UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE INFORMÁTICA

ALLAN TREVISAN

MINERAÇÃO DE TEXTOS NO TWITTER

TRABALHO DE CONCLUSÃO DE CURSO

CURITIBA

2015

ALLAN TREVISAN

MINERAÇÃO DE TEXTOS NO TWITTER

Trabalho de Conclusão de Curso apresentado ao Departamento Acadêmico de Informática da Universidade Tecnológica Federal do Paraná como requisito parcial para obtenção do grau de Bacharel– Área de Concentração: Curso de Bacharelado em Sistemas de Informação.

Orientador: Prof. Dr. Celso A. A. Kaestner

CURITIBA

2015

LISTA DE FIGURAS

LISTA DE QUADROS

SUMÁRIO

1 INTRODUÇÃO

Os serviços de *microblogging* oferecem um novo meio de comunicação, disseminação de informação e compartilhamento de idéias (BECKER et al., 2011). Trata-se de um fenômeno relativamente recente e que pode ser definido como: *"uma forma de blogging que permite ao usuário escrever mensagens curtas (com geralmente menos de 200 caracteres) e enviá-las a amigos e observadores interessados"* (JAVA et al., 2007). Uma das principais diferenças do *microblog* é que a taxa de atualização das informações é muito maior, uma vez que o tempo entre as postagens de conteúdo é reduzido se comparado ao *blog* tradicional, que apresenta um tempo mais longo entre as postagens.

O Twitter [1] é uma plataforma de *microblog* com uma das maiores bases de usuários pelo mundo (ASUR; HUBERMAN, 2010). No segundo quadrimestre de 2012 foram registrados em torno de 30 milhões de contas ativas apenas no Brasil [2].

Há uma grande quantidade de informações circulantes no Twitter e que envolvem tópicos que vão desde a descrição de algum acontecimento, até a manifestação de opiniões sobre determinado produto, empresa ou partido político. Devido a essa variedade de assuntos que podem ser encontrados e o grande volume de dados produzidos diariamente, muitos trabalhos estão sendo desenvolvidos com o intuito de se produzir modelos que possam descobrir padrões nesses dados, e dessa forma extrair informações que venham a ser relevantes. Esses modelos são desenvolvidos com o uso de algoritmos de aprendizado de máquina e podem vir a utilizar certos conceitos de processamento de linguagem natural (PLN) para processar e representar dados textuais (MARCUS et al., 2011; ASUR; HUBERMAN, 2010; BECKER et al., 2011; GO et al., 2009; SAKAKI et al., 2010).

As Mídias Sociais exibem um comportamento que pode ser associado a uma certa forma de inteligência coletiva e que se capturado de maneira adequada pode render indicadores interessantes sobre resultados futuros (ASUR; HUBERMAN, 2010).

[1] http://twitter.com

[2] http://www.infobrasil.inf.br/es/noticia/brasil-ja-e-o-segundo-pais-em-numero-de-contas-no-twitter

Eventos recentes como a primavera árabe, os protestos de Londres de 2011, o movimento *ocupy wallstreet* e as jornadas de junho de 2013 no Brasil, são eventos que tiveram uma grande repercussão graças ao uso massivo das mídias sociais. Plataformas como Facebook[3] e Twitter exercem um efeito potencializador na difusão de idéias e no compartilhamento de opiniões sobre um evento, servindo como uma forma de organização descentralizada para tais acontecimentos (TONKIN et al., 2012).

Mensagens do Twitter podem refletir informações úteis sobre eventos de diferentes tipos e escalas de ocorrência. Um evento é uma ocorrência do mundo real e com (1) um período de tempo T_e associado e (2) um fluxo de mensagens M_e ordenadas pelo tempo, de volume substancial e publicadas durante T_e. Essas mensagens podem fornecer perspectivas únicas sobre os pontos de vista de usuários que estão interessados em participar destes eventos (BECKER et al., 2011). Em se tratando de eventos não planejados, como desastres naturais, usuários do Twitter geralmente disponibilizam informações sobre o evento antes mesmo dos veículos de notícias ou dos mecanismos de alerta tradicionais (SAKAKI et al., 2010).

