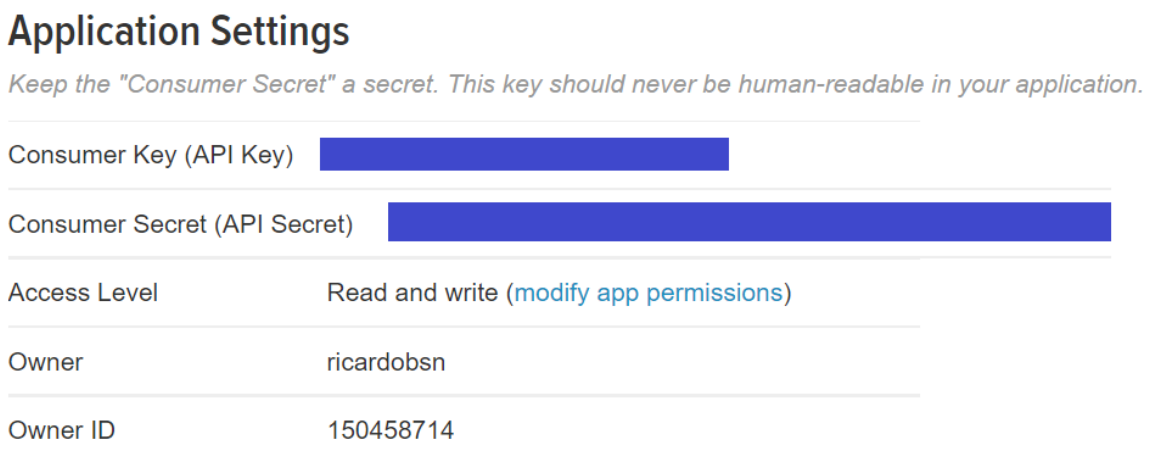

Figura 19: Mostra a geração do *Consumer Key* (API Key) e *Consumer Secret* (API *Secret*).

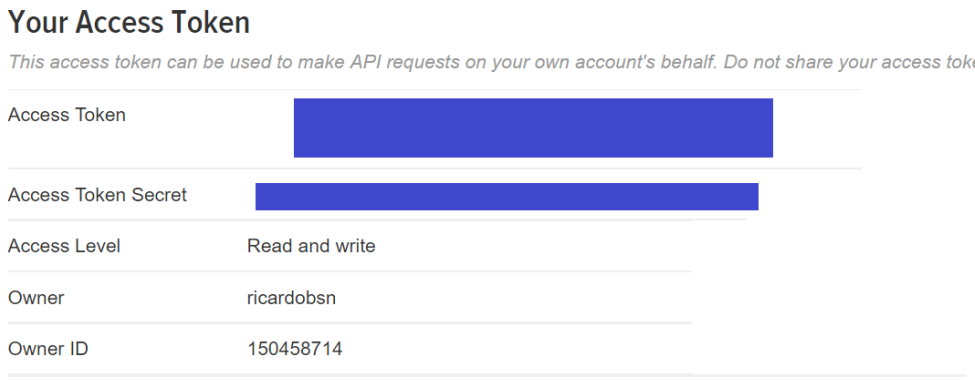

Figura 20: Mostra a geração do *Access Token* e *Access Token Secret*.

Após a geração das chaves, foram instaladas as bibliotecas necessárias para o projeto. Foram usadas na análise as bibliotecas TextBlob (processamento de texto) e Tweepy (para coleta dos dados).

Feito a conexão com a API, pode- se começar a coleta dos dados. A figura 21 é observado o *script* responsável pela extração, na qual ele primeiramente autentica as credenciais do usuário e é na função "*filter Track*" que é feito a busca pelo parâmetro desejado (Moujahid, 2014).

```
# -*- coding: utf-8 -*-
from tweepy.streaming import StreamListener
from tweepy import OAuthHandler
from tweepy import Stream

#vaviaveis com as credenciais de acesso
access_token = "[redacted]"
access_token_secret = "[redacted]"
consumer_key = "[redacted]"
consumer_secret = "[redacted]"


#faz um print em stdout dos tweets
class StdOutListener(StreamListener):

    def on_data(self, data):
        print data
        return True

    def on_error(self, status):
        print status


if __name__ == '__main__':
    #Esta parte do codigo cuida do acesso e autenticaçao da API do Twitter
    l = StdOutListener()
    auth = OAuthHandler(consumer_key, consumer_secret)
    auth.set_access_token(access_token, access_token_secret)
    stream = Stream(auth, l)

    #Esta linha filtra tweets com essa pesquisa
    stream.filter(track=['Haddad'])
```

Figura 21: *Script* responsável para coleta dos *tweets* em tempo real com o nome do Candidato Fernando Haddad (Moujahid, 2014).

Duas pesquisas distintas foram realizadas no dia 03/10/2018, em cada uma delas foi usado os nomes dos dois principais candidatos segundo a pesquisa DataFolha do dia 28/09/2018 para a disputa presidencial (Jair Bolsonaro e Fernando Haddad). Para ter um comparativo de qual dos dois estava sendo mais comentado (independente de forma positiva ou negativa), foi realizado uma coleta em tempo real pelo mesmo período de tempo para ambos (uma hora para cada). Vale ressaltar que as pesquisas foram feitas em sequência, ou seja primeiro foi coletado os *tweets* relacionados com o nome do candidato Bolsonaro e em seguida foi coletado os *tweets* relacionados com o nome do candidato Haddad.

Durante esse período de tempo, foram coletados, na pesquisa "Bolsonaro", 109.589 (cento e nove mil quinhentos e oitenta e nove) *tweets*, enquanto na pesquisa "Haddad" foram coletados 108.792 (cento e oito mil setecentos e noventa e dois) *tweets*, portanto, foi

praticamente coletada a mesma quantidade de *tweets* em ambas as pesquisas. Os resultados das pesquisas foram salvos em um arquivo no formato txt.

Deste montante, 70% (aproximadamente setenta mil *tweets*) foram usados para o corpus de teste (os primeiros setenta mil da base) enquanto aproximadamente 30% (o restante da base, aproximadamente quarenta mil *tweets*) para análise da acurácia do algoritmo. Desse modo é possível mensurar, caso seja necessário uma outra análise usando uma outra base quanto em média seria a margem de erro.

Os dados coletados vêm em formato JSON (Segundo *Mozilla Developer* em 2018, JSON possui formato compacto, de padrão aberto independente, para troca de dados simples e rápida entre sistemas) conforme a figura 22.

Figura 22: Exemplo do JSON de um dos *tweets* coletados.

A função responsável pela contagem dos *tweets* coletados é apresentada na figura 23, em que cada linha corresponde a um JSON e a função faz um *loop* em todo o arquivo, realizando a contagem e mostrando o resultado final (conforme a figura 24) (Moujahid, 2014).

```
tweets_data_path = 'C:/Users/ricardo.neves/Documents/sentiments_analyst-master/sentiments_analyst-master/

tweets_data = []
tweets_file = open(tweets_data_path, "r")
for line in tweets_file:
    try:
        tweet = json.loads(line)
        tweets_data.append(tweet)
    except:
        continue

print len(tweets_data)
```

Figura 23: Função responsável pela contagem total dos *tweets* (Moujahid, 2014).

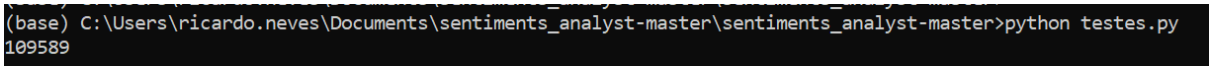

Figura 24: Quantidade total dos *tweets* coletados para o candidato Jair Bolsonaro.

3.3.2. Análises Realizadas

Coleta dos dados concluída, começou a análise propriamente dita. O primeiro script rodado, foi feito usando como base scripts retirados do GitHub (os scripts originais se encontram no Anexo A), o mesmo usa como algoritmo de classificação o NLTK, que utiliza o classificador *Naive* Bayes e tem como objetivo fazer uma análise de sentimento onde o ranking vai de - 5 (sentimento mais negativo) a + 5 (sentimento mais positivo), o script utiliza como fonte, um arquivo anexo (espécie de dicionário) com milhares de palavras e para cada uma delas uma "nota" é dada, o idioma originalmente era o inglês, mas como os dados a serem analisados eram em português, o dicionário de palavras foi adaptado para o idioma, gerando conflitos de tradução que tiveram que ser sanados.

