UNIVERSIDADE FEDERAL DE VIÇOSA
CAMPUS FLORESTAL

DOUGLAS DA CRUZ PEREIRA

UMA METODOLOGIA PARA COLETA E ANÁLISE DE DADOS DE REDES SOCIAIS

FLORESTAL - MINAS GERAIS
2017

DOUGLAS DA CRUZ PEREIRA

UMA METODOLOGIA PARA COLETA E ANÁLISE DE DADOS DE REDES SOCIAIS

Monografia apresentada ao Curso de Ciência da Computação da Universidade Federal de Viçosa, como requisito para obtenção do título de bacharel em Ciência da Computação. Orientador: Prof. Dr. Daniel Mendes Barbosa

FLORESTAL - MINAS GERAIS

2017

DOUGLAS DA CRUZ PEREIRA

UMA METODOLOGIA PARA COLETA E ANÁLISE DE DADOS DE REDES SOCIAIS

Monografia apresentada ao Curso de Ciência da Computação da Universidade Federal de Viçosa, como requisito para obtenção do título de bacharel em Ciência da Computação. Orientador: Prof. Dr. Daniel Mendes Barbosa

---

Prof. Dr. Daniel Mendes Barbosa
UFV - *Campus* Florestal

Profa. Dra. Gláucia Braga e Silva
UFV - *Campus* Florestal

Prof. Dr. Fabrício Aguiar Silva
UFV - *Campus* Florestal

FLORESTAL - MINAS GERAIS

AGRADECIMENTOS

A Deus, por me fazer chegar e alcançar lugares que nunca achei que fossem possíveis.

Aos meus pais, Jadir e Maria Aparecida, por todo carinho, apoio e paciência em todos esses tempos. Amo vocês.

A minha irmã Júlia, meu amigo Jonathan e minha namorada Stefani. O apoio e conselhos de vocês sempre foram fundamentais na minha caminhada.

A todos meus familiares, que sempre estiveram presentes nesses períodos e sempre acreditaram muito em mim. Um agradecimento especial aos meus padrinhos, Daniel, Geraldo e José, minhas madrinhas Aparecida e Cristiane, ao tio Brás e sua família as minhas avós Ana e Maria e as primas Danúbia e Daniele, por todo apoio que sempre me deram.

Ao meu orientador Daniel, por todos os ensinamentos e todas as vezes que me indicou quais caminhos deveriam ser seguidos. Seu jeito alegre sempre me ensinou muito, obrigado.

Aos grandes amigos que fiz nessa faculdade e um agradecimento especial ao Edson, ao Samuel. Vocês são feras.

Aos docentes do curso que sempre desempenham seu trabalho com muita dedicação. Agradeço também à UFV por tornar esse sonho possível na minha vida.

A SetApp e ao Movimento Empresa Júnior, que abriram as portas pra mim. São lugares onde além de aprender muito, conheci pessoas incríveis.

RESUMO

A facilidade de uso das redes sociais, seja em um computador pessoal ou por meio de aplicativos em *smartphones*, juntamente com algoritmos cada vez mais eficazes em mostrar às pessoas os seus interesses, fez com que elas sejam usadas atualmente até por pessoas que não possuíam tanto contato com a tecnologia, como pessoas da terceira idade, por exemplo. Esse crescimento no número de usuários das redes sociais provocou uma revolução na maneira com que as pessoas se comunicam, interagem e consequentemente se influenciam com as opiniões expressadas nesses ambientes. Isso foi rapidamente percebido pelos mais diversos tipos de organizações, que passaram a buscar nesse meio eletrônico formas de atingir novos clientes interessados em consumir seus produtos ou serviços. Muitas dessas organizações possuem setores responsáveis por atualizar seus perfis ou páginas nas redes sociais com as novidades, criando-se diversas publicações diariamente. Além disso, para se entender melhor o público-alvo, seus desejos e suas opiniões sobre produtos e serviços já existentes, bem como a imagem da organização perante os usuários da rede, faz-se necessária uma análise dos dados disponíveis. Atualmente, existem vários estudos sobre como analisar o conteúdo dessas publicações, por serem em linguagem natural e por possuírem em muitos casos elementos complexos como ironia ou sarcasmo, ao mesmo tempo que existem poucos trabalhos em português e que sejam direcionados a iniciantes. Por essa razão, este trabalho tem como objetivo auxiliar pesquisadores que queiram começar a trabalhar nesta área, trazendo alguns dos principais conceitos envolvidos, bem como sua aplicação prática, através da definição de um método para se fazer a coleta e a análise dos dados. Como resultados do trabalho, destacam-se o levantamento e exemplificação detalhada do uso de bibliotecas disponíveis para a coleta de dados de duas das principais redes sociais da atualidade, como também de ferramentas disponíveis para a análise dos dados coletados. Durante todo esse processo, também foi necessária a construção de códigos específicos para se filtrar os dados desejados e para a conversão de formatos de saída de aplicações de coleta para formatos de entrada de ferramentas de análise, obtendo-se os passos que podem agora ser facilmente entendidos e replicados por outros pesquisadores em seus trabalhos, mesmo que não tenham experiências prévias na área.

Palavras-chave: Coleta de dados, Redes sociais, Análise de dados, Análise de sentimento.

ABSTRACT

The ease of use of social networks, whether on a personal computer or through smartphone apps, coupled with increasingly effective algorithms in showing people their interests, has made the social networks nowadays even used by people who did not have so much contact with technology, such as senior citizens, for example. This growth in the number of users of social networks has provoked a revolution in the way people communicate, interact and, consequently, influence the opinions expressed in those environments. This was quickly perceived by the most diverse types of organizations, who came to seek in this electronic environment ways to reach new customers interested in consuming their products or services. Many of these organizations have sectors responsible for updating their profiles or pages in social networks with the news, creating several publications daily. In addition, to better understand the target audience, their wishes and their opinions about existing products and services, as well as the image of the organization vis-à-vis network users, it is necessary to analyze the available data. Currently there are several studies on how to analyze the content of these publications, because they are in natural language and have in many cases complex elements such as irony or sarcasm, while there are few works in Portuguese that are aimed at beginners. For this reason, this work aims to help researchers who want to start working in this area, bringing some of the main concepts involved, as well as their practical application, through the definition of a method for collecting and analyzing the data. As results of the work, we highlight the detailed collection and exemplification of the use of available libraries for the data collection of two of the main social networks of the present time, as well as of tools available for the analysis of the data collected. Throughout this process, it was also necessary to construct specific codes to filter the desired data and convert output formats from applications of collect to input formats of analysis tools, obtaining the steps that now can be easily understood and replicated by other researchers in their work, even if they do not have prior experience in the field.

Keywords: Data collect, Social networks, Data analysis, Sentiment Analysis.

Lista de Figuras

LISTA DE ABREVIATURAS E SIGLAS

API - *Application Programming Interface*
JSON - *JavaScript Object Notation*
OSNs - *Online Social Networks*
SNSs - *Social Network Sites*
URL - *Uniform Resource Locator*

Sumário

1 Introdução

A recente evolução da internet trouxe comodidade aos usuários. O uso de aplicativos e sites que facilitam o dia-a-dia dos usuários tem se tornado cada vez mais comum. A simplicidade de acesso e a usabilidade intuitiva que esses aplicativos proporcionam faz com que o número de usuários aumente a cada dia. Esse crescimento tem despertado o interesse de diversas empresas e instituições que tentam, de alguma forma, se aproximar desse novo público a fim de adquirir visibilidade e anunciar seu produto para os interessados.

As redes sociais, citadas por Benevenuto, Almeida e Silva (2011) como redes sociais *online* (OSNs - *Online Social Networks*) ou conforme apresentadas por Ellison et al. (2007) como sites de redes sociais (SNSs - *Social Network Sites*), que as define como serviço web capaz de permitir aos usuários pelo menos três ações: a primeira delas é a construção de perfis públicos ou semi-públicos; a segunda ação diz respeito à criação de listas contendo usuários que possuem algum tipo de conexão com um determinado perfil; e por fim, acessar as listas tanto deste usuário como de outros.

Entre as OSNs mais populares no Brasil, pode-se citar o Facebook e o Twitter. No Facebook, as instituições não possuem uma conta como usuários normais e sim as chamadas "páginas", onde podem publicar em diversos formatos, desde o mais simples, que seria um texto informando sobre determinado assunto, até um marco na história ou transmissão em vídeo ao vivo, por exemplo. Os usuários dessa rede social que seguem uma determinada página recebem essas atualizações e com isso podem reagir, comentar e até mesmo compartilhá-la para que a publicação possa alcançar também seus amigos. O Twitter, por sua vez, não possui essa diferença de usuários: tanto as organizações quanto as pessoas possuem o mesmo tipo de conta, por meio da qual publicam sua mensagens, chamadas de *Tweets*, que devem conter no máximo 140 caracteres cada uma. As possíveis interações com essas mensagens são: curtir, responder e compartilhar (*Retweetar*).

Por padrão, as publicações feitas no Twitter são públicas, ou seja, todos possuem acesso ao conteúdo independente de seguir ou não o autor, inclusive pessoas sem um perfil na rede social podem visualizar o conteúdo colocado como público. Entretanto, caso queira, o utilizador dessa rede pode tornar suas publicações privadas. Dessa forma, só terão acesso ao conteúdo publicado pessoas que o seguem. Já no Facebook, as publicações de um usuário comum em sua linha do tempo, por padrão, são privadas, o que não impede que o autor altere para pública em algum momento. Por outro lado, as publicações das páginas do Facebook são públicas, sendo assim, inclusive pessoas sem uma conta no site possuem acesso ao conteúdo.

De acordo com Benevenuto, Almeida e Silva (2011), o fato das OSNs permitirem aos seus usuários publicar o próprio conteúdo está relacionado com o aumento de popularidade de tais redes. Com todo esse crescimento, é notável que hoje, muitas pessoas não se vêem longe da Internet, principalmente devido às facilidades proporcionadas pelos sites na rede.

No Facebook, por exemplo, quando se trata da página de uma empresa, os usuários tem acesso ao contato dessas empresas, horário de funcionamento, entre outros. É possível ainda, entrar em contato diretamente com a página da empresa através de mensagens de texto. Essa facilidade de contato aproxima cada vez mais as organizações dos clientes e potenciais consumidores.

Bausch e McGiboney (2009) citam alguns dados interessantes sobre esse crescimento das OSNs, entre eles que a cada 10 pessoas online, três estão no Facebook, e que a cada 11 minutos online, um minuto foi gasto em uma rede social ou *blog*.

O aumento da utilização das redes sociais se dá principalmente pelo fato de ser uma forma gratuita comunicação. Com cada vez mais pessoas utilizando as OSNs há também um aumento de interesse das organizações nos seus dados. Uma forma muito utilizada para se saber o conteúdo de uma publicação é através da área de análise de sentimento. Martins, Pereira e Benevenuto (2015) descrevem essa área como sendo "uma ferramenta popular para mineração de dados em redes sociais *online*".

1.1 Objetivo

O presente trabalho tem como objetivo descrever uma metodologia para apoiar a coleta e a análise dos dados colocados como públicos nas redes sociais Facebook e Twitter. Ou seja, a ideia principal é apresentar um caminho que possa ser seguido por pesquisadores que pretendem elaborar um estudo sobre os dados gerados em redes sociais, mostrando os principais conceitos envolvidos e uma possível aplicação de tais conceitos na prática, sempre com o uso de ferramentas de código aberto ou disponíveis para fins acadêmicos ou ainda códigos criados no próprio trabalho.

Na etapa da coleta, os dados obtidos contendo publicações das OSNs, serão armazenados em arquivos. Os arquivos obtidos, poderão ser utilizados como entrada para a etapa seguinte, que é a análise dos dados. A primeira etapa da análise será a utilização de um software para obtenção do sentimento de cada uma das publicações. Na etapa seguinte, utilizando conceitos de aprendizado de máquina, será treinada uma rede neural com o intuito de prever o sentimento das publicações. O resultado será uma comparação entre as previsões da rede e o sentimento previsto pelo software.

1.2 Justificativa

Apesar de existirem diversos estudos nas áreas de coleta e análise de dados em redes sociais, observa-se que são poucos os trabalhos que tratam do assunto voltado para a prática. Sendo assim, faz-se necessário este estudo que apresenta não só aplicações para a coleta dos dados como também uma apresentação geral sobre o tipo de coleta em cada uma das OSNs estudadas de tal forma que pequenas mudanças que eventualmente possam ser realizadas no futuro, por parte da rede social, não tornariam este trabalho inválido.