No Twitter, conteúdos não relacionados a eventos incluem formas de atividades que geram volume substancial de dados durante períodos específicos de tempo e que também são comuns a conteúdos de eventos, o que acaba por contribuir com o aumento no volume de ruído nos dados (BECKER et al., 2011). Detectar eventos no Twitter é uma tarefa desafiadora devido a heterogeneidade e a grande dimensão da base de dados disponível. Muitas das mensagens do Twitter são escritas em tom informal, contendo abreviações e erros gramaticais, o que acaba dificultando ainda mais a tarefa de análise textual (BECKER et al., 2011).

Uma questão que é crítica em recuperação inteligente de informações é a de se determinar a relevância dos resultados de uma consulta. Uma maneira de se determinar relevância é por meio do cálculo de um *score* para cada mensagem (KAESTNER, 2014). Outra maneira é por meio da previsão do sentimento[4] atrelado à mensagem (ASUR; HUBERMAN, 2010). Consumidores podem utilizar análise de sentimentos para pesquisar a opinião sobre produtos antes de adquirí-los, assim como empresas podem pesquisar a opinião pública sobre seus produtos para fazer análise de mercado. A análise de sentimento no contexto do Twitter visa obter a proporção de mensagens que carregam sentimento positivo/negativo em uma coleção de mensagens sobre algum tópico de interesse (GO et al., 2009).

Além da análise de sentimento outra informação importante que pode ser obtida para se determinar relevância é a de como as mensagens sobre algum tópico de interesse

[3] http://facebook.com

[4] É definido aqui sentimento como uma emoção pessoal, positiva ou negativa, indicativa de polaridade.

estão agrupadas de acordo com alguma característica de similaridade, além de quais são os agrupamentos em si. Por exemplo, dentre as mensagens negativas sobre um determinado produto, há um grupo de pessoas que reclama da característica 1 do produto e um outro grupo que reclama da característica 2, para a empresa que fabrica esse produto seria interessante saber da existência desses dois grupos.

Dou et al. (2012) faz a sumarização de algumas tarefas que merecem destaque quando se está trabalhando com análise de eventos em mídias sociais. As tarefas são as seguintes:

- **Identificação de novos eventos:** refere-se a identificação do primeiro relato em um tópico de interesse ao se monitorar constantemente um fluxo de notícias. É um processo automático que faz uma identificação binária. Geralmente consiste de outras duas sub-tarefas: descobrir eventos previamente não identificados em uma coleção de dados acumulados e identificar novos eventos a partir de fluxos de texto gerados em tempo real, processando os documentos um a um e computando um *score*, se este *score* exceder um determinado limiar o documento é marcado como um novo evento.

- **Acompanhamento de eventos:** refere-se ao estudo de como os eventos se desenrolam, se trata de como acompanhar um evento no decorrer do tempo e extrair informações importantes para auxiliar na consciência situacional, como por exemplo durante situações de crise, com destaque para o estudo da difusão de informação sobre um evento especifico em uma rede social.

- **Sumarização de eventos:** refere-se ao que já aconteceu, ou seja, fazer um resumo das informações acerca de um evento passado baseando-se em características identificadas em uma coleção de documentos, como por exemplo identificar URLs mais compartilhadas, *hashtags* mais usadas ou análise de sentimentos. Um exemplo é o caso do sistema Tweetinfo apresentado em (MARCUS et al., 2011).

1.1 OBJETIVOS

1.1.1 OBJETIVO GERAL

Este trabalho tem por objetivo realizar a recuperação inteligente de informações sobre dados disponíveis no Twitter, de modo a produzir informações de sumarização acerca de algum evento do mundo real.

1.1.2 OBJETIVOS ESPECÍFICOS

- Utilizar aprendizado de máquina supervisionado para realizar análise de sentimentos sobre dados do Twitter.
- Utilizar aprendizado de máquina não supervisionado para gerar agrupamentos de dados do Twitter.
- Utilizar conceitos de PLN para manipular e representar grandes coleções de dados textuais que serão utilizados pelos algoritmos de aprendizado de máquina.
- Realizar análise de erro para determinar a eficiência dos algoritmos de aprendizado de máquina.
- Implementar uma ferramenta de exploração e análise de dados do Twitter que utilize aprendizado de máquina e processamento de linguagem natural para realizar a sumarização de informações acerca de algum evento do mundo real.