Antes de executar o algoritmo, foi necessário realizar a limpeza do texto, onde cada registro irá passar pela função de pré-processamento, que irá retornar o texto *tokenizado*, sem as *stopwords* e *stemitizado*, esse resultado que irá passar pelas funções descritas a seguir para análise.

A função responsável pela iteração dos *tweets* é apresentada na figura 25, em que ela carrega todas as linhas (cada linha é um JSON), busca cada palavra de cada JSON na lista de palavras (conforme figura 26), caso encontre, a nota correspondente é atribuída, caso contrário atribui 0 (zero), em seguida, será atribuída uma nota final para cada *Tweet* (média das notas de todas palavras encontradas no *tweet*) (GitHub, 2018). Cada nota é armazenada em um *array* e ao final do processamento, as notas são somadas e é feita uma média de todas as notas (conforme apresentado na figura 27).

```
for line in tweetfile:

    if line != "\n":
        #pdb.set_trace()
        line.replace("\r", "")
        parsed_json = json.loads(line)
        # Inicializa a análise de sentimentos
        sentiment_value = 0
```

Figura 25: Parte da função apresentada na figura 27 responsável de carregar todas as linhas do arquivo (GitHub, 2018).

```
def tweet_sentiment_value(tweet_text,sent_dictionary):
    sent_value = 0

    for word in tweet_text.split():

        if word in sent_dictionary.keys():

            sent_value += sent_dictionary[word]
    return sent_value
```

Figura 26: Função responsável por buscar cada palavra encontrada em cada *tweet* no banco de palavras e atribuindo (caso encontre) o valor correspondente (GitHub, 2018).

```
def process_tweets(tweet_file,afinn_dict):
    tweetfile = open(tweet_file)
    scores = {}
    count_tweet = 0
    TodosSentimentos = []
    for line in tweetfile:
        if line != "\n":
            #pdb.set_trace()
            line.replace("\r", "")
            parsed_json = json.loads(line)
            # Inicializa a análise de sentimentos
            sentiment_value = 0
        # Checa se o testo foi escrito em português
        if ('text' in parsed_json.keys()) and ('lang' in parsed_json.keys()) and (parsed_json['lang'] == 'pt'):
            unicode_string = parsed_json['text']
            encoded_string = unicode_string.encode('utf-8')
            encoded_string = re.sub(r'[^a-z0-9\s]', '', encoded_string.strip().lower())
            # busca a string na função que busca a palavra no dicionario
            sentiment_value = tweet_sentiment_value(encoded_string,afinn_dict)
            # armazena o valor total de cada tweet em um array
            TodosSentimentos.append(sentiment_value)
            #printa o valor de cada tweet
            print sentiment_value
        else:
            print sentiment_value
    soma_das_notas = sum(TodosSentimentos)
    qtd_de_notas = len(TodosSentimentos)
    media_das_notas = soma_das_notas / qtd_de_notas
    print("Média do Sentimento %s" % (media_das_notas))
```

Figura 27: Função responsável por fazer a soma de todas as notas dos sentimentos de todos *tweets* e mostra a média final.

Como resultado desta primeira análise em cima da base de teste (aproximadamente 70% do total dos *tweets*), foi obtido uma nota -1 (a quantidade de pessoas com sentimento negativo supera com uma margem pequena as pessoas com sentimento positivo) para o candidato Jair Bolsonaro conforme a figura 28 e zero (é equilibrado a quantidade de pessoas com sentimento positivo e negativo) para o candidato Fernando Haddad conforme a figura 29.

```
-3
-3
0
0
-1
-1
MÃ©dia do Sentimento -1
```

Figura 28: Resultado da Análise de sentimento do candidato Jair Bolsonaro (incluindo últimas notas obtidas).

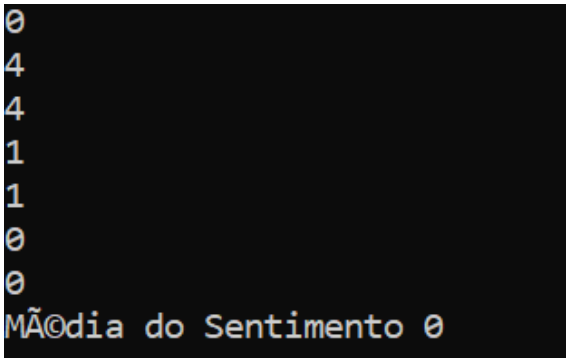

Figura 29: Resultado da Análise de sentimento do candidato Fernando Haddad (incluindo últimas notas obtidas).

Para confirmar a análise, foi feito uma contraprova usando outro script. A diferença é que desta vez, os índices variaram entre -1 (menos um) e +1 (mais um). Este script utiliza a biblioteca TextBlob para realizar a análise. A função responsável é mostrada na figura 30 (Vasconcelos, 2018).

Ao rodar o *script* ele apresenta na tela os *tweets* daquele momento, faz a análise e entrega o resultado final da análise de sentimentos (Vasconcelos, 2018).

Nesta segunda análise, o candidato Jair Bolsonaro obteve uma nota aproximada de -0.0033 conforme figura 31 e o candidato Fernando Haddad uma nota aproximada de -0.066 conforme figura 32. Ou seja, ambos os candidatos receberam mais comentários negativos à positivos, porém o candidato Fernando Haddad recebeu na média um pouco mais de comentários positivos em relação ao candidato Jair Bolsonaro.

```python
from TwitterSearch import *
try:

    ts = TwitterSearch(
        consumer_key = [redacted],
        consumer_secret = [redacted],
        access_token = [redacted],
        access_token_secret = [redacted]
     )
    tso = TwitterSearchOrder()
    tso.set_keywords(['Bolsonaro'])
    tso.set_language('pt')
    for tweet in ts.search_tweets_iterable(tso):
        print( '@%s tweeted: %s' % ( tweet['user']['screen_name'], tweet['text'] ) )

except TwitterSearchException as e:
    print(e)
tweets = [] # Lista vazia para armazenar scores
for tweet in public_tweets:
    print(tweet.text)
    analysis = tb(tweet.text)
    polarity = analysis.sentiment.polarity
    tweets.append(polarity)
    print(polarity)
print('MÉDIA DE SENTIMENTO: ' + str(np.mean(tweets)))
```

Figura 30: Script responsável por fazer a segunda análise de sentimentos (Vasconcelos, 2018).

```
0.0
RT @Vagazoide: Só queria falar que eu vou passar o pinto no numero 7 da urna entao quem apertar 17 pra votar no bolsonaro vai ta tocando in…
0.0
RT @doisdedosdeteo: O politicamente correto do conservadorismo brasileiro

1. Louve Olavo como deus
2. Trate Bolsonaro como político modelo…
0.05
RT @MPF_MG: MPF denuncia agressor de Bolsonaro por crime contra a segurança nacional https://t.co/mPsdhGNkBN
0.0
RT @mah_13_: “A campanha petista diagnosticou bem onde Bolsonaro cresce: na podridão das correntes de WhatsApp.

É um espaço imune a fiscal…
0.0
RT @OenningDuda: maconheiro q apoia o bolsonaro kk
0.0
- Presidente: Bolsonaro
- Governador: Sartori
- Senadores: Heinze e Carmen Flores
- Deputado federal: Marcel Van Ha… https://t.co/3SLvaeJMXE
0.0
RT @jornal_asemana: Até o mercado financeiro quer o Bolsonaro como presidente.
Só a turma do elenão não quer enxergar o que é melhor para o…
0.0
RT @eipedroxavier: Votar no Bolsonaro por causa do porte kkkkkkkkkk     gnt uma arma custa 10k, vcs nem nome limpo tem kkkkkkkk
0.0
RT @claudioedantas: Ciro desafia Bolsonaro a ir a debate: “Atestado médico falso é crime” https://t.co/8ufGnzqfMI
0.0
Se o Bolsonaro for eleito vou culpar todos vocês que fazem propaganda de graça pra ele usando o nome dele na foto de perfil do Facebook.
-0.1
Atriz Luma Costa chama mulheres que protestaram contra Bolsonaro de ‘burras’ https://t.co/4Mb1UKAOpP
0.0
MÃ‰DIA DE SENTIMENTO: -0.0033333333333333335
```

Figura 31: Resultado da segunda Análise de sentimento do candidato Jair Bolsonaro (incluindo últimos *tweets* analisados).