Já com relação às ferramentas para análise dos dados coletados de redes sociais, pode-se observar principalmente que o campo da análise de sentimento ainda é pouco explorado em se tratando de dados em português.

1.3 Organização do Trabalho

Este trabalho está organizado da seguinte forma: na seção 2 são apresentados trabalhos relacionados que não só motivaram, como também indicaram os caminhos que foram seguidos neste trabalho. Em seguida, a seção 3 apresenta o conteúdo relacionado à coleta de dados, onde são apresentados trechos de códigos e ferramentas utilizadas para a coleta dos dados no Facebook e no Twitter. A análise dos dados é apresentada na seção 4 onde é citado um software utilizado para análise de sentimento em textos. Em seguida, são definidos alguns dados a serem utilizados tanto para treino, quanto para teste de uma rede de aprendizado de máquina. Na seção 5, é apresentada uma breve conclusão deste trabalho e algumas propostas para trabalhos futuros. Por fim, são listadas as referências utilizadas neste trabalho e os apêndices contendo os códigos.

2 Trabalhos Relacionados

Nesta seção, são apresentados alguns dos estudos tomados como referência para realização e motivação deste trabalho. Para um melhor entendimento, foi dividida em três subseções, sendo a primeira sobre análise de dados em redes sociais e a segunda apresenta estudos sobre análise de sentimento.

2.1 Motivação para análise de dados em redes sociais

Benevenuto, Almeida e Silva (2011) apontam quatro elementos que consideram como sendo os motivos para que estudos sejam realizados na área das Redes Sociais. O primeiro deles é o motivo "comercial", onde, nos últimos anos, nota-se facilmente um aumento do número de empresas que usam do *marketing* online com a finalidade de atingir a grande quantidade de usuários presentes nas OSNs, sendo importante ressaltar também o papel da análise de sentimento nesse contexto, visto que publicações positivas são mais atrativas aos leitores. A motivação "sociológica" é a segunda apresentada pelos autores e visa explicar o comportamento dos usuários. A motivação seguinte são as "melhorias dos sistemas atuais", sendo que um estudo nessa área pode indicar possíveis melhorias com o intuito de fazer com que os usuários não abandonem a rede social. Por fim, existe a motivação da "segurança e conteúdo indesejável" que diz respeito ao estudo daquilo que os autores chamam de conteúdo não solicitado que nada mais é do que spam[1].

2.2 Métodos de análise de sentimento

A análise de sentimento nada mais é do que uma maneira computacional de se analisar textos e julgá-los como positivos, negativos ou neutros. O principal objetivo desse tipo de análise é "mensurar e qualificar sentimentos expressos em uma ou mais frases" (MALHEIROS, 2014, p. 2).

Para as empresas presentes nas redes sociais, a análise de sentimento pode ser muito interessante, pois a partir dela, é possível observar os comentários que clientes fazem a respeito de um determinado produto. Caso a maioria seja positivo, tem-se um indicativo de que o produto está sendo bem aceito no mercado.

Com relação à análise de dados, Araújo et al. (2013) revela uma comparação entre oito diferentes métodos de análise de sentimento: LIWC, Happiness Index, SentiWordNet, SASA, PANAS-t, Emoticons, SenticNet e SentiStrength. Para cada um dos métodos, utilizou-se duas bases de dados provenientes de OSNs, sendo a primeira composta por 1,8 bilhões de mensagens em inglês coletadas do Twitter e a segunda composta por dados cujos sentimentos foram rotulados por humanos. Por fim, os autores apresentam os resultados

[1] O termo, de acordo com o portal Techmundo, "é usado para se referir às mensagens eletrônicas que são enviadas para você sem o seu consentimento". Fonte: https://www.tecmundo.com.br/spam/223-o-que-e-spam-.htm Acesso em: 06/11/2017

da comparação dos métodos, onde o Emoticons obteve melhor resultado. Porém, esse método só analisa dados que possuam emoticons[2], por conta disso, sua abrangência foi menos de 10%. Dos métodos analisados, o SentiStrength obteve a segunda melhor taxa de acertos, por volta de 76% e uma abrangência de 61%.

Um estudo apresentado por Gonçalves et al. (2015), apresenta alguns desafios da análise de sentimento, que são a detecção de sarcasmo e ironia. Essa dificuldade se dá principalmente pelo fato da análise ser feita em texto escrito, visto que na fala, a identificação é possível graças aos trejeitos do locutor.

2.3 Aprendizado de máquina e redes neurais

De acordo com o estudo de Monard e Baranauskas (2003), pode-se dividir o aprendizado de máquina em supervisionado, onde são passados para o algoritmo dados considerados corretos de maneira que é feito um treinamento, e não-supervisionado, que por sua vez, não possui dados de treino, ou seja, são passados os dados e o algoritmo tenta encontrar algumas características em comum nesses dados. Os autores apresentam ainda uma divisão do aprendizado supervisionado em classificação e regressão.

Em sua tese, Batista et al. (2003), explica que as redes neurais foram inspiradas no sistema nervoso, e a partir dessa inspiração, foram construídos os modelos matemáticos. Haykin (2007) apresenta também alguns termos comumente utilizados para se referir a redes neurais, são eles: "neurocomputadores", "redes conexionistas" e "processadores paralelamente distribuídos".

Outro conceito importante nesta área é o *deep learning*, que segundo LeCun, Bengio e Hinton (2015), permite que modelos computacionais aprendam sobre dados representados com diversos níveis de abstração.

Para o estudo do aprendizado de máquina, foi utilizado três cursos *online* pela plataforma Coursera[3]: Fundações do aprendizado de máquina: uma abordagem por estudo de caso (GUESTRIN; FOX, 2015a), *Machine Learning: Classification* (GUESTRIN; FOX, 2015b) e *Applied Machine Learning in Python* (COLLINS-THOMPSON, 2017). Sendo os dois primeiros desenvolvidos pela Universidade de Washington e o último pela Universidade de Michigan. Estes cursos possuem aulas teóricas e práticas, e seu conteúdo é voltado para a utilização de técnicas de aprendizado de máquina. Foi utilizado na parte prática deste trabalho o conceito de classificação, para se fazer a análise de sentimento, classificando postagens das redes sociais como positivas, negativas ou neutras.

[2]"Representação de uma expressão facial". "Normalmente utilizado para expressar sentimentos ou humor". Fonte: https://en.wikipedia.org/wiki/Emoticon Acesso em: 09/11/2017

[3]Um site que possui aulas em vídeo de diversos cursos. Pode ser acessado pelo *link*: https://www.coursera.org/

3 Coleta de Dados

Neste capítulo será apresentado, de forma detalhada, o que é necessário para se conseguir extrair publicações das OSNs. No caso do Facebook, serão extraídas publicações de uma determinada página. No Twitter, as publicações podem ser obtidas tanto de um usuário público, quanto por uma busca feita por publicações contendo um determinado conjunto de caracteres ou até mesmo publicações em uma determinada região geográfica. As saídas geradas neste capítulo servirão como entrada para a análise dos dados apresentada no capítulo 4.

Como mencionado no capítulo 1 deste trabalho, será discutida aqui a coleta de dados públicos em duas grandes redes sociais: o Facebook e o Twitter. Para isso, é necessário primeiramente criar uma aplicação nos respectivos sites para desenvolvedores. No caso do Facebook, o site é "*Facebook for Developers*" (https://developers.facebook.com/), e no caso do Twitter, é o "*Application Management*" (https://apps.twitter.com/). Em ambos os casos, é necessário que o programador possua um usuário na rede social. Sendo assim, no caso do Facebook pode-se clicar em "adicionar um novo aplicativo", enquanto no Twitter é possível clicar em "*create new app*". Feito isso, o programador é capaz de obter suas chaves de acesso.

3.1 Chaves de Acesso

Para validar o acesso da aplicação aos servidores da rede social, é necessário entrar com as chaves de acesso. Essa chave fornecida ao programador é única para cada aplicação que ele vá criar. No caso do Twitter, por exemplo, é preciso definir quais recursos seu *software* terá acesso, sendo que existem três tipos de permissões para essa rede social. A primeira delas permite apenas o acesso de leitura dos dados públicos. O segundo tipo de permissão é de leitura e escrita, ou seja, é permitido também a publicação de novos conteúdos. Na última forma de autorização da aplicação, além de ler e publicar, permite que ela tenha acesso às *Direct Messages*, que é o nome dado à forma de se enviar mensagens privadas a outro usuário nesta OSN.

Após a criação da aplicação, para obter a chave de acesso, que no Facebook é chamada de "Chave Secreta do Aplicativo", é necessário que, no painel da aplicação, o usuário clique em "Mostrar", no campo indicado para essa chave. Para que a solicitação seja processada, é necessário primeiro, digitar a senha. No Twitter, por sua vez, existem quatro chaves, são elas: a "*Consumer Key*", "*Consumer Secret*", "*Access Token*" e "*Access Token Secret*". Para obter essas chaves, o programador precisa ir em "*Keys and Access Token*", duas dessas chaves aparecerão. Para gerar as demais, é necessário apenas clicar em "*Create my access token*", e assim, estas chaves serão geradas.

3.2 Linguagem de programação e bibliotecas

A linguagem de programação utilizada para a coleta de dados foi Java. A principal motivação para essa escolha foi justamente o fato de que as bibliotecas escolhidas, tanto para fazer a coleta dos dados, quanto para imprimir os dados no formato Notação de Objetos JavaScript (JSON[4] - *JavaScript Object Notation*) no arquivo, são feitas para a linguagem Java.

Durante o desenvolvimento do trabalho, foram utilizadas bibliotecas específicas para recuperação de dados em cada uma das redes sociais. Além disso, a biblioteca Gson foi utilizada em comum por ambas aplicações. Esta biblioteca foi desenvolvida pela Google e seu objetivo é fazer a conversão de objetos Java para um texto no formato JSON e também é capaz de transformar um texto em um objeto Java equivalente. Os principais métodos para fazer essa conversão são respectivamente "toJson()" e "fromJson()".

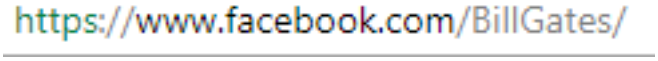
Para a coleta dos dados do Facebook foi utilizada a biblioteca "restfb". Alguns dados importantes e curiosidades podem ser encontradas no site (http://restfb.com/). A restfb, como seu próprio site indica, é um cliente da *Graph API*, biblioteca do Facebook.

A biblioteca "twitter4j" (leia-se: *Twitter for J*, em português: Twitter para J), como o próprio nome sugere, é utilizada para recuperação de dados no Twitter voltada para linguagem Java. Como indicada no próprio site (http://twitter4j.org/), esta é uma biblioteca extra oficial Java para a Twitter API.

3.3 Coleta de dados do Facebook

A aplicação para coleta dos dados nas páginas do Facebook necessita da identificação da página. Esta identificação pode ser encontrada de duas maneiras, sendo que a primeira é pelo link da página que segue o formato apresentado na Figura 1. Através da URL[5] (Localizador Padrão de Recursos) pode-se acessar partes diferentes de uma página. Por exemplo, para visualizar as fotografias publicadas, basta utilizar a URL padrão (Figura 1) e acrescentar "/photos". Neste trabalho, são coletadas publicações das páginas, portanto, foi adicionado "/posts" ao final da URL.

Figura 1: Identificação da página do Facebook pela URL



A segunda maneira de se encontrar essa identificação da página consiste em verificar logo abaixo do nome da página, onde a mesma estará precedida pelo símbolo arroba (@).

[4] "É uma formatação leve de troca de dados. Para seres humanos, é fácil de ler e escrever. Para máquinas, é fácil de interpretar e gerar". Fonte: https://www.json.org/json-pt.html Acesso em 11/11/2017

[5] "É o endereço de um recurso disponível em uma rede, seja a rede internet ou intranet". Fonte: https://www.significados.com.br/url/. Acesso em 22/11/2017

Na Figura 2, está destacado de vermelho o local onde se encontra essa identificação da página.

Nas duas maneiras apresentadas (Figuras 1 e 2), foi usado como exemplo a página do magnata e filantropo estadunidense Bill Gates, podendo-se observar que a identificação correspondente é “BillGates”.