1.1.3 JUSTIFICATIVA

Ao se buscar informações sobre determinado evento no Twitter é possível notar uma certa insatisfação por parte do usuário na utilização do mecanismo tradicional de buscas (MARCUS et al., 2011). Buscas por palavras-chave são problemáticas, uma vez que os resultados de uma consulta nem sempre são aquilo que o usuário esperava. A busca por correspondências ao nível de índice dos termos de uma consulta é muitas vezes imprecisa, pois ela pode não retornar documentos relevantes que incluem sinônimos, ou pode ainda retornar documentos irrelevantes que incluem termos ambíguos, e a falta de treinamento em construção de consultas por parte dos usuários acaba por contribuir com este cenário (KAESTNER, 2014). O resultado de uma busca também depende da perspectiva de quem o está interpretando, uma análise mais profunda pode possibilitar a recuperação de informações baseadas na sua relevância ao invés da presença de palavras-chave (BAEZA-YATES; RIBEIRO-NETO, 2011).

1.2 CENÁRIO MOTIVADOR

É maio de 2015 e João vem acompanhando pela televisão e pelas mídias sociais informações sobre a onda de protestos que estão ocorrendo por todo o país. Muitos usuários do Twitter utilizam a plataforma para compartilhar informações e opiniões acerca do evento que está acontecendo. Os veículos tradicionais de notícias não apresentam todas as informações

sobre o evento e apenas uma pequena parcela dos compartilhamentos são baseadas em relatos por testemunhas oculares que o presenciaram.

João é um jornalista e quer ter uma visão geral sobre o que as pessoas estão comentando sobre o evento. Ele faz então uma pesquisa no Twitter pelos termos que ele acredita serem relevantes. O resultado da busca retorna milhares de mensagens individuais que não estão agrupadas nem organizadas de nenhuma maneira significativa.

Para obter um maior grau de significância das informações João então decide usar uma ferramenta para exploração e análise de dados do Twitter. Primeiramente, João usa a ferramenta para fazer uma busca por "manifestação de professores em São Paulo". A ferramenta passa então a coletar as mensagens que correspondam à consulta e então as armazena em uma base de dados para posterior análise.

No momento da análise João tem uma visão geral sobre os termos mais importantes presentes nas mensagens pela exibição de uma nuvem de palavras. João também tem uma visão de como as mensagens estão agrupadas, ele pode selecionar o agrupamento de interesse e exibir uma nuvem de palavras para esse agrupamento, tendo uma noção dos termos mais importantes e da distribuição de diferentes assuntos em cada agrupamento.

Se João quiser ter uma visão geral sobre a polarização das opiniões das pessoas sobre o que está acontecendo ele pode visualizar qual o agregado geral de sentimento positivo/negativo nas mensagens. João também pode visualizar o agregado de sentimento em cada agrupamento separadamente, tendo assim uma visão da distribuição da polarização de opiniões por diferentes assuntos.

1.3 METODOLOGIA

Para uma melhor organização, o trabalho foi subdividido em duas fases:

1. Aquisição de embasamento teórico sobre aprendizado de máquina e PLN, e levantamento do estado da arte com relação a recuperação inteligente de informações em mídias sociais.
2. Implementação de uma ferramenta que utilize esses conceitos para realizar a sumarização de informações acerca de um evento do mundo real.

Inicialmente, na fase 1 é feito um estudo sobre as tarefas de classificação e *clusterização* em aprendizado de máquina. Nesta fase é também feita a escolha dos algoritmos utilizados para realizar essas tarefas. A tarefa de classificação é utilizada para se construir um

classificador de sentimento, cujo o objetivo é classificar a emoção inerente a *tweets* em positivo ou negativo. A tarefa de *clusterização* tem por objetivo agrupar *tweets* sobre algum evento alvo em *clusters*, cada *cluster* obtido por esse processo irá conter *tweets* que são semelhantes entre si, refletindo possivelmente a distribuição de diferentes assuntos (ou sub-tópicos) pelos *clusters* obtidos. Essa fase também tem por objetivo estudar diferentes formas de se analisar a qualidade dos modelos gerados nas tarefas de classificação e *clusterização*. Os conceitos de PLN são utilizados para pré-processar e representar os *tweets* em vetores numéricos de *features*, que são utilizados pelos algoritmos de aprendizado. São também analisados trabalhos semelhantes que utilizam aprendizado de máquina e outros conceitos relacionados a recuperação inteligente de informações para analisar mídias sociais.