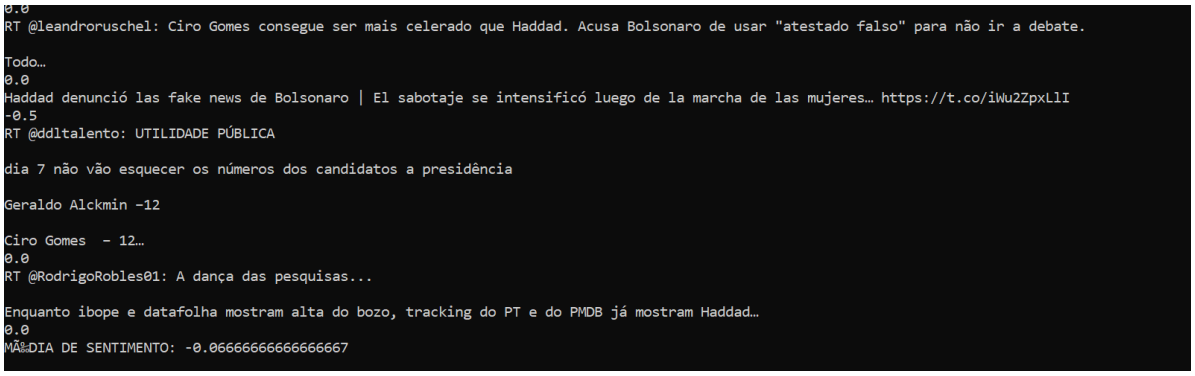

Figura 32: Resultado da segunda Análise de sentimento do candidato Fernando Haddad (incluindo últimos *tweets* analisados).

Análise de sentimentos concluída, foi feito um estudo dos termos mais frequentes de ambos os candidatos. Para isso, foi usada a função descrita na figura 33 em que ela conta a quantidade que cada palavra aparece no texto e divide pela quantidade total de palavras (GitHub, 2018). Como o texto era muito grande (a título de curiosidade cada arquivo possuía mais de 600 MB), a fração do resultado final tinha precisão de quatro casas decimais. A figura 34 mostra parte do resultado obtido.

```python
def get_freq_tweets(tweet_file):
    tweetfile = open(tweet_file)
    frequencia_dicionario = {}
    contador_palavra = 0
    for line in tweetfile:
        try:
            parsed_json = json.loads(line)
        except ValueError:
            pass
        if ('text' in parsed_json.keys()) and ('lang' in parsed_json.keys()) and (parsed_json['lang'] == 'pt'):
            unicode_string = parsed_json['text']
            encoded_string = unicode_string.encode('utf-8')
            encoded_string = re.sub(r'[^a-z0-9\s]', '', encoded_string.strip().lower())
            # Loop dentro das palavras
            for word in encoded_string.split():
                # checa se a palavra já esta na lista
                # filtra palavras q não possuem https
                if ( word in frequencia_dicionario.keys() ) and ('https' not in word):
                    # incrementa o contador da palavra
                    frequencia_dicionario[word] +=1
                    contador_palavra += 1
                # se a palavra ainda não esta na lista, é colocada
                elif (word not in frequencia_dicionario.keys()) and ('https' not in word):
                    frequencia_dicionario[word] = 1
                    contador_palavra += 1
    # com a lista de palavras completa
    for word in frequencia_dicionario.keys():
        #calcula a porcentagem de cada palavra no contexto total com a precisão de 4 casas decimais
        pct = float(frequencia_dicionario[word])/contador_palavra
        # printa apenas se a porcentagem for significativa
        if (pct > 0.0000):
            print "%s %.4f" % (word, pct)
    # Retorna a palavra e a frequencia
    return contador_palavra , frequencia_dicionario
```

Figura 33: Função responsável pela contagem e média de cada palavra encontrada dentro do contexto total (GitHub, 2018).

```
lembra 0.0027
aloprao 0.0000
radiantes 0.0000
apoiaro 0.0000
padre 0.0000
11 0.0006
10 0.0007
13 0.0010
12 0.0118
15 0.0001
14 0.0001
17 0.0003
```

Figura 34: Fração da quantidade de parte das palavras encontradas na pesquisa do candidato Fernando Haddad.

Com a finalidade de deixar o resultado mais legível, foi realizada uma limpeza prévia das palavras encontradas sem muito significado (verbos soltos, palavras sem sentido, etc..) e uma nuvem de palavras com os termos que mais apareceram nos *tweets* foi gerada. A ferramenta IRaMuTeQ foi utilizada para esse objetivo (na figura 35 pode ser observada a

interface do software). As figuras 36 e 37 apresentam as nuvens de palavras de cada candidato, quanto maior a palavra, maior a frequência que ela aparece nos *tweets*. Em ambas as nuvens nota-se uma divisão muito grande entre palavras positivas e negativas, com frequências muito parecidas, resultado da grande polarização dos votos para ambos candidatos.

Figura 35: Interface do IRaMuTeQ (Iramuteq, 2018).

Figura 36: Nuvem de palavras do candidato Jair Bolsonaro com as palavras mais frequentes (Fonte do autor).

Figura 37: Nuvem de palavras do candidato Fernando Haddad com as palavras mais frequentes
(Fonte do autor).

3.3.3. Ferramenta iFeel (Análise Complementar)

O serviço online iFeel criado na Universidade Federal de Minas Gerais e disponibilizado em www.ifeel.dcc.ufmg.br facilita o acesso a diversos métodos de análise de sentimentos. Portanto uma ferramenta bem útil não só para as pessoas que gostariam de ter contato com análise de sentimento (mesmo sem conhecimento na área), mas também para ajudar pesquisadores que pretendem avaliar novos métodos em uma única plataforma (Araújo, 2013).

A ferramenta é bem simples de usar. A figura 38 apresenta a interface. É possível analisar um arquivo de texto (resultado é exportado como uma planilha Excel), ou é possível ainda escrever um texto e processá-lo.

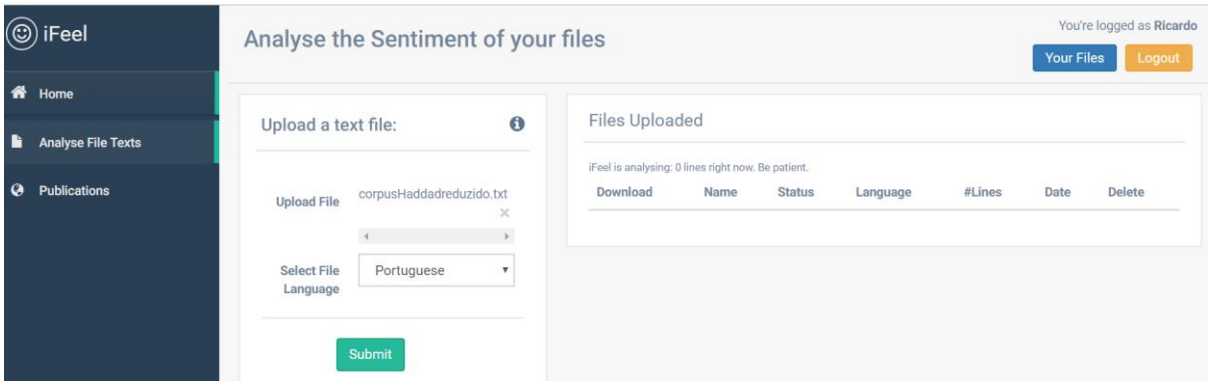

Figura 38: Interface da ferramenta iFeel (iFeel, 2018).