Figura 2: Identificação da página do Facebook abaixo do nome

3.3.1 Construção da aplicação para coleta dos dados

Na visão do usuário, a aplicação que coleta dados das páginas do Facebook, solicita para que o mesmo digite a identificação da página e em seguida, começa a coletar os dados que serão armazenados em dois arquivos. O primeiro deles possui o mesmo nome de identificação da página com extensão “.json” e cada uma das linhas desse arquivo possui um objeto referente a uma publicação em formato JSON, sendo que esse objeto possui a mensagem da publicação, a data que foi publicada, e um ID (um inteiro de valor único). O segundo arquivo também possui o mesmo nome, porém sua extensão é “.txt”, e cada linha deste arquivo possui a mensagem de uma publicação. Quando a aplicação coleta todas as publicações contidas na página solicitada, o usuário pode encerrar a aplicação digitando 2, ou pode digitar 1 e coletar dados de outras páginas.

Por outro lado, na visão do programador, o código apresenta alguns passos a mais. Na Figura 3, tem-se um fragmento do código que faz a conexão com o Facebook. O código completo está disponibilizado no Apêndice A. A variável “fbCliente” recebe uma instância de “DefaultFacebookClient” e para que a conexão seja feita, um dos parâmetros é “accessToken”, variável que contém o texto referente à Chave Secreta do Aplicativo, que

foi apresentada na seção 3.1. A linha seguinte carrega a página, passando como parâmetro a identificação da página desejada e por fim, na última linha, a variável "postFeed" recebe os dados das publicações da página.

Figura 3: Código para conexão com o Facebook e acesso à página desejada

```
FacebookClient fbClient = new DefaultFacebookClient(accessToken,
                                                    Version.VERSION_2_10);
Page page = fbClient.fetchObject(pageName, Page.class);
Connection<Post> postFeed = fbClient.fetchConnection(page.getId()+
                                                    "/posts", Post.class);
```

O que se deseja obter é um objeto da classe "Post", correspondente a cada uma das publicações da página. Para isso, ao percorrer a variável "postFeed", encontra-se algumas listas de "Post", e percorrendo cada uma dessas listas, obtém-se as publicações. Para isso, tem-se o código que possui dois laços de repetição "*for each*", conforme representado pela Figura 4.

Figura 4: Código para obter todas as publicações

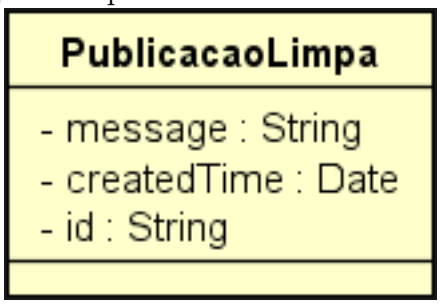
```
//Percorre "postFeed" obtendo uma lista de publicações
for (List<Post> postPage : postFeed){
    //Percorre a lista de publicações pegando post por post
    for (Post post : postPage){
        //...
    }
}
```

Como o objetivo aqui é recuperar dados para posteriormente fazer uma análise de sentimento sobre cada uma das mensagens publicadas, o primeiro passo que é feito dentro desse comando de repetição é verificar se a mensagem é nula. Caso seja, o código não faz nada na iteração.

3.3.2 Criação da Classe Publicação Limpa

Apesar de um objeto da classe "Post" possuir diversos atributos, a maioria deles não será usada no contexto aqui descrito. Então, foi criada uma nova classe chamada de "PublicacaoLimpa" (Figura 5) que contém apenas três atributos: o primeiro corresponde à mensagem da publicação, o segundo indica a data em que a publicação foi criada, o terceiro é apenas a ID da publicação.

Figura 5: Diagrama representando a Classe Publicação Limpa



Em cada iteração do comando de repetição presente na Figura 4, caso o conteúdo da mensagem não seja vazio, uma instância da classe "PublicacaoLimpa" é criada. Na sequência, com ajuda da biblioteca GSON, transforma-se essa instância em um texto com notação JSON que é armazenado em um arquivo cuja extensão é JSON.

3.3.3 Arquivos gerados

Para os dois arquivos gerados pela aplicação, foi necessária a conversão do objeto para o formato JSON. Isso se dá pelo fato de ser interessante que cada linha contenha uma publicação. Ao converter uma *String* contendo o texto da publicação em uma *String* em formato JSON, algumas características como quebra de linha, são representadas no arquivo como "\n".

Sendo assim, o arquivo em formato texto, cujo conteúdo é apenas a mensagem da publicação possui em cada linha uma mensagem diferente. Por outro lado, o arquivo em formato JSON contém o conteúdo da classe mencionada na seção anterior após sua conversão para a notação. A Figura 6 exemplifica esse formato. Pode-se observar que os atributos da classe são sucedidos pelo símbolo ":" (dois pontos), que por sua vez é acompanhado do conteúdo daquele atributo entre aspas e cada atributo é separado por vírgula. Vale ressaltar também que a linha começa com "{" (abre chave) e termina com "}" (fecha chave).

Figura 6: Formato JSON para a Classe Publicação Limpa

```
{"mensagem":"","createdTime":"","Id":""}
```

3.4 Coleta de dados de um usuário do Twitter

Coletas de dados na OSN Twitter podem ser feitas de diferentes maneiras. Pode-se pesquisar por publicações com determinadas palavras ou frases; pode-se pesquisar diretamente pela identificação perfil desejado; ou até mesmo por publicações em uma determinada região.

Analogamente às recuperações feitas no Facebook, nesta subseção a coleta de dados do Twitter será feita com o usuário informando a identificação do perfil desejado. Pela URL da página do perfil alvo, essa identificação está logo após o *link* do Twitter, como pode ser observado na Figura 7. Já a Figura 8 destaca com um retângulo arredondado que a identificação pode ser encontrada logo abaixo do nome. Vale ressaltar que na aplicação deve-se inserir essa identificação sem a utilização do símbolo arroba (@). Tanto a Figura 7 quanto a Figura 8 são capturas de tela do perfil do Bill Gates no Twitter, e a identificação desse usuário é, portanto, "BillGates".

Figura 7: Identificação de um usuário do Twitter

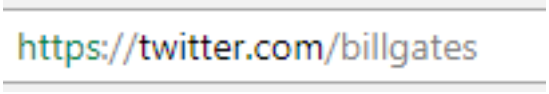


Figura 8: Identificação de um usuário do Twitter pela URL

3.4.1 Construção da aplicação

Na visão do usuário, a aplicação solicita que se identifique qual é o perfil no qual a coleta deverá ser feita. A API do Twitter limita esse tipo de coleta de dados e é possível que a aplicação recupere apenas cerca de 3.200 dos *tweets* mais recentes do perfil desejado. Caso o perfil desejado contenha menos do que o limite, a aplicação recolhe todas as

publicações. Ao final, dois arquivos são gerados e eles possuem o mesmo nome da identificação do perfil do Twitter, porém com extensões diferentes. O primeiro arquivo possui extensão ".json", e, portanto, cada linha deste arquivo contém um texto em formato JSON representando uma publicação. O segundo arquivo tem a extensão ".txt" e cada uma das suas linhas contém o texto referente a uma publicação.

Já na visão do programador, o código possui diversas etapas, sendo que a primeira delas é a validação das chaves de acesso obtidas. A Figura 9 refere-se a um trecho do código de coleta de dados de um determinado usuário do Twitter, sendo que as variáveis entre parênteses correspondem às quatro chaves de acesso fornecidas pela *Application Management*. O código completo encontra-se no Apêndice B.

Figura 9: Código para conexão no Twitter usando as chaves de acesso

```
ConfigurationBuilder cb = new ConfigurationBuilder();
cb.setDebugEnabled(true).setOAuthConsumerKey(consumerKey)
        .setOAuthConsumerSecret(consumerSecret)
        .setOAuthAccessToken(accessToken)
        .setOAuthAccessTokenSecret(accessTokenSecret);
```

Em seguida, o padrão de projeto *Singleton*[6], com seu método "getInstance", é utilizado para se obter uma instância da classe Twitter, conforme apresentado na Figura 10.

Figura 10: Obtendo instância única do Twitter

```
Twitter twitter = new TwitterFactory(cb.build()).getInstance();
```

Usando a instância obtida, o método "getUserTimeline" é chamado, recebendo como um dos parâmetros o perfil no qual deseja-se fazer a coleta. Todas as publicações são armazenadas em uma lista que posteriormente é percorrida a fim de se armazenar os *status* nos arquivos.

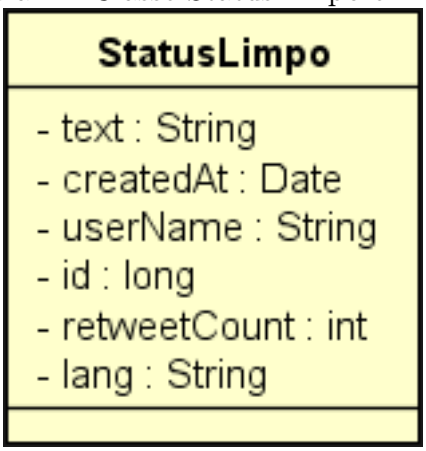
3.4.2 Criação da Classe *Status* Limpo

Assim como na coleta de dados do Facebook, no Twitter a classe "*Status*" possui diversos atributos que não serão utilizados neste trabalho, e que normalmente não são

[6]"Tem como definição garantir que uma classe tenha apenas uma instância de si mesma e que forneça um ponto global de acesso a ela". Fonte: https://www.devmedia.com.br/padrao-de-projeto-singleton-em-java/26392 Acesso em 23/11/2017

mesmo úteis quando se deseja avaliar puramente os textos publicados. Sendo assim, fez-se necessária a criação da classe "StatusLimpo". Conforme ilustrado na Figura 11, essa classe possui apenas 6 atributos, sendo eles: "*text*" que armazena o conteúdo do *status*, "*createdAt*" que indica a data de publicação, "*userName*" que representa o nome do autor da mensagem, enquanto "*id*", que corresponde ao número de identificação da publicação (um inteiro único). A variável "*retweetCount*", por sua vez, tem o valor de quantas vezes uma publicação foi compartilhada e, por fim, "*lang*" é a variável que contém o idioma da publicação.

Figura 11: Classe StatusLimpo em UML



3.4.3 Arquivos gerados

Os arquivos gerados por esta aplicação seguem o mesmo modelo apresentado na Subseção 3.3.3. No entanto, o conteúdo do arquivo de extensão ".json" possui um formato parecido, porém que corresponde aos atributos da classe "StatusLimpo", conforme ilustrado pela Figura 12. Faz-se necessário ressaltar que o valor correspondente às variáveis "id" e "retweetCount" não possuem aspas por se tratar de números inteiros.

Figura 12: Formato JSON para a classe StatusLimpo


```
{"text":"","createdAt":"","userName":"","id":0,"retweetCount":0,"lang":""}
```


3.5 Coleta de dados do Twitter a partir de uma localização ou um texto

Com relação à coleta de tweets a partir de um texto ou até mesmo publicações de uma determinada localização, grande parte do que foi exposto na Subseção 3.4 se encaixa neste contexto. O código completo sobre esta forma de coleta de dados está no Apêndice C.

Após obter a instância única do Twitter, é necessário implementar os métodos abstratos existentes em "*StatusListener*". Ao filtrar tweets por uma determinada localização o código deve se assemelhar ao trecho de código da figura 13. Pode-se observar que a variável "*locations*" é uma matriz do tipo "*double*".

Figura 13: Código para coleta de tweets em uma determinada região

```
twitterStream.addListener(listener);
FilterQuery filterQuery = new FilterQuery();
//those are the boundary from New York City
double[][] locations = {{-74,40}, {-73,41}};
filterQuery.locations(locations);
twitterStream.filter(filterQuery);
```

A cidade de Nova Iorque possui uma latitude aproximada de 40,714 e uma longitude aproximada de -74,006 (http://dateandtime.info). Essas informações foram utilizadas na Figura 13, como exemplo.

Além dessa forma, é possível coletar tweets que contenham uma determinada palavra ou texto. Isso pode ser visto na Figura 14, que se assemelha com a Figura 13, porém ao contrário da imagem anterior, esta possui uma variável do tipo "*String*" que, por sua vez, corresponde ao texto utilizado para filtrar as publicações.

Figura 14: Código para coleta de tweets que contenham um determinado texto

```
twitterStream.addListener(listener);
FilterQuery filterQuery = new FilterQuery();
String chave = "Bill Gates";
filterQuery.track(chave);
twitterStream.filter(filterQuery);
```

4 Análise de Dados

Nesta seção, será abordado o tema de análise dos dados, dividido em três subseções. A primeira trata da utilização de um método de análise de sentimento já existente. A segunda faz uma modificação no arquivo JSON para servir de entrada para outra ferramenta. A terceira subseção faz uma análise utilizando técnicas de aprendizado de máquina, com base no sentimento obtido na primeira subseção.