Posteriormente, na fase 2 é realizado o desenvolvimento da ferramenta. Nessa fase é feita a análise, projeto e implementação. São levantados requisitos funcionais e não funcionais e apresentado o diagrama de casos de uso e diagrama de classes da ferramenta. Posteriormente, é também realizada a validação da ferramenta, com sua utilização em um contexto real onde são coletados *tweets* sobre algum evento do mundo real e esses *tweets* são analisados pela ferramenta, que produzirá informações de sumarização sobre o evento.

1.4 ORGANIZAÇÃO DO TRABALHO

No capítulo 2 é apresentado um levantamento bibliográfico sobre o estado da arte envolvendo aprendizado de máquina e processamento de linguagem natural. É apresentada a ideia geral por trás do aprendizado supervisionado e não supervisionado, no qual são explicados os algoritmos que serão utilizados na ferramenta, respectivamente: *Suport Vector Machines, Naïve Bayes* e K-Means. Também é falado sobre o processo de análise de erro que é necessário para que os modelos gerados sejam suficientemente corretos. Na seção sobre processamento de linguagem natural é explicado como características textuais podem ser extraídas e representadas para serem utilizadas pelos algoritmos de aprendizado.

No capítulo 3 são descritas as principais funcionalidades da ferramenta bem como são levantados alguns pontos importantes com relação ao processo de desenvolvimento da mesma.

No capítulo 4 é finalizado o trabalho com algumas conclusões e apontamentos de possíveis melhorias e trabalhos futuros.

2 LEVANTAMENTO BIBLIOGRÁFICO E ESTADO DA ARTE

2.1 APRENDIZADO DE MÁQUINA

O aprendizado de máquina é uma área que vem ganhando uma importância crescente com o rápido crescimento dos conjuntos de dados disponíveis às ciências experimentais (HAN et al., 2011). Problemas abordados vão desde construir uma função de previsão ligando diferentes observações de dados para determinado fenômeno, até a classificação de observações ou o aprendizado de estrutura em um conjunto de dados não previamente rotulado.

Segundo Samuel (2000) o aprendizado de máquina é o campo de estudo que permite que computadores aprendam sem que sejam explicitamente programados. Um exemplo de aplicação que não pode ser programada manualmente é o reconhecimento de escrita [1], que necessita que inicialmente sejam apresentados padrões de diferentes tipos de escrita para que dessa forma o programa possa realizar a tarefa de reconhecimento. Samuel (2000) escreveu um programa que era capaz de jogar damas; esse programa jogava 10000 jogos contra si mesmo e dessa forma conseguia determinar (aprendizado) quais posições do tabuleiro eram boas ou ruins baseado nas vitórias/derrotas.

Mitchell (1997) define um problema de aprendizado da seguinte forma: *"Um programa de computador é dito aprender a partir de uma experiência E com relação a alguma tarefa T e alguma medida de performance P, se a performance em T, medida por P, melhorar com a experiência E*".

Um exemplo de aprendizado pode ser dado por: suponha que um programa cliente de e-mails observe quais e-mails são marcados ou não como *spam*, e baseado nisso aprenda como filtrar melhor *spam*. Nesse caso, a tarefa T é classificar e-mails como sendo *spam* ou não *spam*. A experiência E é observar o usuário rotular e-mails como sendo *spam* ou não *spam*. E a medida de performance P é o número (ou fração) de e-mails classificados corretamente como *spam* ou não *spam* (NG, 2014).