Ao enviar um arquivo, a ferramenta processa todas as linhas conforme figura 39 e o resultado é uma planilha Excel em que para cada linha do arquivo são feitas todas as análises disponíveis na ferramenta e é dado um veredicto se o sentimento foi negativo, neutro ou positivo. A figura 40 mostra parte do resultado obtido.

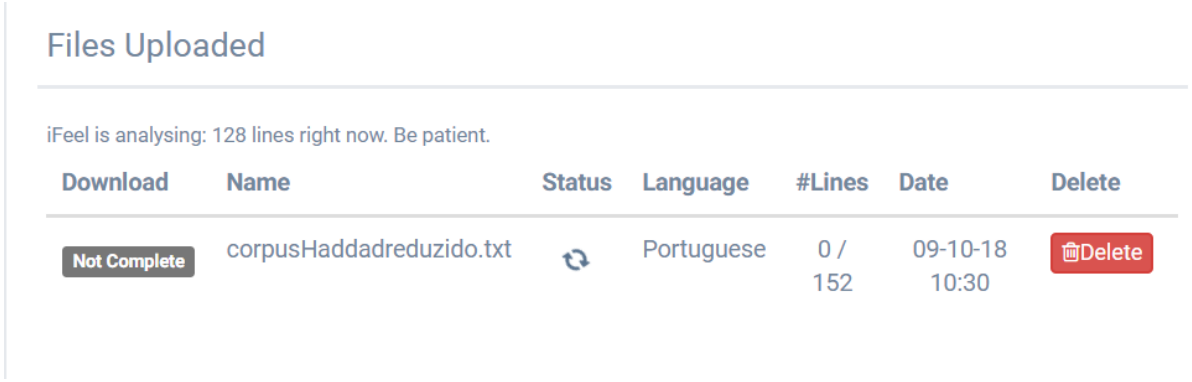

Figura 39: Processamento do texto enviado.

@RaposaoDoPovo tweeted: @	Negative	Negative	Negative	Neutral	Positive	Positive	Neutral	Negative	Negative	Negative	Negative	Neutral	Neutral	Neutral	Neutral
@betobraga22 tweeted: RT @	Neutral	Neutral	Positive	Positive	Positive	Positive	Positive	Negative	Negative	Neutral	Negative	Negative	Neutral	Neutral	Neutral
@uesleycosta tweeted: RT @b	Neutral	Neutral	Positive	Positive	Positive	Positive	Positive	Negative	Negative	Neutral	Negative	Negative	Neutral	Neutral	Neutral
@Antonio_Gava tweeted: RT @	Neutral	Neutral	Neutral	Positive	Positive	Positive	Positive	Negative	Neutral	Neutral	Positive	Positive	Neutral	Neutral	Negative
@AlanBrado42 tweeted: @Mr	Negative	Neutral	Negative	Positive	Negative	Positive	Negative	Negative	Negative	Neutral	Negative	Neutral	Neutral	Neutral	Negative
@Moura_ias08 tweeted: RT @	Neutral	Neutral	Neutral	Neutral	Positive	Positive	Positive	Negative	Neutral	Neutral	Negative	Neutral	Neutral	Neutral	Neutral
@HelioMedeiros16 tweeted: R	Positive	Neutral	Negative	Neutral	Positive	Positive	Positive	Negative	Positive	Negative	Positive	Negative	Neutral	Neutral	Neutral
	Neutral	Neutral	Neutral	Neutral	Neutral	Neutral	Neutral	Neutral	Neutral	Neutral	Neutral	Neutral	Neutral	Neutral	Neutral
O Jair Bolsonaro ficou em prim	Neutral	Negative	Neutral	Neutral	Positive	Positive	Negative	Negative	Negative	Neutral	Negative	Negative	Neutral	Neutral	Neutral
@msarsur tweeted: @claritia \	Positive	Neutral	Positive	Positive	Positive	Positive	Negative	Negative	Positive	Positive	Negative	Negative	Neutral	Neutral	Neutral
@fogao1910 tweeted: RT @be	Neutral	Neutral	Positive	Positive	Positive	Positive	Positive	Negative	Negative	Neutral	Negative	Negative	Neutral	Neutral	Neutral

Figura 40: Parte do resultado da ferramenta iFeel (envio de arquivo .txt).

Conforme dito anteriormente, a ferramenta também disponibiliza um modo de análise, em que é possível avaliar um texto ao invés do envio de um arquivo.

Foram realizadas quatro análises para testar a acurácia desse módulo da ferramenta. Dentre todos os *tweets*, foram selecionados dois *tweets* (cada um falando bem de um dos candidatos) e outros dois *tweets* (cada um falando mal de um dos candidatos). As Figuras 41, 42, 43 e 44 mostram os testes obtidos.

O cabeçalho contém o texto analisado e as colunas são (da esquerda para direita): o nome do teste realizado, status da análise, nota para o teste e a polaridade.

Vale ressaltar que não foram escolhidos *tweets* em que o contexto é óbvio (com palavras que denotariam sentimento positivo e/ou negativo), mas sim *tweets* onde era necessário ler a frase e entender o contexto.

A figura 41 apresenta a análise feita do seguinte *tweet*: "A solução de Haddad contra a violência não é punir criminosos e sim soltar! Claro que isso vai resolver o problema da violência". Lendo o texto, nota-se um tom negativo, mas para o seu entendimento, é preciso ler e compreender o contexto, pois o mesmo a princípio não contém palavras com tom negativo, além de fazer uso de ironia. Mesmo assim a ferramenta identificou esses padrões e na maioria dos testes realizados classificou o texto com uma polaridade negativa.

Your input: A solução de Haddad contra a violência não é punir criminosos e sim soltar! Claro que isso vai resolver o problema da violência

Method Name	Status	Method Score	Polarity
OPINIONLEXICON	Completed	-1	Negative
SENTISTRENGTH	Completed	-0.75	Negative
SOCAL	Completed	-2.5	Negative
HAPPINESSINDEX	Completed	0	Neutral
SANN	Completed	-1	Negative
EMOTICONSDS	Completed	1	Positive
SENTIMENT140	Completed	-38.94799999999999	Negative
STANFORD	Completed	-1	Negative
AFINN	Completed	-1.2222222222222223	Negative
MPQA	Completed	-1	Negative
NRCHASHTAG	Completed	-22.094000000000005	Negative
EMOLEX	Completed	-1	Negative
EMOTICONS	Completed	0	Neutral
PANAST	Completed	0	Neutral
SASA	Completed	1	Positive
SENTIWORDNET	Completed	-1.0760990080970267	Negative
VADER	Completed	0	Neutral
UMIGON	Completed	1	Positive

Figura 41: Análise da frase candidato Fernando Haddad com um contexto negativo.

Seguindo a linha de pensamento da figura 41, a figura 42 mostra a análise feita da mensagem: "a corja da hipocrisia e (sic) grande, conheço 2 que foram funcionários fantasmas

e votam em (sic) Bolsonaro haha". O texto apresenta uma conotação negativa, mas não possui palavras que expressam sentimento positivo e/ou negativo (além de conter linguagem coloquial com a palavra "fantasmas"). A ferramenta dessa vez não identificou muito bem esses padrões e na maioria dos testes realizados classificou o texto com uma polaridade neutra.

Your input: a corja da hipocrisia e grande, conheço 2 que foram funcionários fantasmas e votam em bolsonaro haha

Method Name	Status	Method Score	Polarity
OPINIONLEXICON	Completed	0	Neutral
SENTISTRENGTH	Completed	0.25	Positive
SOCAL	Completed	0	Neutral
HAPPINESSINDEX	Completed	0	Neutral
SANN	Completed	0	Neutral
EMOTICONSDS	Completed	1	Positive
SENTIMENT140	Completed	2.0989999999999998	Positive
STANF0RD	Completed	-1	Negative
AFINN	Completed	3	Positive
MPQA	Completed	0	Neutral
NRCHASHTAG	Completed	1.3669999999999993	Positive
EMOLEX	Completed	0	Neutral
EMOTICONS	Completed	0	Neutral
PANAST	Completed	0	Neutral
SASA	Completed	0	Neutral
SENTIWORDNET	Completed	0	Neutral
VADER	Completed	0	Neutral
UMIGON	Completed	1	Positive

Figura 42: Análise da frase candidato Jair Bolsonaro com um contexto negativo.