4.1 Análise de sentimento usando SentiStrength

Para a análise dos dados que foram obtidos como exemplo, foi utilizado o SentiStrenght[7] na versão 2.3. Conforme mencionado na Subseção 2.2, no estudo apresentado por Araújo et al. (2013), esse método de análise de sentimentos foi o que obteve segunda melhor taxa de acertos, sendo que sua abrangência foi cerca de 10 vezes maior do que o método dos Emoticons, que obteve melhor resultado. Uma captura da tela inicial do SentiStrength pode ser observada na Figura 15.

Figura 15: Captura da tela inicial do SentiStrength versão 2.3

O SentiStrength é uma ferramenta para análise de sentimento, porém por padrão esta ferramenta utiliza um dicionário próprio em inglês. É possível, através do próprio site, procurar e fazer o *download* do idioma pelo qual será feita sua análise de sentimento. Feito

[7]É um programa para Windows (versão gratuita) ou Java (versão comercial), que faz a análise de sentimento. Seu site oficial é: http://sentistrength.wlv.ac.uk/

esse *download*, é necessário registrar a nova pasta pela qual o *software* usará o dicionário, no caso, a pasta com o dicionário em português.

O motivo da escolha pelo português é pela sua grande presença na *web*. Segundo o site *Internet World Stats*[8], esse é o quinto idioma mais utilizado na rede. Hong, Convertino e Chi (2011) mencionam ainda que na OSN Twitter, o português é o terceiro idioma mais utilizado.

Os arquivos com extensão ".txt" mencionados nas Seções 3.3 e 3.4 também são gerados quando é feita uma coleta de dados buscando por um texto ou uma localização no Twitter, conforme Seção 3.5. Estes arquivos serão utilizados nesta etapa, principalmente pelo fato de que neles, cada linha possui apenas o texto referente a uma publicação.

Utilizando o SentiStrength, é possível analisar o sentimento de pequenas frases separadas ou então de um arquivo de texto. Em caso de análise em arquivo, serão analisadas linhas, e a frente dessa linha, separando-a com uma tabulação (\t), é colocado o quanto a frase é positiva, sendo que esse valor varia de 1 a 5. O valor 5 significa que a frase é muito positiva e o valor 1 significa que é pouco positiva. Logo em seguida, ainda na mesma linha e também separado por tabulação, é colocado o quanto a frase é negativa e essa avaliação estará entre -1 para pouco negativa e -5 para muito negativa.

O campo da análise de sentimento visa classificar os textos em uma das três possibilidades: positivo, negativo ou neutro. Tomando como base um estudo apresentado por Chalothorn e Ellman (2012), se o valor positivo atribuído ao trecho for maior que o módulo do valor negativo, então o trecho é positivo. Caso contrário, o trecho é negativo. E caso o módulo dos valores sejam iguais, o texto é neutro. Dessa forma, pode-se facilmente transformar a classificação do texto obtida através do SentiStrength em uma das três classificações da análise de sentimento.

4.2 Adicionar sentimento no arquivo JSON

Com o intuito de fazer a junção entre o arquivo no formato JSON e o arquivo gerado pelo SentiStrength, o qual contém um texto referente a uma publicação e também valores que representam o quanto o texto é positivo e negativo, outra aplicação foi criada. Esta aplicação encontra-se no Apêndice D.

Cada linha do arquivo gerado pelo SentiStrenght é recuperada. Essa linha é separada cada vez que encontra um "\t", ou seja, é separada três vezes. A primeira parte contém o texto da publicação, a segunda parte contém o quão positivo é o texto, e por fim, o quão negativo é o texto.

Conforme mencionado na Seção 3.2, a biblioteca Gson, além de transformar os dados de um objeto em um texto no formato JSON, também é capaz de converter um texto em

[8]Site apresenta estimativa dos 10 idiomas mais utilizados na internet. *Link:* http://www.internetworldstats.com/stats7.htm Acesso em 23 de novembro de 2017

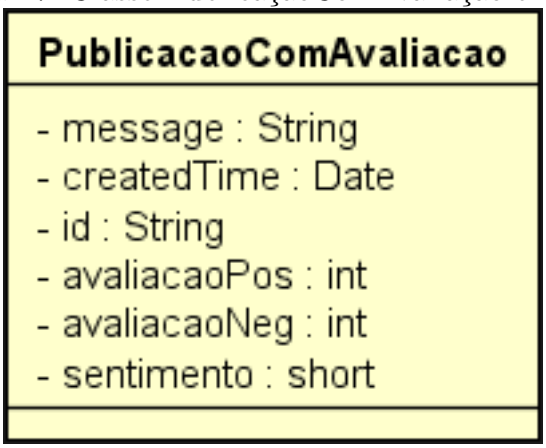
JSON em um objeto da classe que possua os atributos equivalentes. Para isso, conforme Figura 16, utilizou-se a variável "json" que contém uma instância da classe Gson. Nesta variável fez-se uma requisição ao método "fromJson" passando dois atributos. O primeiro é o texto no formato JSON que será convertido para a classe, e o segundo representa uma instância literal da classe para a qual o texto em JSON deve ser transformado. O retorno desse método é armazenado em uma variável da mesma classe.

Figura 16: Uso de Gson para transformar texto em JSON para um objeto equivalente em Java

```
publicacao = json.fromJson(linha, PublicacaoLimpa.class);
```

A Figura 17 é uma representação UML da classe "PublicaçãoComAvaliação", que contém não só os dados extraídos do Facebook, como também as avaliações do texto dessa publicação e uma variável para o sentimento. Os três primeiros atributos presentes nesta classe são os mesmos contidos na classe "PublicaçãoLimpa" (Figura 5). Os três últimos correspondem às avaliações positivas e negativas obtidas usando SentiStrength e também ao sentimento obtido com base nesta avaliação.

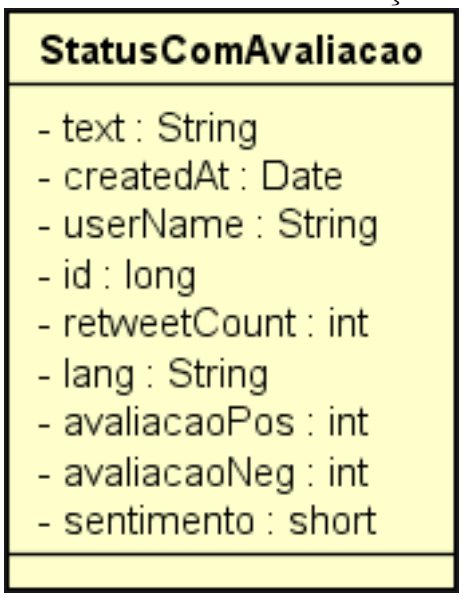
Figura 17: Classe PublicaçãoComAvaliação em UML



Para os dados extraídos do Twitter, a classe segue o mesmo raciocínio. Ou seja, possui os mesmos atributos da classe "StatusLimpo" (Figura 11) e também atributos correspondentes às avaliações e ao sentimento. Esta classe está representada na Figura 18.

Após obter os dados do arquivo de texto, e também os dados do arquivo JSON, faz-se uma junção desses dados na classe correspondente. Caso sejam dados extraídos do Facebook, a junção é adicionada a uma instância da classe "PublicacaoComAvaliacao". Já os dados recuperados do Twitter, são colocados em uma instância da classe "StatusComAvaliacao". Por fim, essa instância é convertida, com auxílio da biblioteca Gson, para um texto em formato JSON, que é armazenado em um novo arquivo.

Figura 18: Classe StatusComAvaliação em UML



4.3 Aprendizado de Máquina utilizando GraphLab

Nesta seção, são apresentados a linguagem utilizada pra esta etapa, o ambiente de programação utilizado, as bases de dados coletadas para treinamento e teste da rede neural e também o código utilizado explicando o que de fato representam.

4.3.1 Linguagem de programação e ambiente

Para o desenvolvimento desta etapa, foi utilizada a linguagem de programação Python e como ambiente de desenvolvimento, optou-se pelo uso da plataforma Jupyter Notebook, que é um projeto de código aberto, criado em 2014 e surgiu a partir do Ipython. Assim como mencionado por Pérez e Granger (2007), Ipython surgiu a partir da necessidade de se criar um sistema que permitisse a "rápida exploração algorítmica, análise de dados e visualização", tal como é feito pelo MatLab[9], por exemplo.

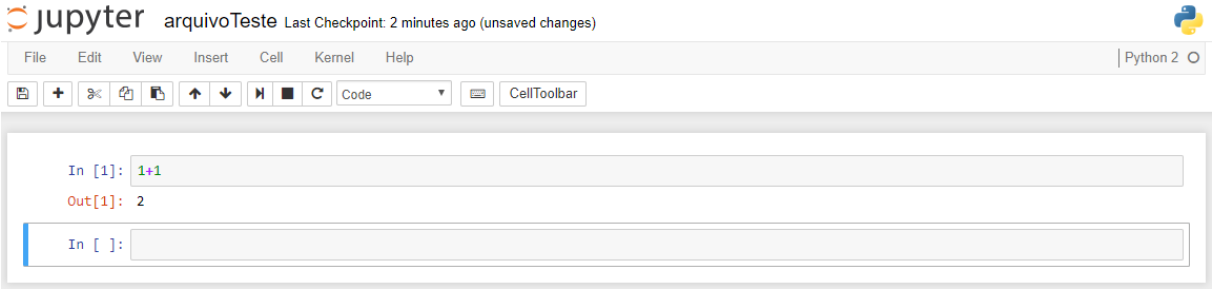
A Figura 19 é uma captura de tela do Jupyter, apenas para se ilustrar seu ambiente. Devido a isso, foi realizada uma simples soma e logo abaixo observa-se a saída para este cálculo.

4.3.2 Base de dados recuperada para teste

Faz-se necessário a obtenção de alguns dados, com intuito de utilizá-los para a análise de sentimento descrita nesta seção. Para obtenção desses dados, pelo fato da OSN Twitter restringir a coleta de dados de um determinado perfil, limitando a recuperação dos dados em uma quantidade próxima a 3.200 publicações, decidiu-se pela coleta dos dados

[9]"Trata-se de um software interativo de alta performance voltado para o cálculo numérico". Fonte: https://pt.wikipedia.org/wiki/MATLAB Acesso em 29/11/2017

Figura 19: Utilizando Jupyter Notebook para uma simples soma



contidos em páginas do Facebook. Foram selecionados três grandes clubes de futebol da capital mineira: América Futebol Clube, Clube Atlético Mineiro e Cruzeiro Esporte Clube, cujas identificações na rede social são, respectivamente, "americafcoficial", "atletico" e "cruzeirooficial". Os dados foram extraídos das publicações contidas nas páginas desses clubes.

Os dados relativos à página dos clubes foram coletados no dia 28/11/17. As publicações do América foram obtidas às 14h58min. Já os dados do Atlético, foram obtidos às 14h53min horas. Por fim, os dados do Cruzeiro foram obtidos no mesmo dia, porém às 14h56min.

Após a coleta, os dados foram submetidos ao SentiStrength. Esse software tem a capacidade de ler cada linha de um arquivo de texto e gerar como saída outro arquivo de texto contendo a linha original e o quanto o texto dessa linha é positiva e negativa. O nome do arquivo gerado pelo software é composto do nome do arquivo original com acréscimo de "+results.txt" no sufixo.

4.3.3 Jupyter Notebook

O início do código feito no Jupyter consiste na importação de algumas bibliotecas a serem utilizadas inicialmente. Dentre essas, vale ressaltar as bibliotecas referentes à importação do GraphLab Create[10], conforme indicado na Figura 20. Essa biblioteca, consiste em uma ferramenta para aprendizado de máquina, que fornece, já implementados, desde métodos para importação de dados em diferentes formatos, até métodos para treinamento da rede neural.

Figura 20: Trecho de código com a importação da biblioteca GraphLab

```
import graphlab
from graphlab import SFrame
```

Para leitura dos dados, fez-se necessário a utilização do SFrame, que é a parte de

[10]"Um mecanismo para criação rápida de produtos de dados em larga escala e de alto desempenho." Fonte: https://turi.com/products/create/docs/ Acesso em: 02/12/2017

engenharia dos dados presente na biblioteca GraphLab. O método "read_json" recebe dois parâmetros: o primeiro diz respeito ao nome do arquivo que está em formato JSON, e que será carregado; o segundo parâmetro corresponde à forma na qual os dados estão no arquivo e serão lidos pelo método. Como cada linha do arquivo possui um elemento JSON, será utilizado como "*lines*". Caso o arquivo inteiro possuísse um único elemento JSON, deveria ser utilizado "*records*". Os resultados de cada arquivo são armazenados na variável correspondente ao nome do clube em questão. Um exemplo disto pode ser observado na Figura 21.