[1] http://en.wikipedia.org/wiki/Optical_character_recognition

O objetivo principal de um sistema de aprendizado é generalizar a partir da experiência. Generalização nesse contexto é a habilidade de uma máquina em lidar com precisão com novos exemplos/tarefas ainda não observados, após ter sido submetida a um conjunto de dados de aprendizado (BISHOP et al., 2006). Os exemplos de treinamento vêm de alguma distribuição de probabilidades geralmente desconhecida (considerada representativa do espaço de ocorrências), e o sistema de aprendizado constrói um modelo geral sobre este espaço, que permite a ele produzir previsões suficientemente corretas em novos casos.

Em aprendizado de máquina os algoritmos podem ser organizados dentro de uma taxonomia baseada no tipo de resultado desejado, ou no tipo de entrada disponível durante a fase de treinamento, dentre os quais podem ser citados:

- **Aprendizado Supervisionado:** os algoritmos são treinados com exemplos rotulados, neste caso há a figura do supervisor (ou professor) que apresenta as "respostas certas" para o algoritmo, ou seja, existe uma função que mapeia uma entrada em uma saída conhecida. Podemos citar como exemplo a tarefa de classificar um e-mail como *spam* ou não *spam* a partir de um conjunto de e-mails previamente classificados como *spam* ou não *spam* (NG, 2014).

- **Aprendizado Não Supervisionado:** neste caso os algoritmos atuam sobre exemplos não rotulados, ou seja, a partir de uma entrada onde a saída desejada é desconhecida. Aqui o objetivo é descobrir estrutura nos dados e não apenas generalizar um mapeamento de entradas em saídas. Podemos citar como exemplo o problema do *cocktail party* [2], em que o objetivo é separar sons de pessoas falando de uma música de fundo (NG, 2014).

- **Aprendizado por Reforço:** através de uma sequência de ações um agente visa obter conhecimento sobre como o ambiente responde às suas ações, e desta forma tenta sintetizar um conjunto de ações que maximizem uma recompensa cumulativa, cada ação do agente pode ser avaliada como positiva ou negativa, podendo receber recompensa ou punição respectivamente. Um exemplo é o programa que jogava damas de Samuel (2000).

2.1.1 APRENDIZADO DE MÁQUINA SUPERVISIONADO

Nesta seção é apresentada a ideia geral sobre as tarefas de regressão e classificação em aprendizado supervisionado.

[2] http://en.wikipedia.org/wiki/Cocktail_party_effect

2.1.1.1 A TAREFA DE REGRESSÃO

Em uma tarefa de regressão, o objetivo é prever uma saída de valor contínuo para uma determinada entrada (NG, 2014).

Como pode ser observado, os pontos apresentados no gráfico da Figura 1 seguem uma distribuição em que uma função linear poderia ser ajustada aos dados. Nesse sentido, para realizar uma previsão do valor em y de um novo ponto ainda não observado e que siga o mesmo padrão dos pontos presentes poderia ser feita com o uso de um modelo conhecido como Regressão Linear com uma variável (NG, 2014).

Os dados coletados sobre o problema são chamados de exemplos de treinamento. Esses dados são então apresentados para o algoritmo de Regressão Linear que tem como objetivo encontrar uma função (linear), conhecida como função hipótese, definida pela equação (1), que neste caso representa a reta que passa o mais próximo possível dos pontos do gráfico, de modo a minimizar a soma das distâncias entre os pontos e a reta.

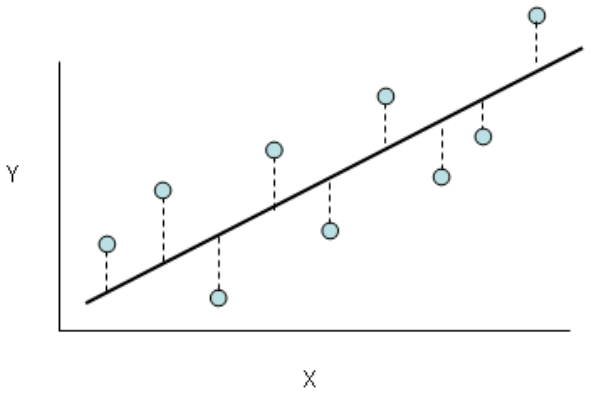

Figura 1: Regressão Linear com Uma Variável.
Fonte: Adaptado de NG (2014)

$$h_\theta(x) = \theta_0 + \theta_1 x \tag{1}$$

Como exemplo ilustrativo, os dados do gráfico poderiam ser interpretados como a relação entre o tamanho em metros quadrados e o preço de venda de casas em uma região específica. Nesse caso, o objetivo é descobrir um modelo geral capaz de prever o preço de venda de uma casa baseado no seu tamanho. O tamanho é a característica (ou *feature*) utilizada para realizar as previsões.