A figura 43 mostra a análise do *tweet*: "Apoiar Haddad não é ser petista, é ser humano". A escolha desse texto se deu por possuir uma conotação positiva, mas apresenta a palavra "não" e por isso é válido testar a ferramenta e analisar como seria a classificação. A ferramenta não foi pega nessa "armadilha" e de modo geral classificou essa mensagem como neutra e em alguns testes identificou esse padrão e classificou o texto com uma polaridade positiva.

Your input: Apoiar Haddad não é ser petista, é ser humano

Method Name	Status	Method Score	Polarity
OPINIONLEXICON	Completed	0	Neutral
SENTISTRENGTH	Completed	0	Neutral
SOCAL	Completed	0	Neutral
HAPPINESSINDEX	Completed	0	Neutral
SANN	Completed	1	Positive
EMOTICONSDS	Completed	1	Positive
SENTIMENT140	Completed	0	Neutral
STANFORD	Completed	-1	Negative
AFINN	Completed	0	Neutral
MPQA	Completed	0	Neutral
NRCHASHTAG	Completed	3.154	Positive
EMOLEX	Completed	0	Neutral
EMOTICONS	Completed	0	Neutral
PANAST	Completed	0	Neutral
SASA	Completed	0	Neutral
SENTIWORDNET	Completed	0	Neutral
VADER	Completed	0	Neutral
UMIGON	Completed	0	Neutral

Figura 43: Análise da frase candidato Fernando Haddad com um contexto positivo.

A figura 44 mostra a análise feita do *tweet*: “Bolsonaro é o mais votado da história”. O texto apresenta uma conotação positiva, mas também não possui palavras que expressam sentimento positivo e/ou negativo. A ferramenta não identificou muito bem esse contexto e também na maioria das vezes classificou o texto com uma polaridade neutra.

Your input: Bolsonaro é o mais votado da história

Method Name	Status	Method Score	Polarity
OPINIONLEXICON	Completed	0	Neutral
SENTISTRENGTH	Completed	0	Neutral
SOCAL	Completed	0	Neutral
HAPPINESSINDEX	Completed	0	Neutral
SANN	Completed	0	Neutral
EMOTICONSDS	Completed	1	Positive
SENTIMENT140	Completed	1.436	Positive
STANFORD	Completed	-1	Negative
AFINN	Completed	0	Neutral
MPQA	Completed	0	Neutral
NRCHASHTAG	Completed	0	Neutral
EMOLEX	Completed	0	Neutral
EMOTICONS	Completed	0	Neutral
PANAST	Completed	0	Neutral
SASA	Completed	0	Neutral
SENTIWORDNET	Completed	0	Neutral
VADER	Completed	0	Neutral
UMIGON	Completed	0	Neutral

Figura 44: Análise da frase candidato Jair Bolsonaro com um contexto positivo.

A ferramenta de modo geral teve boa performance, visto que mesmo em *tweets* que para a ferramenta poderiam ser contraditórios, ela no pior dos casos classificou a mensagem como neutra, ao invés de fazer uma classificação trocada (positiva para negativa e vice-versa).

3.4. Análise da Acurácia

Foram feitas as mesmas análises, porém dessa vez em cima do restante da base (30%) com aproximadamente 40.000 (quarenta mil *tweets*). Desta vez, foi encontrado para o candidato Jair Bolsonaro um índice de -0,0026 (o resultado em cima da base de controle foi -0,0033), diferença de aproximadamente 20,2%.

Para o candidato Fernando Haddad foi obtido um índice de -0,053 (o resultado em cima da base de controle foi -0,066), diferença de aproximadamente 19,3%.

Ou seja, as mensagens tiveram uma variação em sua polaridade de aproximadamente 20%. Tendo isso em mente, a margem de erro para mais positivo ou mais negativo pode variar dentro desse espectro.

3.5. Análise e Comparação dos Resultados

Segundo pesquisa de rejeição do dia 03/10/2018 realizada pelo DataFolha e publicada no portal globo.com (Figura 45), em que foi perguntado aos eleitores quais candidatos eles não votariam de forma alguma, ambos os candidatos favoritos para disputar o segundo turno (Jair Bolsonaro e Fernando Haddad, diante pesquisa eleitoral do dia 03/10/2018), receberam uma alta rejeição (42% e 37% respectivamente), ou seja dentre todos os candidatos postulantes a presidente da república, os dois principais candidatos receberam as maiores rejeições e o candidato Bolsonaro a maior delas.

A princípio, a pesquisa é compatível com os resultados obtidos nas amostras, em que entre uma escala de -1 à +1, onde -1 é a nota com sentimento mais negativo e +1 a mais positivo, Jair Bolsonaro e Fernando Haddad receberam respectivamente -0,0033 e -0,066, ou seja, ambos sofreram muitos comentários negativos na rede social, porém entre os dois, o candidato Bolsonaro sofreu mais comentários negativos.

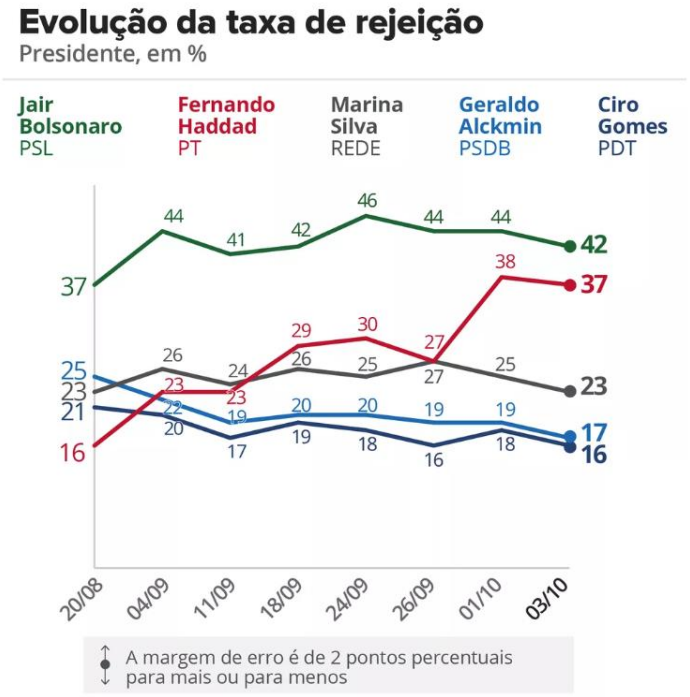

Figura 45: Gráfico da evolução da pesquisa de rejeição pela corrida presidencial 2018 (Pesquisa Rejeição, 03/10/2018).

Segundo pesquisa divulgada pelo jornal Folha de São Paulo do dia 06/10/2018, em que foi apresentada a intenção de voto por região, o candidato Bolsonaro tem predileção por eleitores de quatro das cinco regiões do país, enquanto o candidato Haddad tem vantagem na

região nordeste (36% contra 22%). As intenções do Bolsonaro foram: Centro Oeste (49% das intenções); Sul (44%); Sudeste (40%) e Norte (35%). Comparando a porcentagem dos votos com os mapas de pesquisa publicados no Google *Trends*, pode- se notar uma relação quase direta entre as regiões em que existiram mais pesquisas com os nomes dos candidatos (não importando o teor da busca) com as intenções de votos.

Vale ressaltar que mesmo o candidato Jair Bolsonaro estar liderando as intenções de voto segundo a pesquisa divulgada pelo portal globo.com do dia 03/10/2018, recebeu mais mensagens com sentimento negativo do que o candidato Fernando Haddad (segundo lugar nas pesquisas), isso talvez se dê devido a uma pulverização (até esse momento antes de começar o segundo turno) dos comentários negativos feito pelos eleitores a favor de Bolsonaro entre os demais candidatos, enquanto os comentários negativos feito por praticamente todos os demais eleitores se concentrarem no candidato Bolsonaro.