Figura 21: Trecho do código que se refere à leitura dos arquivos JSON

```
america = SFrame.read_json('americafcoficialSenti.json', orient='lines')
atletico = SFrame.read_json('atleticoSenti.json', orient='lines')
cruzeiro = SFrame.read_json('cruzeirooficialSenti.json', orient='lines')
```

O comando presente na Figura 22 retorna quantas publicações foram recuperadas das páginas de cada Clube, uma vez que recupera o número de linhas de cada variável SFrame. Pode-se observar que ao executar o comando, logo abaixo, o Jupyter Notebook apresenta as saídas.

Figura 22: Trecho do código que imprime na tela quantas publicações existem em cada variável

```
print len(america)
print len(atletico)
print len(cruzeiro)
```

```
2879
3811
3755
```

Para treinamento e teste da rede, os dados foram divididos, assim como representado na Figura 23, onde para cada clube, foi utilizado um comando de separação aleatória e 80% dos dados foram designados para treino e 20% para teste.

Para fins de exemplo, apenas serão apresentados trechos de código referentes aos dados do América Futebol Clube, tendo em vista que os comandos utilizados são os mesmos e o que muda é apenas o sufixo da variável, que no caso do América, foi utilizado "Am", para o Atlético, foi utilizado "At" e para o Cruzeiro, "Cr".

Para treino da rede, utilizou-se o método "logistic_classifier" da biblioteca GraphLab, conforme apresentado na Figura 24. Para essa classificação, são necessários alguns parâmetros: o primeiro deles corresponde aos dados de treino, o segundo diz respeito a qual coluna contém o sentimento que será levado em conta no treino da rede. Na sequência

Figura 23: Divisão dos dados em treino e teste

```
treinoAm, testeAm = america.random_split(.8)
print len(treinoAm)
print len(testeAm)
```

```
2310
569
```

```
treinoAt, testeAt = atletico.random_split(.8)
print len(treinoAt)
print len(testeAt)
```

```
3088
723
```

```
treinoCr, testeCr = cruzeiro.random_split(.8)
print len(treinoCr)
print len(testeCr)
```

```
3007
748
```

leva-se em conta qual coluna contém os dados atrelados a esse sentimento. Vale ressaltar que assim como apresentado por Guestrin e Fox (2015a), utilizamos como terceiro parâmetro, a coluna denominada "contaPalavras", que foi obtida levando em conta as palavras de uma publicação e quantas vezes elas se repetem. Sendo assim, a coluna "contaPalavras" possui cada uma das palavras de uma publicação e o número correspondente à quantidade de vezes essa palavra se repete. O parâmetro seguinte diz respeito aos dados que poderiam ser utilizados para validação da rede. No entanto esse parâmetro foi colocado como *None*[11], simplesmente pelo fato de que a utilização os dados de teste foi feita posteriormente. O último atributo passado como parâmetro para essa função, diz respeito ao máximo de iterações feitas pela rede, esse valor foi colocado como 200 para os dados dos três clubes. O resultado obtido é armazenado na variável "sentiment_model_Am".

A saída obtida pelo treinamento da rede pode ser observado na Figura 25. No caso dos dados do América, pode ser observado que foi obtida uma solução ótima e uma acurácia de treino de 0,995671.

Para facilitar a visualização, a Figura 26 apresenta a obtenção de três linhas presen-

[11] Termo em inglês que quer dizer nenhum. Em Python, atribuir o valor *None* a uma variável, faz com que essa variável retorne ao seu estado inicial

Figura 24: Utilizando dados de treino para treinar a rede

```
sentiment_model_Am = graphlab.logistic_classifier.create(treinoAm,
                                                         target = 'sentimento',
                                                         features=['contaPalavras'],
                                                         validation_set=None,
                                                           max_iterations=200)
```

Figura 25: Saída obtida para o treinamento da rede

Iteration	Passes	Step size	Elapsed Time	Training-accuracy
1	3	0.000433	1.097683	0.754978
2	5	1.000000	1.193169	0.940260
3	6	1.000000	1.248538	0.956710
4	7	1.000000	1.302431	0.974892
5	8	1.000000	1.361382	0.983983
6	9	1.000000	1.423476	0.985281
11	14	1.000000	1.699271	0.994372
25	28	1.000000	2.463963	0.995671
50	62	0.500000	4.335265	0.995671
51	63	0.500000	4.413125	0.995671
75	100	1.000000	5.989572	0.995671

SUCCESS: Optimal solution found.

tes na variável que contém os dados de teste. Esses três dados foram armazenados em uma variável intitulada “sample_test_data_Am”. Ainda na Figura 26, pode-se observar que os três valores obtidos começam a partir da linha 31. Esse número foi totalmente aleatório, podendo-se pegar qualquer um dos dados presentes no teste.

A Figura 27 apresenta a previsão de sentimento para os três dados de teste. Para isso, utilizou-se a variável sentiment_model_Am que contém a rede treinada com os dados de treino e foi preciso chamar o método “predict_topk” passando três atributos. O primeiro atributo é relativo ao dados de teste, que neste caso foi utilizado a sample_test_data_Am. No segundo parâmetro, é necessário indicar a saída que se deseja obter. Neste caso, a saída desejada é a probabilidade e por fim, é necessário dizer quantas classes possui a rede que está sendo criada. Neste caso, por se tratar de análise de sentimento, são três classes, positivo (1), negativo (-1) e neutro (0). É possível observar que para cada uma das publicações, é apresentado a probabilidade de ser da classe 0, da classe 1 e da classe -1. Sendo assim, nessa rede, a primeira publicação tem probabilidade de 0,979886 de ser

Figura 26: Visualização de três itens dos dados de testes

```
sample_test_data_Am = testeAm[31:34]
sample_test_data_Am
```

id	avaliacaoNeg	avaliacaoPos	createdTime	mensagem	sentimento
151533471640669_123835736 6291602 ...	-1	1	Aug 5, 2017 6:22:25 PM	Hoje o América faz sua estreia na terceira e ...	0
151533471640669_123752811 6374527 ...	-1	2	Aug 4, 2017 9:30:56 PM	Bola rolando em Maceió para América x CRB! ...	1
151533471640669_123742943 3051062 ...	-2	1	Aug 4, 2017 6:33:01 PM	Acompanhe todos os jogos do #Coelho pelo Premiere ...	-1

contaPalavras
{'em': 1, 'etapa': 1, 'liga': 1, 'ao': 1, ' ...
{'pracimadelescoelho': 1, '\xf0\x9f\x90\xb0\xf0 ...
{'canal': 1, 'do': 1, 'claro': 1, 'e': 2, ...

[3 rows x 7 columns]

neutra. Para a segunda publicação, pode-se dizer que há uma incerteza, entre a publicação ser neutra (0,567093) e positiva (0,432897). E por fim a última publicação tem 0,988996 de probabilidade de ser negativa.

Figura 27: Fazendo previsão de sentimento para os três dados de treino

```
scores_Am = sentiment_model_Am.predict_topk(sample_test_data_Am, output_type='probability', k=3)
print scores_Am
```

```
+----+-------+-------------------+
| id | class |    probability    |
+----+-------+-------------------+
| 0  |   0   |   0.979886687186  |
| 0  |   -1  |  0.0199508180868  |
| 0  |   1   | 0.000162494727348 |
| 1  |   0   |   0.567093286246  |
| 1  |   1   |   0.432897022983  |
| 1  |   -1  | 9.69077129898e-06 |
| 2  |   -1  |   0.988996911719  |
| 2  |   0   |  0.0110018698173  |
| 2  |   1   | 1.21846383594e-06 |
+----+-------+-------------------+
[9 rows x 3 columns]
```

Comparando-se as previsões feitas pela rede com os resultados obtidos pela classificação do SentiStrength (coluna "sentimento" da Figura 26), pode-se observar que de fato, a rede acertou ao prever o sentimento da primeira e da terceira publicação. A segunda publicação, por sua vez, obteve pouco mais de 50% de probabilidade de ser neutra, enquanto o SentiStrength avaliou como sendo positiva.

Para o cálculo da acurácia, foi necessário a criação de uma função. Como atributos, essa função necessita de uma rede treinada (variável "model"), os dados de teste (variável "data") e por fim, a coluna que possui o sentimento obtido pelo SentiStrength. Essa função

está representada na Figura 28. A primeira tarefa desta função é obter as previsões, de acordo com o modelo, dos dados de teste passados.

Figura 28: Função para fazer o cálculo da acurácia

```
def get_classification_accuracy(model, data, true_labels):
    # First get the predictions
    predictions = model.predict(data)

    # Compute the number of correctly classified examples
    correctly_classified = data[true_labels == predictions]
    number_correctly_classified = len(correctly_classified)
    print number_correctly_classified

    # Then compute accuracy by dividing num_correct by total number of examples
    total_examples = len(data)
    accuracy = number_correctly_classified / total_examples

    return accuracy
```

Segundo Mikhail e Ackermann (1976 apud MONICO et al., 2009), a acurácia é "o grau de proximidade de uma estimativa com seu parâmetro (ou valor verdadeiro)". O cálculo da acurácia pode ser obtido pela divisão do número de dados corretamente classificados pelo número total de dados. Devido a isso, o segundo passo da função apresentada na Figura 28, é obter quantos dados foram corretamente classificados e depois de fazer essa divisão, a função retorna a acurácia.

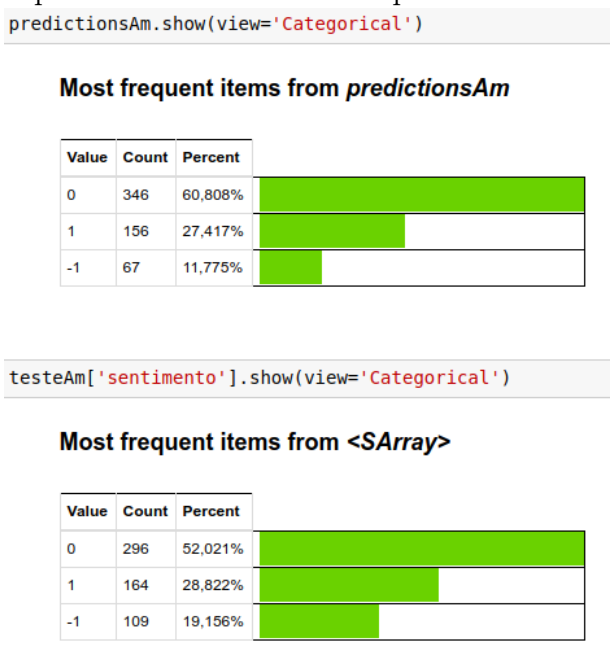
A Figura 29 apresenta o código para a função para cálculo da acurácia, passando dados dos três clubes. A acurácia obtida para o América foi de 0,6326, para o Atlético, 0,6473 e para o Cruzeiro, 0,6697.

Figura 29: Acurácia obtida para os dados de América, Atlético e Cruzeiro

```
get_classification_accuracy(sentiment_model_Am, testeAm, testeAm['sentimento'])
360
0.632688927943761
get_classification_accuracy(sentiment_model_At, testeAt, testeAt['sentimento'])
468
0.6473029045643154
get_classification_accuracy(sentiment_model_Cr, testeCr, testeCr['sentimento'])
501
0.6697860962566845
```

Os gráficos presentes na Figura 30, indicam o sentimento para os dados do América. O primeiro gráfico está relacionado com a quantidade de publicações cujos sentimentos foram previsto para cada uma das classes. O segundo gráfico, por sua vez, apresenta a quantidade de publicações cujos sentimentos foram obtidos através do SentiStrength. Pode-se notar que pouco mais de 52% dos dados de treino foram julgados como neutros pelo SentiStrength, enquanto nos dados treinados pela rede, esse valor chegou a quase 61%. Os dados avaliados como positivo foram quase 29%, pelo SentiStrength, que avaliou pouco mais de 19% desses dados como sendo negativos. Já os dados previstos pelo modelo treinado indicam que pouco mais de 27% dos dados foram previstos como positivos e cerca de 12% foram avaliados como sendo negativos.

Figura 30: Gráfico apresentando o sentimento para os dados de teste do América



Os gráficos presentes na Figura 31, por sua vez, apresentam dados sobre sentimento das publicações do Atlético. O sentimento obtido pelo SentiStrength foi de 46,75% dos dados como neutros, cerca de 32% negativos e pouco mais de 21% foram avaliados como positivos. Já pelo modelo treinado pelos dados do Atlético, o que se obteve foi quase 51,6% dos dados como neutros, mais de 31% como negativos e cerca de 17% como positivos.

Os dados do Cruzeiro por sua vez, em se tratando da avaliação do SentiStrength obteve pouco mais de 56,5% como sendo neutros, cerca de 30,5% dos dados como sendo positivo e quase 13% dos dados como sendo negativos. Já a avaliação obtida pelo treinamento dos dados, apresentou mais de 64% dos dados como neutros, pouco mais de 27% como positivos e pouco mais de 8% como sendo negativos.

Figura 31: Gráfico apresentando o sentimento para os dados de teste do Atlético

```
predictionsAt.show(view='Categorical')
```

Most frequent items from *predictionsAt*

Value	Count	Percent
0	373	51,591%
-1	226	31,259%
1	124	17,151%

```
testeAt['sentimento'].show(view='Categorical')
```

Most frequent items from <*SArray*>

Value	Count	Percent
0	338	46,75%
-1	232	32,089%
1	153	21,162%

Figura 32: Gráfico apresentando o sentimento para os dados de teste do Cruzeiro

```
predictionsCr.show(view='Categorical')
```

Most frequent items from *predictionsCr*

Value	Count	Percent
0	482	64,439%
1	204	27,273%
-1	62	8,289%

```
testeCr['sentimento'].show(view='Categorical')
```

Most frequent items from <*SArray*>

Value	Count	Percent
0	423	56,551%
1	229	30,615%
-1	96	12,834%

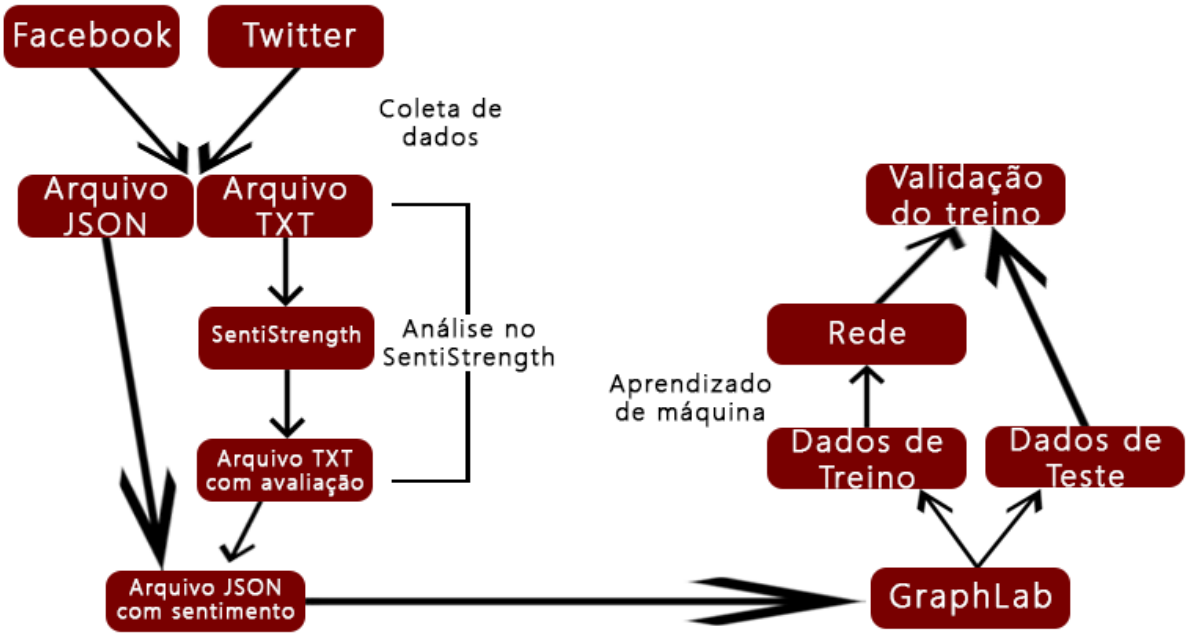
5 Conclusão e Trabalhos Futuros

Apesar da existência de muitos trabalhos publicados na área de coleta de dados em redes sociais, poucos tratam do assunto um pouco mais voltados para a prática. Neste trabalho, uma metodologia para coleta e análise de dados foi desenvolvida e utilizada, apresentando assim algumas alternativas práticas para a coleta destes dados em duas grandes redes sociais online, o Facebook e o Twitter.

Já com relação à análise dos dados, voltada principalmente para a área da análise

de sentimento, existem também diversos estudos, mas observa-se que análise de dados em OSNs, publicados em português é uma área ainda pouco explorada.

Figura 33: Visão geral da Metodologia desenvolvida neste trabalho



A figura 33 é uma representação de toda a metodologia de coleta e análise de dados de redes sociais utilizada neste trabalho. Primeiramente, tem-se a extração dos dados das redes sociais onde são gerados dois arquivos, um contendo apenas a publicação, em formato de texto e outro contendo os dados da publicação em formato JSON. Para a análise de sentimento da publicação, foi utilizado o software SentiStrength, que aponta o quanto cada publicação é positiva, com valores entre 1 e 5, e o quando a publicação é negativa, com valores que variam de -1 a -5. Na sequência, pega-se esses valores de positivo e negativo e adiciona-os em outro arquivo em formato JSON. Feito isso, com utilização da biblioteca GraphLab, os dados são divididos em treino e teste. Como o próprio nome indica, os dados de treino são utilizados para treinar a rede neural. Por fim, são colocados nessa rede com o intuito de prever o sentimento, cada uma das publicações colocadas como teste.

Com o desenvolvimento e apresentação da metodologia, e com o detalhamento que foi feito neste trabalho do uso de todo o ferramental envolvido, espera-se ter contribuído para que novos pesquisadores da área possam rapidamente ter uma primeira experiência completa de coleta e análise de dados em redes sociais, uma vez que os principais conceitos envolvidos e exemplos de utilização na prática foram mostrados.

Como trabalhos futuros, propõe-se uma melhora nos códigos de coleta de dados nas redes sociais, e também uma integração desses códigos, de maneira que o façam parte de um único projeto em Java e consiga ser feito a coleta tanto no Facebook quanto no twitter

sem a necessidade de ter que carregar mais de um projeto.

Outra proposta é o uso dessa metodologia para se analisar uma massa maior de dados, além de se usar mais métodos presentes na biblioteca Graphlab, obtendo-se dados relevantes em determinado contexto.

Referências

ARAÚJO, M. et al. Métodos para análise de sentimentos no twitter. In: *Proceedings of the 19th Brazilian symposium on Multimedia and the Web (WebMedia'13)*. [S.l.: s.n.], 2013.

BATISTA, G. E. d. A. P. et al. *Pré-processamento de dados em aprendizado de máquina supervisionado*. Tese (Doutorado) — Universidade de São Paulo, 2003.

BAUSCH, S.; MCGIBONEY, M. Social networks and blogs now 4th most popular online activity, ahead of personal email. nielsen reports. *New York: Nielsen Online. Prieiga per internetą: http://www.nielsen.com/us/en/press-room/2009/social-networks–.html*, 2009.

BENEVENUTO, F.; ALMEIDA, J.; SILVA, A. Coleta e análise de grandes bases de dados de redes sociais online. *Jornadas de Atualização em Informática (JAI)*, p. 11–57, 2011.

CHALOTHORN, T.; ELLMAN, J. Sentiment analysis of web forums: Comparison between sentiwordnet and sentistrength. In: THE 4TH INTERNATIONAL CONFERENCE ON COMPUTER TECHNOLOGY AND DEVELOPMENT (ICCTD 2012). 24-25 NOVEMBER 2012. [S.l.], 2012.

COLLINS-THOMPSON, K. *Applied Machine Learning in Python*. 2017. <https://www.coursera.org/learn/python-machine-learning/>. Acessado em 28/11/17.

ELLISON, N. B. et al. Social network sites: Definition, history, and scholarship. *Journal of Computer-Mediated Communication*, Wiley Online Library, v. 13, n. 1, p. 210–230, 2007.

GONÇALVES, P. et al. Bazinga! caracterizando e detectando sarcasmo e ironia no twitter. *Brazilian Workshop on Social Network Analysis and Mining (BraSNAM)*, 2015.

GUESTRIN, C.; FOX, E. *Fundações do aprendizado de máquina: uma abordagem por estudo de caso*. 2015a. <https://www.coursera.org/learn/ml-foundations/>. Acessado em 28/11/17.

GUESTRIN, C.; FOX, E. *Machine Learning: Classification*. 2015b. <https://www.coursera.org/learn/ml-classification/>. Acessado em 28/11/17.

HAYKIN, S. *Redes neurais: princípios e prática*. [S.l.]: Bookman Editora, 2007.

HONG, L.; CONVERTINO, G.; CHI, E. H. Language matters in twitter: A large scale study. In: *ICWSM*. [S.l.: s.n.], 2011.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. *Nature*, Nature Research, v. 521, n. 7553, p. 436–444, 2015.

MALHEIROS, Y. Emotte: Uma ferramenta de análise de sentimentos para o twitter. In: *XX Brazilian Symposium on Multimedia and the Web-Webmedia*. [S.l.: s.n.], 2014. v. 2014.

MARTINS, R.; PEREIRA, A.; BENEVENUTO, F. Uma abordagem para análise de sentimentos de aplicações da web em língua portuguesa. *Brazilian Symposium on Multimedia and the Web. (Webmedia)*, 2015.

MIKHAIL, E. M.; ACKERMANN, F. E. Observations and least squares. Harper and Row, 1976.

MONARD, M. C.; BARANAUSKAS, J. A. Conceitos sobre aprendizado de máquina. *Sistemas Inteligentes-Fundamentos e Aplicações*, v. 1, n. 1, 2003.

MONICO, J. F. G. et al. Acurácia e precisão: revendo os conceitos de forma acurada. *Boletim de Ciências Geodésicas*, Universidade Federal do Paraná, v. 15, n. 3, 2009.

PÉREZ, F.; GRANGER, B. E. IPython: a system for interactive scientific computing. *Computing in Science and Engineering*, IEEE Computer Society, v. 9, n. 3, p. 21–29, maio 2007. ISSN 1521-9615. Disponível em: <http://ipython.org>.

Apêndice A Código para coleta de publicações das páginas Facebook

Classe intitulada “PostDePagina”, que contém o “main”:

```
package postdepagina;

import java.util.Scanner;
/**
 *
 * @author dougl
 */
public class PostDePagina {
    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        int aux=1;
        while (aux==1){
            String pageName = null;
            Scanner scanner = new Scanner( System.in );
            System.out.println("Digite o id da página: ");
            pageName = scanner.nextLine();

            ColetarFacebook conecta = new ColetarFacebook(pageName);

            conecta.recuperaPublicacoes();

            System.out.println("Outra Página?\n1-SIM\n2-Não");
            aux = scanner.nextInt();
        }
    }
}
```

Classe “PublicacaoLimpa”:

```
package postdepagina;

import java.util.Date;
/**
 *
 * @author dougl
 */
public class PublicacaoLimpa {
    private String message;
    private Date createdTime;
    private String Id;

    public PublicacaoLimpa(String mensagem, Date createdTime, String Id) {
        this.message = mensagem;
        this.createdTime = createdTime;
        this.Id = Id;
    }

    public String getMensagem() {
        return message;
    }

    public void setMensagem(String mensagem) {
        this.message = mensagem;
    }

    public Date getCreatedTime() {
        return createdTime;
    }

    public void setCreatedTime(Date createdTime) {
        this.createdTime = createdTime;
    }

    public String getId() {
        return Id;
    }
```

```
    public void setId(String Id) {
        this.Id = Id;
    }
}
```

Classe “Json”:

```
package postdepagina;

import com.google.gson.Gson;
import com.google.gson.GsonBuilder;
/**
 *
 * @author dougl
 */
public class Json {
    public Json() {}

    public String publicacaoParaJson(PublicacaoLimpa post){
        String json = new String();
        Gson gson = new GsonBuilder().create();

        json = gson.toJson(post);

        return json;
    }

    public String stringParaJson(String post){
        String json = new String();
        Gson gson = new GsonBuilder().create();

        json = gson.toJson(post);

        return json;
    }
}
```