Vale ressaltar também um engajamento nas redes sociais maior entre os demais eleitores tentando que seu candidato consiga ir ao segundo turno.

4. Conclusão

Este trabalho teve como objetivo analisar algumas formas de busca de informação na web. Vale ressaltar que o intuito do trabalho foi exemplificar esses meios e não aprofundar em algum deles. Em todas as formas apresentadas, foram encontrados alguns prós e contras, além de algumas limitações, porém de forma geral o resultado foi satisfatório e embasado com pesquisas realizadas por instituições com credibilidade como o IBOPE e o DataFolha. Como assunto da pesquisa, foi usada a eleição presidencial no Brasil de 2018 (tema amplamente discutido no momento).

Dentre as pesquisas realizadas, foi feita uma análise de sentimento de *tweets*, para isso o método de aprendizagem estatístico *Naive* Bayes analisou *tweets* em português brasileiro. Esse método independe de idioma, pode ser utilizado para analisar texto poluído como é característico dos *tweets*. Uma parte de cada corpus (70%) foi utilizada para treino e a outra (30%) foi utilizada para testar o algoritmo e verificar a acurácia obtida. A princípio foi encontrada uma margem de erro de 20,2% para o candidato Jair Bolsonaro e 19,3% para o candidato Fernando Haddad.

Após a verificação e aprovação dos resultados de treino e teste do algoritmo de análise de sentimentos adotado, foi realizada a verificação da polaridade do conjunto de mensagens coletadas durante o período observado. Verificou-se uma divisão bem clara das mensagens onde mensagens de apoio (positivas) e mensagens de repúdio (negativa) para ambos os candidatos foram recebidas praticamente em igualdade, ou seja, ambos os candidatos receberam muitas mensagens positivas e negativas.

Esses resultados eram esperados por causa do que foi demonstrado por grande parte da população e após observar matérias e pesquisas eleitorais disponibilizadas por portais da internet feitas por empresas de pesquisa como o IBOPE e o DataFolha.

Tendo em vista a falta de um dicionário de palavras em português de boa qualidade (basicamente os dicionários disponíveis são em inglês), e como o tema da pesquisa gera muitas discussões acaloradas na rede (muitos xingamentos e palavras de baixo calão), sem contar os erros de português, os resultados obtidos acerca do pré-processamento, treino e testes a princípio foram satisfatórios quando considerados aspectos de classificação realizada por humanos.

Outra forma de análise, o Google *Trends* (demonstrado no trabalho) serve como excelente ferramenta visual, disponibilizando por exemplo gráficos, além de poder ser visto

como um “termômetro”, onde é possível analisar “o quê?”, “quanto?”, “quando?” e “onde?” estão sendo feitos essas buscas. Sendo assim, a mesma é interessante como ferramenta complementar para a pesquisa.

Como trabalho futuro, pode-se criar um conjunto maior de treinos e testes, além de melhorar e aumentar o dicionário de palavras em português disponível. Outro ponto que pode ser melhorado é a realização de análises mais detalhadas usando funções de distribuição acumulativa e teoria da decisão.

5. Referências Bibliográficas

Araújo, M.; Gonçalves; P.; Benevenuto, F.; & Cha, M. Métodos para análise de sentimentos no twitter. In Proceedings of the 19th Brazilian symposium on Multimedia and the Web (WebMedia'13), 2013.

Branski, R. Recuperação de informações na Web. Perspectivas em ciência da informação, v. 9, n. 1, 2008.

Brito, E. Mineração de Textos: Detecção automática de sentimentos em comentários nas mídias sociais Programa de Pós-Graduação Stricto Sensu. Dissertação de Mestrado, Faculdade Sistemas de Informação e Gestão do Conhecimento, Fundação Mineira de Educação e Cultura, 2016.

Cervi, C.R. Um Estudo sobre Mineração de Dados em Redes Sociais. Programa de Pós-Graduação em Computação, Universidade Federal do Rio Grande do Sul, 2008.

Chen, H.; Chiang, R.; Storey, C. Business intelligence and analytics: from big data to big impact. MIS quarterly, p. 1165-1188, 2012.

Digital in 2018. [Online] Disponível: https://wearesocial.com/blog/2018/01/global-digital-report-2018. Acesso em: 24 de Setembro de 2018

Estadão [Online] Disponível: https://link.estadao.com.br/noticias/empresas,facebook-chega-a-2-13-bilhoes-de-usuarios-em-todo-o-mundo,70002173062 , Acesso em: 24 de Setembro de 2018.

Fayyad, U.; Piatetsky-Shapiro, G. ; Smyth, P. From data mining to knowledge discovery in databases. AI magazine, v. 17, n. 3, p. 37, 1996.

Folha de São Paulo [Online] Disponível: https://www1.folha.uol.com.br/poder/2018/10/bolsonaro-lidera-em-4-das-5-regioes-e-tem-vies-de-alta-no-nordeste-reduto-do-pt.shtml , Acesso em: 06 de Outubro de 2018.

França, T.; Oliveira, J. Análise de Sentimento de Tweets Relacionados aos Protestos que ocorreram no Brasil entre Junho e Agosto de 2013. BraSNAM- III Brazilian Workshop on Social Network Analysis and Mining, 2014.

GitHub [Online] Disponível: https://github.com/rcuevass/Twitter-Sentiment-Analysis-Python Acesso em: 03 de Outubro de 2018.

Google Trends [Online] Disponível: https://trends.google.com.br/trends/?geo=BR, Acesso em: 29 de Setembro de 2018.

iFeel [Online] Disponível: www.ifeel.dcc.ufmg.br Acesso em: 29 de Setembro de 2018.

Iramuteq [Online] Disponível: http://www.iramuteq.org/, Acesso em: 04 de Outubro de 2018.

Junior, R. Um Framework Para Mineração De Textos De Redes Sociais. Programa de Pós-Graduação em Computação, Universidade Federal Do Ceará, 2016.

Kim, Y.; Street, W. N.; Menczer, F. Feature selection in unsupervised learning via evolutionary search. In: ACM. Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining. p. 365–369, 2000.

Machado, A; Ferreira, R.; Bittencourt, I.; Elias, E.; Brito, P. Mineração de texto em Redes Sociais aplicada à Educação a Distância. Colabor@-A Revista Digital da CVA-RICESU, v. 6, n. 23, 2010.

Manning, C.;Schütze; Hinrich; Christopher D. Introduction to information retrieval. Cambridge University Press, 2008.

Morais, E.; Ambrósio, A. Mineração de textos. Relatório Técnico–Instituto de Informática. Universidade Federal de Goiás, 2007.

Moujahid, A. [Online] Disponível: http://adilmoujahid.com/posts/2014/07/twitter-analytics/, Acesso em: 03 de Outubro de 2018.

Mozilla Developer Network [Online] Disponível: http://www.iramuteq.org/, Acesso em: 21 de Outubro de 2018.

Näsholm, P. Extracting Data from NoSQL Databases-A Step towards Interactive Visual Analysis of NoSQL Data, 2012.

Neves, J. Pesquisa qualitativa: características, usos e possibilidades. Caderno de pesquisas em administração, São Paulo, v. 1, n. 3, p. 1-5, 1996.

Pereira Jr., E. A. Google: Ferramenta De Busca De Informação Na Web. Saber Digital: Revista Eletrônica do CESVA, Valença, v. 1, n. 1, p. 18-32, 2008.

Pesquisa Presidencial [Online] Disponível: https://g1.globo.com/politica/eleicoes/2018/noticia/2018/09/28/pesquisa-datafolha-para-presidente-bolsonaro-28-haddad-22-ciro-11-alckmin-10-marina-5.ghtml , Acesso em: 28 de Setembro de 2018.

Pesquisa Rejeição [Online] Disponível:. https://g1.globo.com/politica/eleicoes/2018/noticia/2018/10/03/pesquisa-ibope-para-presidente-bolsonaro-32-haddad-23-ciro-10-alckmin-7-marina-4.ghtml, Acesso em: 03 de Outubro de 2018.