Classe “ManipularArquivo”

```
package postdepagina;

import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.util.logging.Level;
import java.util.logging.Logger;
/**
 *
 * @author dougl
 */
public class ManipularArquivo {
    private String nomeDaPagina;
    private String nomeArquivoJson, nomeArquivoTxt;

    public ManipularArquivo(String nomeDaPagina) {
        this.nomeDaPagina = nomeDaPagina;
        nomeArquivoJson = nomeDaPagina + ".json";
        nomeArquivoTxt = nomeDaPagina + ".txt";
        inicializaArquivo();
    }

    private void inicializaArquivo(){
        try {
            FileWriter arquivoJSON = new FileWriter(nomeArquivoJson);
            FileWriter arquivoTXT = new FileWriter(nomeArquivoTxt);
            BufferedWriter gravarJSON = new BufferedWriter(arquivoJSON);
            BufferedWriter gravarTXT = new BufferedWriter(arquivoTXT);
            gravarJSON.close();
            arquivoJSON.close();
            //Primeira linha deve ser vazia pois o SentiStrength a ignora
            gravarTXT.write("\n");
            gravarTXT.close();
            arquivoTXT.close();
        } catch (IOException ex) {
            Logger.getLogger(ManipularArquivo.class.getName())
                    .log(Level.SEVERE, null, ex);
        }
```

```
    }

    public void escreveArquivo(PublicacaoLimpa post){
        Json converter = new Json();
        String postEmJson = converter.publicacaoParaJson(post);
        String publicacaoEmJson = converter.stringParaJson(post.getMensagem());

        try {
            FileWriter arquivoJSON = new FileWriter(nomeArquivoJson, true);
            FileWriter arquivoTXT = new FileWriter(nomeArquivoTxt, true);
            BufferedWriter gravarJSON = new BufferedWriter(arquivoJSON);
            BufferedWriter gravarTXT = new BufferedWriter(arquivoTXT);
            gravarJSON.write(postEmJson + "\n");
            gravarJSON.close();
            arquivoJSON.close();
            gravarTXT.write(publicacaoEmJson + "\n");
            gravarTXT.close();
            arquivoTXT.close();
        } catch (IOException ex) {
            Logger.getLogger(ManipularArquivo.class.getName())
                    .log(Level.SEVERE, null, ex);
        }
    }
}
```

Classe "ColetarFacebook":

```
package postdepagina;

import com.restfb.Connection;
import com.restfb.DefaultFacebookClient;
import java.util.List;
import com.restfb.FacebookClient;
import com.restfb.Version;
import com.restfb.types.Page;
import com.restfb.types.Post;
/**
 *
 * @author dougl
 */
```

```
public class ColetarFacebook {
    private String accessToken = /*SUA CHAVE DE ACESSO*/;
    private FacebookClient fbClient;
    private Page pagina;
    private Connection<Post> postFeed;
    private String nomeDaPagina;

    public ColetarFacebook(String nomeDaPagina) {
        this.nomeDaPagina = nomeDaPagina;
        recuperarPagina();
    }

    private void recuperarPagina(){
        fbClient = new DefaultFacebookClient(accessToken, Version.VERSION_2_10);
        pagina = fbClient.fetchObject(nomeDaPagina, Page.class);
        postFeed = fbClient.fetchConnection(pagina.getId()+"/posts", Post.class);
    }

    public void recuperaPublicacoes(){
        PublicacaoLimpa publicacao;
        ManipularArquivo arquivos = new ManipularArquivo(nomeDaPagina);
        //Percorre "postFeed" obtendo uma lista de publicações
        for(List<Post> postPage : postFeed){
            //Percorre a lista de publicações pegando post por post
            for (Post post : postPage){
                if (post.getMessage() == null){
                    /* Se mensagem for nula, então não é uma publicação.
                     * Pode ser apenas um evento (Nasceu em dd/mm/yyyy).
                     * Então não faz nada.*/
                }else{
                    publicacao = new PublicacaoLimpa(post.getMessage(),
                                                     post.getCreatedTime(),
                                                     post.getId());
                    arquivos.escreveArquivo(publicacao);
                }
            }
        }
    }
}
```

Apêndice B Código para coleta de tweets de um determinado usuário do Twitter

Classe "TweetsDeUsuário", que contém o "main":

```
package tweetsdeusuario;

import java.util.Scanner;
/**
 *
 * @author dougl
 */
public class TweetsDeUsuario {
    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        int aux = 1;
        while(aux == 1){
            Scanner scanner = new Scanner( System.in );
            System.out.print("Digite a identificação do usuário do Twitter: ");
            String nomeUsuario = scanner.nextLine();

            ColetaTwitter coleta = new ColetaTwitter(nomeUsuario);

            coleta.recuperaTweets();

            System.out.print("Outra Página?\n1-SIM\n2-Não");
            aux = scanner.nextInt();
        }
    }
}
```

Classe "StatusLimpo":

```
package tweetsdeusuario;

import java.util.Date;
/**
 *
 * @author dougl
```

```
 */
public class StatusLimpo {
    public String text;
    public Date createdAt;
    public String userName;
    public long id;
    public int retweetCount;
    public String lang;

    public StatusLimpo(String text, Date createdAt, String userName,
                long id, int retweetCount, String lang) {
        this.text = text;
        this.createdAt = createdAt;
        this.userName = userName;
        this.id = id;
        this.retweetCount = retweetCount;
        this.lang = lang;
    }

    public String getText() {
        return text;
    }

    public void setText(String text) {
        this.text = text;
    }

    public Date getCreatedAt() {
        return createdAt;
    }

    public void setCreatedAt(Date createdAt) {
        this.createdAt = createdAt;
    }

    public String getUserName() {
        return userName;
    }
```

```
    public void setUserName(String userName) {
        this.userName = userName;
    }

    public long getId() {
        return id;
    }

    public void setId(long id) {
        this.id = id;
    }

    public int getRetweetCount() {
        return retweetCount;
    }

    public void setRetweetCount(int retweetCount) {
        this.retweetCount = retweetCount;
    }

    public String getLang() {
        return lang;
    }

    public void setLang(String lang) {
        this.lang = lang;
    }
}
```

Classe “Json”:

```
package tweetsdeusuario;

import com.google.gson.Gson;
import com.google.gson.GsonBuilder;
/**
 *
 * @author dougl
 */
public class Json {
```

```
    public Json() {}

    public String publicacaoParaJson(StatusLimpo post){
        String json = new String();
        Gson gson = new GsonBuilder().create();

        json = gson.toJson(post);

        return json;
    }

    public String stringParaJson(String post){
        String json = new String();
        Gson gson = new GsonBuilder().create();

        json = gson.toJson(post);

        return json;
    }
}
```

Classe “ManipularArquivo”:

```
package tweetsdeusuario;

import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.util.logging.Level;
import java.util.logging.Logger;
/**
 *
 * @author dougl
 */
public class ManipularArquivo {
    private String nomeUsuario;
    private String nomeArquivoJson, nomeArquivoTxt;

    public ManipularArquivo(String nomeUsuario) {
        this.nomeUsuario = nomeUsuario;
```

```
    nomeArquivoJson = nomeUsuario + ".json";
    nomeArquivoTxt = nomeUsuario + ".txt";
    inicializaArquivo();
}

private void inicializaArquivo(){
    try {
        FileWriter arquivoJSON = new FileWriter(nomeArquivoJson);
        FileWriter arquivoTXT = new FileWriter(nomeArquivoTxt);
        BufferedWriter gravarJSON = new BufferedWriter(arquivoJSON);
        BufferedWriter gravarTXT = new BufferedWriter(arquivoTXT);
        gravarJSON.close();
        arquivoJSON.close();
        //Primeira linha deve ser vazia pois o SentiStrength a ignora
        gravarTXT.write("\n");
        gravarTXT.close();
        arquivoTXT.close();
    } catch (IOException ex) {
        Logger.getLogger(ManipularArquivo.class.getName())
                .log(Level.SEVERE, null, ex);
    }
}

public void escreveArquivo(StatusLimpo post){
    Json converter = new Json();
    String postEmJson = converter.publicacaoParaJson(post);
    String publicacaoEmJson = converter.stringParaJson(post.getText());

    try {
        FileWriter arquivoJSON = new FileWriter(nomeArquivoJson, true);
        FileWriter arquivoTXT = new FileWriter(nomeArquivoTxt, true);
        BufferedWriter gravarJSON = new BufferedWriter(arquivoJSON);
        BufferedWriter gravarTXT = new BufferedWriter(arquivoTXT);
        gravarJSON.write(postEmJson + "\n");
        gravarJSON.close();
        arquivoJSON.close();
        gravarTXT.write(publicacaoEmJson + "\n");
        gravarTXT.close();
        arquivoTXT.close();
```

```
        } catch (IOException ex) {
            Logger.getLogger(ManipularArquivo.class.getName())
                    .log(Level.SEVERE, null, ex);
        }
    }
}
```

Classe “ColetaTwitter”:

```
package tweetsdeusuario;

import java.util.ArrayList;
import java.util.List;
import twitter4j.Paging;
import twitter4j.Status;
import twitter4j.Twitter;
import twitter4j.TwitterException;
import twitter4j.TwitterFactory;
import twitter4j.conf.ConfigurationBuilder;
/**
 *
 * @author dougl
 */
public class ColetaTwitter {
    String consumerKey = /*Sua Consumer Key aqui*/;
    String consumerSecret = /*Sua Consumer Secret aqui*/;
    String accessToken = /*Seu Access Token aqui*/;
    String accessTokenSecret = /*Seu Access Token Secret aqui*/;
    ConfigurationBuilder cb;
    String nomeUsuario;
    Twitter twitter;

    public ColetaTwitter(String nomeUsuario) {
        this.nomeUsuario = nomeUsuario;
        cb = conectaTwitter();
    }

    private ConfigurationBuilder conectaTwitter(){
        ConfigurationBuilder config = new ConfigurationBuilder();
        config.setDebugEnabled(true)
```

```
                .setOAuthConsumerKey(consumerKey)
                .setOAuthConsumerSecret(consumerSecret)
                .setOAuthAccessToken(accessToken)
                .setOAuthAccessTokenSecret(accessTokenSecret);

        twitter = new TwitterFactory(config.build()).getInstance();

        return config;
    }

    public void recuperaTweets (){
        List<Status> status = new ArrayList();
        int pageno = 1;
        ManipularArquivo arquivos = new ManipularArquivo(nomeUsuario);

        while(true){
            try {
                int size = status.size();
                Paging page = new Paging(pageno++, 100);
                status.addAll(twitter.getUserTimeline(nomeUsuario, page));
                if (status.size() == size)
                    break;
            } catch (TwitterException ex) {
                ex.printStackTrace();
                System.out.println("Failed to get timeline: " + ex.getMessage());
                System.exit(-1);
            }
        }

        for (Status tweet : status){
            StatusLimpo statusClean = new StatusLimpo(tweet.getText(),
                        tweet.getCreatedAt(), tweet.getUser().getName(),
                        tweet.getId(), tweet.getRetweetCount(), tweet.getLang());

            arquivos.escreveArquivo(statusClean);
        }

    }
}
```

Apêndice C Código para coleta de tweets com base em palavras ou localização

Classe "PesquisarPorTweets", contém "main":

```
package pesquisaportweets;
/**
 *
 * @author dougl
 */
public class PesquisaPorTweets {
    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        ColetarTwitter coleta = new ColetarTwitter("resultadoColeta.json");
        coleta.procuraTweets();
    }
}
```

Classe "StatusLimpo":

```
package classificacaonojson;

import java.util.Date;

/**
 *
 * @author douglas
 */
public class StatusLimpo {
    public String text;
    public Date createdAt;
    public String userName;
    public long id;
    public int retweetCount;
    public String lang;

    public StatusLimpo(String text, Date createdAt, String userName, long id, int r
        this.text = text;
```

```
        this.createdAt = createdAt;
        this.userName = userName;
        this.id = id;
        this.retweetCount = retweetCount;
        this.lang = lang;
    }

    public String getText() {
        return text;
    }

    public void setText(String text) {
        this.text = text;
    }

    public Date getCreatedAt() {
        return createdAt;
    }

    public void setCreatedAt(Date createdAt) {
        this.createdAt = createdAt;
    }

    public String getUserName() {
        return userName;
    }

    public void setUserName(String userName) {
        this.userName = userName;
    }

    public long getId() {
        return id;
    }

    public void setId(long id) {
        this.id = id;
    }
```

```
    public int getRetweetCount() {
        return retweetCount;
    }

    public void setRetweetCount(int retweetCount) {
        this.retweetCount = retweetCount;
    }

    public String getLang() {
        return lang;
    }

    public void setLang(String lang) {
        this.lang = lang;
    }
}
```