Pezzini, A. Mineração De Textos: Conceito, Processo E Aplicações. REAVI-Revista Eletrônica do Alto Vale do Itajaí, v. 5, n. 8, p. 58-61, 2017.

Prass, F. KDD–uma Visão Geral Do Processo. Recuperado em, v. 15, 2016.

Piatetsky-Shapiro, G., Fayyad, U., & Smyth, P. Advances in knowledge discovery and data mining. American Association for Artificial Intelligence. Menlo Park: American Association for Artificial Intelligence, 1996.

Russel, M . Mining the Social Web: Data Mining Facebook, Twitter, LinkedIn, Google+, GitHub, and More. " O'Reilly Media, Inc.", 2013.

Santos, A.P. Data Mining em Redes Sociais. Programa de Pós-Graduação, Universidade Federal de Santa Catarina, 2017.

Static Brain . Twitter Statics. [Online] Disponível em: https://www.statisticbrain.com/twitter-statistics/ , Acesso em: 03 de Outubro de 2018.

Suporte Google [Online] Disponível em: https://support.google.com/webmasters/answer/70897?hl=pt-BR, Acesso em: 24 de Setembro de 2018.

Teixeira, G.; Brum, B. ITeligence: Sistema de Apoio à Análise de Intenções. Trabalho Conclusão de Curso, Universidade Federal do Rio de Janeiro, 2018.

Turban, E., Sharda, R. E., & Delen, D. Decision support and business intelligence systems (9th ed.). New Jersey: Prentice Hall, 2010.

Wattenberg, M. Visual Exploration of Multivariate Graphs. In Proceedings of Conference Human Factors in Computing Systems (SIGCHI 2006). Montreal, Canada, 2006.

Weiand, A. Análise de sentimentos do Twitter com Naïve Bayes e NLTK .Trajetória Multicursos - volume 7, número 2, 2016.

Wix [Online] Disponível em: https://pt.wix.com/blog/2017/08/fatos-interessantes-sobre-o-algoritmo-do-google-que-voce-deveria-saber/ Acesso em: 4 de Novembro de 2018.

Vasconcelos, P. [Online] Disponível em: https://paulovasconcellos.com.br/aprenda-a-fazer-um-analisador-de-sentimentos-do-twitter-em-python-3979454f2d0d, Acesso em: 24 de Setembro de 2018.

Anexo A

1- Códigos GitHub

Os scripts mostrados a seguir, foram retirados de um projeto no GitHub chamado Twitter-Sentiment-Analysis-Python (https://github.com/rcuevass/Twitter-Sentiment-Analysis-Python) e serviram de base para a pesquisa.

1.1. Script 1

Este script lê o arquivo onde os *tweets* no formato JSON estão armazenados e filtra o valor do sentimento atribuído a cada tweet. O valor do sentimento é calculado pela soma da pontuação de cada termo presente no *tweet*. A pontuação de cada termo é obtida de um arquivo anexo que tem um número de termos com as pontuações correspondentes, caso um termo no *tweet* não seja encontrado no arquivo de referência, o valor atribuído será zero.

```
"""
        This script reads a file where tweets in JSON format are stored and
        prints to screen the sentiment value assigned to each tweet.

        The sentiment value is computed by adding up the score of each term
        in the tweet. The score of each term is obtained from a file that
        has a number of terms with corresponding scores. If a term in the
        tweet is not in the reference file then that score in the tweet
        gets assigned a value of zero.

        This script is excecuted as

                python 02_tweet_sentiment_scores.py AFINN-111.txt output.txt

        where AFINN-111.txt is the reference file containing the scores of terms
        and output.txt is the file containing the tweets that are going to be scored


    """



    import sys
    # Library to convert a JSON string into a Python data structure
    import json
```

```
# Library to handle regular expressions
import re


def hw():
    print 'Hello, world!'


def lines(fp):
    print str(len(fp.readlines()))


def return_parsed_afinn(sent_file):

        """
                This function parses the sentiment file (AFINN-111.txt in this case)
                and gets a dictionary from it.

                Arguments -- file containing sentiments
                Returns -- dictiionary

                dictionary keys -- words in file
                dictionary value -- sentiment value assigned to a given word

        """

        # We open file
        afinnfile = open(sent_file)


        # initialize an empty dictionary
        scores = {}


        # For every line in file...
        for line in afinnfile:
                 # The file is tab-delimited. "\t" means "tab character"
                 # So we split lines in tab
                term, score  = line.split("\t")
                # Convert the score to an integer and assign value to key in dictionary
                scores[term] = int(score)


        # return dictionary
        return scores
```

```
def tweet_sentiment_value(tweet_text,sent_dictionary):

    """
        This function assigns a sentiment value sent_value to a
        tweeter text, tweet_text, based on a given dictionary sent_dictionary

        Arguments -- tweet_text , sent_dictionary
        Returns -- sent_value

    """


    # Initialize sentiment value to zero
    sent_value = 0


    # We split the tweeter text to get a list out of it
    # and iterate over the resulting list.
    # For every word in the resulting list...
    for word in tweet_text.split():
        # ... we check if the word is in the set of the
        # given dictionary list...
        if word in sent_dictionary.keys():
            # If so, we update the sentiment value by the
            # value given to the word found
            sent_value += sent_dictionary[word]

    # Return resulting sentiment value
    return sent_value


def process_tweets(tweet_file,afinn_dict):

    """
        This function, based  on a sentiment dictionary afinn_dict,
        assigns a sentiment value to every tweet in a collection of
        tweets contained in a file tweet_file

        Important -- tweet_file is given in JSON format

    """
    # We open tweet_file and store it
    tweetfile = open(tweet_file)

    # Initialize an empty dictionary
```

```
    scores = {}

    # Initialize counter of tweets to be processed
    count_tweet = 0

    # For every line in tweetfile...
    for line in tweetfile:

        # Since each line is in JSON format we convert it
        # into a Python data structure
        parsed_json = json.loads(line)

        # Initialize sentiment value
        sentiment_value = 0

        # Since we are only interested in tweets containing text in English
        # we check there is actually text in a given tweet and that is written in
        # English
        if ('text' in parsed_json.keys()) and ('lang' in parsed_json.keys()) and (parsed_json['lang']
== 'en'):

            # If so, we update the number of tweet to be processed
            #count_tweet += 1

            # Since the text in the parsed_json dictionary is unicoded ...
            unicode_string = parsed_json['text']
            # ... we encode it to a Python string
            encoded_string = unicode_string.encode('utf-8')

            # We do some extra cleaning by means of regular expressions
            # IMPORTANT .- We have not considered extra cleaning regarding
            # URL's, hashtags and/or @ included in the tweets. This is something
            # that definetly can be refined in the future...
            encoded_string = re.sub(r'[^a-z0-9\s]', '', encoded_string.strip().lower())
```

```
                # We assign sentiment value based invoking the tweet_sentiment_value function
                sentiment_value = tweet_sentiment_value(encoded_string,afinn_dict)

                # Finally print tweet along its sentiment value
                #print encoded_string , sentiment_value
                print sentiment_value


            else:


                print sentiment_value

def main():


    sent_file , tweet_file = sys.argv[1] , sys.argv[2]

    #parse_afinn()
    dict_scores = {}
    dict_scores = return_parsed_afinn(sent_file)
    #print dict_scores.items()
    process_tweets(tweet_file,dict_scores)


if __name__ == '__main__':
    main()
```

1.2. **Script 2**

Este script lê o arquivo onde os *tweets* no formato JSON estão armazenados e rastreia a pontuação de cada termo de cada *tweet*.

```
"""

        This script reads a file where tweets in JSON format are stored and
        prints to screen the score of each term composing each of the tweets
        read from file.

        The score of each term is obtained as follows:

            i. If the term is in the bank reference file, it gets that score
```

ii. If the term is not in the bank, then it gets assigned the score
of the whole tweet

This script is excecuted as

python 03_term_sentiment.py AFINN-111.txt output.txt

where AFINN-111.txt is the reference file containing the scores of terms
and output.txt is the file containing the tweets that are going to be scored