Classe "Json":

```
package pesquisaportweets;

import com.google.gson.Gson;
import com.google.gson.GsonBuilder;
/**
 *
 * @author dougl
 */
public class Json {
    public Json() {}

    public String publicacaoParaJson(StatusLimpo post){
        String json = new String();
        Gson gson = new GsonBuilder().create();

        json = gson.toJson(post);

        return json;
    }

    public String stringParaJson(String post){
```

```
        String json = new String();
        Gson gson = new GsonBuilder().create();

        json = gson.toJson(post);

        return json;
    }
}
```

Classe “ManipularArquivo”:

```
package pesquisaportweets;

import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.util.logging.Level;
import java.util.logging.Logger;
/**
 *
 * @author dougl
 */
public class ManipularArquivo {
    private String nome;
    private String nomeArquivoJson, nomeArquivoTxt;

    public ManipularArquivo(String nomeArquivoResultado) {
        this.nome = nomeArquivoResultado;
        nomeArquivoJson = nomeArquivoResultado + ".json";
        nomeArquivoTxt = nomeArquivoResultado + ".txt";
        inicializaArquivo();
    }

    private void inicializaArquivo(){
        try {
            FileWriter arquivoJSON = new FileWriter(nomeArquivoJson);
            FileWriter arquivoTXT = new FileWriter(nomeArquivoTxt);
            BufferedWriter gravarJSON = new BufferedWriter(arquivoJSON);
            BufferedWriter gravarTXT = new BufferedWriter(arquivoTXT);
            gravarJSON.close();
```

```
            arquivoJSON.close();
            //Primeira linha deve ser vazia pois o SentiStrength a ignora
            gravarTXT.write("\n");
            gravarTXT.close();
            arquivoTXT.close();
        } catch (IOException ex) {
            Logger.getLogger(ManipularArquivo.class.getName())
                                                .log(Level.SEVERE, null, ex);
        }
    }

    public void escreveArquivo(StatusLimpo post){
        Json converter = new Json();
        String postEmJson = converter.publicacaoParaJson(post);
        String publicacaoEmJson = converter.stringParaJson(post.getText());

        try {
            FileWriter arquivoJSON = new FileWriter(nomeArquivoJson, true);
            FileWriter arquivoTXT = new FileWriter(nomeArquivoTxt, true);
            BufferedWriter gravarJSON = new BufferedWriter(arquivoJSON);
            BufferedWriter gravarTXT = new BufferedWriter(arquivoTXT);
            gravarJSON.write(postEmJson + "\n");
            gravarJSON.close();
            arquivoJSON.close();
            gravarTXT.write(publicacaoEmJson + "\n");
            gravarTXT.close();
            arquivoTXT.close();
        } catch (IOException ex) {
            Logger.getLogger(ManipularArquivo.class.getName())
                                                .log(Level.SEVERE, null, ex);
        }
    }
}
```

Classe “ColetarTwitter”:

```
package pesquisaportweets;

import twitter4j.FilterQuery;
import twitter4j.StallWarning;
```

```
import twitter4j.Status;
import twitter4j.StatusDeletionNotice;
import twitter4j.StatusListener;
import twitter4j.TwitterStream;
import twitter4j.TwitterStreamFactory;
import twitter4j.conf.ConfigurationBuilder;
/**
 *
 * @author dougl
 */
public class ColetarTwitter {
    String consumerKey = /*Sua Consumer Key aqui*/;
    String consumerSecret = /*Sua Consumer Secret aqui*/;
    String accessToken = /*Seu Access Token aqui*/;
    String accessTokenSecret = /*Seu Access Token Secret aqui*/;
    ConfigurationBuilder cb;
    String nomeArquivo;
    TwitterStream twitter;

    public ColetarTwitter(String nomeArquivo) {
        this.nomeArquivo = nomeArquivo;
        cb = conectaTwitter();
    }

    private ConfigurationBuilder conectaTwitter(){
        ConfigurationBuilder config = new ConfigurationBuilder();
        config.setDebugEnabled(true)
                .setOAuthConsumerKey(consumerKey)
                .setOAuthConsumerSecret(consumerSecret)
                .setOAuthAccessToken(accessToken)
                .setOAuthAccessTokenSecret(accessTokenSecret);

        twitter = new TwitterStreamFactory(config.build()).getInstance();

        return config;
    }

    public void procuraTweets(){
        ManipularArquivo arquivos = new ManipularArquivo(nomeArquivo);
```

```
//DEVE SER IMPLEMENTADO TODOS OS MÉTODOS ABSTRATOS
StatusListener listener = new StatusListener() {
    //Variável para limitar a busca (caso seja necessário);
    int contador = 0;
    @Override
    public void onStatus(Status status) {
        //Caso não seja necessário limitar a busca, basta comentar o if
        System.out.println(contador);
        if(contador >= 10000){
            return;
        }
        StatusLimpo novo = new StatusLimpo(status.getText(),
                                           status.getCreatedAt(),
                                           status.getUser().getName(),
                                           status.getId(),
                                           status.getRetweetCount(),
                                           status.getLang());
        arquivos.escreveArquivo(novo);
        contador++;
    }

    @Override
    public void onDeletionNotice(StatusDeletionNotice sdn) {}

    @Override
    public void onTrackLimitationNotice(int i) {}

    @Override
    public void onScrubGeo(long l, long l1) {}

    @Override
    public void onStallWarning(StallWarning sw) {}

    @Override
    public void onException(Exception excptn) {}
};
twitter.addListener(listener);
FilterQuery filterQuery = new FilterQuery();
```

```
        // Busca por tweets que contenham uma palavra:
        String chave = "Futebol"; //Palavra que será procurada
        filterQuery.track(chave);

        // Busca por tweets em uma localização:
        //double[][] locations = {{-74,40}, {-73,41}}; //Coordenadas são de NY
        //filterQuery.locations(locations);

        // Restante do código
        twitter.filter(filterQuery);
    }
}
```

Apêndice D Código para adicionar sentimento ao texto em formato JSON

Classe "AddSentimentoNoJson", que contém o "main":

```
package addsentimentonojson;

import java.util.Scanner;
/**
 *
 * @author dougl
 */
public class AddSentimentoNoJson {
    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        Scanner scanner = new Scanner( System.in );
        System.out.println("Digite o id da página: ");
        String nomeArquivo = scanner.nextLine();

        ManipularArquivo arquivo = new ManipularArquivo(nomeArquivo);
        arquivo.ler();
    }
}
```

Classe "PublicacaoLimpa":

```
package sentimentonojson;

import java.util.Date;

/**
 *
 * @author douglas
 */
public class PublicacaoLimpa {
    public String message;
    public Date createdTime;
    public String Id;
```

```
    public PublicacaoLimpa(String mensagem, Date createdTime, String Id) {
        this.message = mensagem;
        this.createdTime = createdTime;
        this.Id = Id;
    }

    public String getMensagem() {
        return message;
    }

    public void setMensagem(String mensagem) {
        this.message = mensagem;
    }

    public Date getCreatedTime() {
        return createdTime;
    }

    public void setCreatedTime(Date createdTime) {
        this.createdTime = createdTime;
    }

    public String getId() {
        return Id;
    }

    public void setId(String Id) {
        this.Id = Id;
    }
}
```

Classe "PublicacaoComAvaliacao":

```
package sentimentonojson;

import java.util.Date;

/**
 *
```

```
 * @author douglas
 */
public class PublicacaoComAvaliacao {
    public String message;
    public Date createdTime;
    public String Id;
    public int avaliacaoPos;
    public int avaliacaoNeg;
    public short sentimento;

    public PublicacaoComAvaliacao(String mensagem, Date createdTime, String Id, int
        this.message = mensagem;
        this.createdTime = createdTime;
        this.Id = Id;
        this.avaliacaoPos = avaliacaoPos;
        this.avaliacaoNeg = avaliacaoNeg;
        this.sentimento = sentimento;
    }

    public String getMensagem() {
        return message;
    }

    public void setMensagem(String mensagem) {
        this.message = mensagem;
    }

    public Date getCreatedTime() {
        return createdTime;
    }

    public void setCreatedTime(Date createdTime) {
        this.createdTime = createdTime;
    }

    public String getId() {
        return Id;
    }
```

```
    public void setId(String Id) {
        this.Id = Id;
    }

    public int getAvaliacaoPos() {
        return avaliacaoPos;
    }

    public void setAvaliacaoPos(int avaliacaoPos) {
        this.avaliacaoPos = avaliacaoPos;
    }

    public int getAvaliacaoNeg() {
        return avaliacaoNeg;
    }

    public void setAvaliacaoNeg(int avaliacaoNeg) {
        this.avaliacaoNeg = avaliacaoNeg;
    }

    public short getSentimento() {
        return sentimento;
    }

    public void setSentimento(short sentimento) {
        this.sentimento = sentimento;
    }
}
```

Classe "Json":

```
package addsentimentonojson;

import com.google.gson.Gson;
import com.google.gson.GsonBuilder;
/**
 *
 * @author dougl
 */
public class Json {
```

```
    public Json() {}

    public String publicacaoParaJson(PublicacaoComAvaliacao post){
        String json = new String();
        Gson gson = new GsonBuilder().create();

        json = gson.toJson(post);

        return json;
    }

    public PublicacaoLimpa jsonParaPublicacao(String json){
        PublicacaoLimpa publicacao;
        Gson gson = new Gson();

        publicacao = gson.fromJson(json, PublicacaoLimpa.class);

        return publicacao;
    }
}
```

Classe "Sentimento":

```
package addsentimentonojson;

/**
 *
 * @author dougl
 */
public class Sentimento {

    public Sentimento() {}

    public short calculaSentimento(int positivo, int negativo){
        short sentimento;

        if(positivo == Math.abs(negativo)){
            sentimento = 0;
        }else if(positivo > Math.abs(negativo)){
            sentimento = 1;
```

```
        }else{
            sentimento = -1;
        }
        return sentimento;
    }
}
```

Classe “ManipularArquivo”:

```
package addsentimentonojson;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.util.logging.Level;
import java.util.logging.Logger;
/**
 *
 * @author dougl
 */
public class ManipularArquivo {
    Sentimento s = new Sentimento();
    Json json = new Json();
    PublicacaoComAvaliacao p;
    BufferedReader lerJSON, lerTXT;
    String nomeArquivo;

    public ManipularArquivo(String nomeArquivo) {
        this.nomeArquivo = nomeArquivo;
        iniciaArquivo();
    }

    private void iniciaArquivo(){
        try {
            FileReader arquivoJSON = new FileReader(nomeArquivo + ".json");
            FileReader arquivoTXT = new FileReader(nomeArquivo + "+results.txt");
            lerJSON = new BufferedReader(arquivoJSON);
            lerTXT = new BufferedReader(arquivoTXT);
```

```
            FileWriter arquivoGravar = new FileWriter(nomeArquivo +"Senti.json");
            BufferedWriter gravar = new BufferedWriter(arquivoGravar);
            gravar.close();
            arquivoGravar.close();
        } catch (IOException ex) {
            Logger.getLogger(ManipularArquivo.class.getName())
                                                .log(Level.SEVERE, null, ex);
        }
    }

    public void ler(){
        String linhaJson, linhaTxt;

        try {
            while(lerJSON.ready() && lerTXT.ready()){
                linhaJson = lerJSON.readLine();
                linhaTxt = lerTXT.readLine();
                String linhaTxtSeparada[] = linhaTxt.split("\t");

                int positivo = Integer.parseInt(linhaTxtSeparada[1]);
                int negativo = Integer.parseInt(linhaTxtSeparada[2]);
                short sentimento;
                sentimento = s.calculaSentimento(positivo,negativo);

                PublicacaoLimpa publicacao = json.jsonParaPublicacao(linhaJson);
                p = new PublicacaoComAvaliacao(publicacao.getMensagem(),
                        publicacao.getCreatedTime(), publicacao.getId(),
                        positivo, negativo, sentimento);

                String texto = json.publicacaoParaJson(p);
                gravar(texto);
            }
        } catch (IOException ex) {
            Logger.getLogger(ManipularArquivo.class.getName())
                                                .log(Level.SEVERE, null, ex);
        }
    }

    public void gravar(String texto){
```

```
        try {
            FileWriter arquivo = new FileWriter(nomeArquivo +"Senti.json", true);
            BufferedWriter gravar = new BufferedWriter(arquivo);
            gravar.write(texto + "\n");
            gravar.close();
            arquivo.close();
        } catch (IOException ex) {
            Logger.getLogger(ManipularArquivo.class.getName())
                                                .log(Level.SEVERE, null, ex);
        }
    }
}
```