"""

```
import sys
# Library to convert a JSON string into a Python data structure
import json
# Library to handle regular expressions
import re


def hw():
    print 'Hello, world!'


def lines(fp):
    print str(len(fp.readlines()))
def return_parsed_afinn(sent_file):

        """
                This function parses the sentiment file (AFINN-111.txt in this case)
                and gets a dictionary from it.

                Arguments -- file containing sentiments
                Returns -- dictiionary

                dictionary keys -- words in file
                dictionary value -- sentiment value assigned to a given word

        """

        # We open file
        afinnfile = open(sent_file)
```

```
    # initialize an empty dictionary
    scores = {}

    # For every line in file...
    for line in afinnfile:
            # The file is tab-delimited. "\t" means "tab character"
            # So we split lines in tab
            term, score  = line.split("\t")
            # Convert the score to an integer and assign value to key in dictionary
            scores[term] = int(score)

    # return dictionary
    return scores


def tweet_sentiment_value(tweet_text,sent_dictionary):

    """
            This function assigns a sentiment value sent_value to a
            tweeter text, tweet_text, based on a given dictionary sent_dictionary

            Arguments -- tweet_text , sent_dictionary
            Returns -- sent_value

    """

    # Initialize sentiment value to zero
    sent_value = 0

    # We split the tweeter text to get a list out of it
    # and iterate over the resulting list.
    # For every word in the resulting list...
    for word in tweet_text.split():
            # ... we check if the word is in the set of the
            # given dictionary list...

            if word in sent_dictionary.keys():
```

```
                    # If so, we update the sentiment value by the
                    # value given to the word found
                    sent_value += sent_dictionary[word]

    # Return resulting sentiment value
    return sent_value


def print_tweet_term_value(tweet_text,sent_dictionary):

    """
        This function determines the score of each term that composes a
        given tweet, tweet_text, based on a dictionary, sent_dictionary


        Arguments -- tweet_text , sent_dictionary
        Returns -- It prints to screen the pair (term,score)

    """


    # We firs determine the sentiment of the tweet by means of the
    # function tweet_sentiment_value
    sent_tweet_value = tweet_sentiment_value(tweet_text,sent_dictionary)



    # We split the tweeter text to get a list out of it
    # and iterate over the resulting list.
    # For every word in the resulting list...
    for word in tweet_text.split():
        # ... we check if the word is in the set of the
        # given dictionary list...


        # If so ...
        if word in sent_dictionary.keys():
            # ... we print the term along the score obtained
            # from the dictionary
            print word , sent_dictionary[word]


        # If the word is not part of the dictionary and does not contain indiciation
        # of being a hiperlink....
        elif (word not in sent_dictionary.keys()) and ('http' not in word):
```

```
                    # ... we print the term along the score obtained from
                    # the overall sentiment value of the tweet
                    print word , sent_tweet_value



def print_term_scores_tweets(tweet_file,afinn_dict):

        """
                This function, based  on a sentiment dictionary afinn_dict,
                assigns a sentiment value to every tweet in a collection of
                tweets contained in a file tweet_file

                Important -- tweet_file is given in JSON format

        """
        # We open tweet_file and store it
        tweetfile = open(tweet_file)

        # Initialize an empty dictionary
        scores = {}


        # For every line in tweetfile...
        for line in tweetfile:


                # Since each line is in JSON format we convert it
                # into a Python data structure
                parsed_json = json.loads(line)


                # Initialize sentiment value
                sentiment_value = 0


                # Since we are only interested in tweets containing text in English
                # we check there is actually text in a given tweet and that is written in
                # English
                if ('text' in parsed_json.keys()) and ('lang' in parsed_json.keys()) and (parsed_json['lang']
== 'en'):
```

```
                # Since the text in the parsed_json dictionary is unicoded ...
                unicode_string = parsed_json['text']
                # ... we encode it to a Python string
                encoded_string = unicode_string.encode('utf-8')


                # We do some extra cleaning by means of regular expressions
                # IMPORTANT .- We have not considered extra cleaning regarding
                # URL's, hashtags and/or @ included in the tweets. This is something
                # that definetly can be refined in the future...
                encoded_string = re.sub(r'[^a-z0-9\s]', '', encoded_string.strip().lower())


                # Invoke the function print_tweet_term_value to print the pair (term,score)
                # for each term in the tweet in turn
                print_tweet_term_value(encoded_string,afinn_dict)


def main():


        # Determine names of files from command line
    sent_file , tweet_file = sys.argv[1] , sys.argv[2]

    # Create an empty dictionary...
    dict_scores = {}
    # ... and store in it the dictionary obtained from the reference file, sent_file
    dict_scores = return_parsed_afinn(sent_file)
    # Feed the names of such files to the function that will print terms along scores
    print_term_scores_tweets(tweet_file,dict_scores)


if __name__ == '__main__':
    main()
```

1.3. **Script 3**

O script imprime a frequência encontrada de cada termo em todos os *tweets*.

```
"""

        This script reads a file where tweets in JSON format are stored and
```

```
        prints to screen the frequency of each term in all of the tweets

        This script is excecuted as

                python 04_tweet_frequency.py output.txt

        where output.txt is the file containing the tweets that are going to be scored

"""



import sys
# Library to convert a JSON string into a Python data structure
import json
# Library to handle regular expressions
import re


def hw():
    print 'Hello, world!'


def lines(fp):
    print str(len(fp.readlines()))


def get_freq_tweets(tweet_file):

        """
                This function reads a file containing tweets in JSON format
                and prints to screen the pair (word,pct), where word is the term
                in tweets and pct is the fraction of that term in the whole corpus
                of tweets

                Important -- tweet_file is given in JSON format

                Argument -- tweet_file

                Rerurn -- total count of terms in tweets (word_count)
                                dictionary of frequency terms (freq_dict)

        """
```

```
    # We open tweet_file and store it
    tweetfile = open(tweet_file)

    # Initialize an empty dictionary where we will store
    # the unique terms
    freq_dict = {}


    # Initialize count of words in tweets
    word_count = 0


    # For every line in tweetfile...
    for line in tweetfile:


        # Since each line is in JSON format we convert it
        # into a Python data structure
        parsed_json = json.loads(line)


        # Since we are only interested in tweets containing text in English
        # we check there is actually text in a given tweet and that is written in
        # English
        if ('text' in parsed_json.keys()) and ('lang' in parsed_json.keys()) and (parsed_json['lang']
== 'en'):


            # Since the text in the parsed_json dictionary is unicoded ...
            unicode_string = parsed_json['text']


            # ... we encode it to a Python string
            encoded_string = unicode_string.encode('utf-8')


            # We do some extra cleaning by means of regular expressions
            # IMPORTANT .- We have not considered extra cleaning regarding
            # URL's, hashtags and/or @ included in the tweets. This is something
            # that definetly can be refined in the future...
            encoded_string = re.sub(r'[^a-z0-9\s]', '', encoded_string.strip().lower())


            # For a particular tweet, we make a list out of it and loop over it...
            for word in encoded_string.split():
```

```
                # We check if the word is already in the dictionary.
                # If it is and is not related to a hyper-link....
                if ( word in freq_dict.keys() ) and ('https' not in word):
                        # we increase the counter for that term...
                        freq_dict[word] +=1
                        # ... and increase the word counter
                        word_count += 1
                # If the term is not in the dictionary...
                elif (word not in freq_dict.keys()) and ('https' not in word):
                        # ... we add the term to it...
                        freq_dict[word] = 1
                        # ... and increase the word counter
                        word_count += 1



        # Now that we have the dictionary, we loop over the
        for word in freq_dict.keys():
                # We compute the percentage associated with this term
                pct = float(freq_dict[word])/word_count
                # And print to screen the pair.
                print "%s %.4f" % (word, pct)


        # We return word_count and dictionary of frequency in case we need it latter
        return word_count , freq_dict



def main():


        # Determine name of files from command line
    tweet_file = sys.argv[1]

    # Feed the names of such files to the function that will print terms along scores
    get_freq_tweets(tweet_file)


if __name__ == '__main__':
    main()
```