Os algoritmos aqui descritos são treinados com m exemplos de treinamento: $(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})$, cada exemplo é composto de um par, em que o primeiro componente é um exemplo codificado na forma de um vetor numérico (variável de entrada) e o segundo componente é o valor que se está interessado em prever (variável de saída). Cada exemplo de treinamento é composto por *n features*.

Para encontrar a função hipótese é necessário resolver um problema de minimização, em que o objetivo de otimização é minimizar a função erro $J(\theta_0, \theta_1)$ na equação (2), também chamada de erro quadrático médio (EQM), de modo a encontrar os melhores valores possíveis para os parâmetros θ_o, θ_1 que serão utilizados pela função hipótese. A função hipótese $h_\theta(x)$ realiza previsões dos valores y para valores x ainda não observados.

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 \tag{2}$$

O caso definido acima é o mais simples, pois lida apenas com uma *feature*. Problemas mais complexos lidam com uma quantidade maior de *features*, em que $n > 1$. Será necessário então estender a notação utilizada. É utilizada a notação da álgebra linear, em que as variáveis são representadas na forma de matrizes e vetores.

Para a representação dos exemplos de treinamento é utilizada uma matriz X de dimensão $m \times n + 1$, dada pela equação (3).

$$X = \begin{bmatrix} x^{(1)} \\ x^{(2)} \\ \vdots \\ x^{(m)} \end{bmatrix} \in R^{m \times n+1} \tag{3}$$

Para cada exemplo de treinamento há um vetor de *features* $x^{(i)} \in X$ de dimensão $n+1$, em que o elemento $x_j^{(i)}$ representa a *feature* j do exemplo de treinamento i, dado pela equação (4).

$$x^{(i)} = \begin{bmatrix} x_0^{(i)} \\ x_1^{(i)} \\ \vdots \\ x_n^{(i)} \end{bmatrix} \in R^{n+1} \tag{4}$$

Os valores alvo dos exemplos de treinamento são representados na forma de um vetor m-dimensional, em que cada $y^{(i)} \in y$ representa o valor alvo para o exemplo de treinamento $x^{(i)}$, dado pela equação (5).

$$y = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix} \in R^m \tag{5}$$

Os parâmetros do modelo são representados na forma de um vetor de pesos (coeficientes) $n+1$ dimensional, em que cada $\theta_j \in \theta$ é o peso dado à *feature* x_j e que foi aprendido pelo algoritmo de modo a minimizar o erro do modelo, dado pela equação (6).

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix} \in R^{n+1} \tag{6}$$

As funções hipótese e erro podem então ser redefinidas pelas equações (7) e (8).

$$h_\theta(x) = \sum_{i=0}^{n} \theta_i x_i = \theta^T x \tag{7}$$

$$J(\theta_0, \theta_1, ..., \theta_n) = J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 \tag{8}$$

2.1.1.2 A TAREFA DE CLASSIFICAÇÃO

Classificação é a tarefa de se escolher um rótulo de classe correto para uma determinada entrada (MITCHELL, 1997). Em tarefas de classificação básicas, cada entrada é considerada isoladamente de todas as outras entradas, e o conjunto de rótulos é definido *a priori* (BIRD et al., 2009). Um classificador é considerado supervisionado se ele for construído baseado em dados de treinamento contendo o rótulo correto para cada entrada, essa ideia é ilustrada pelo diagrama da Figura 2.

Durante o treinamento um extrator de *features* é utilizado para converter cada valor de entrada em um vetor de *features*. Esses vetores capturam as informações básicas sobre cada

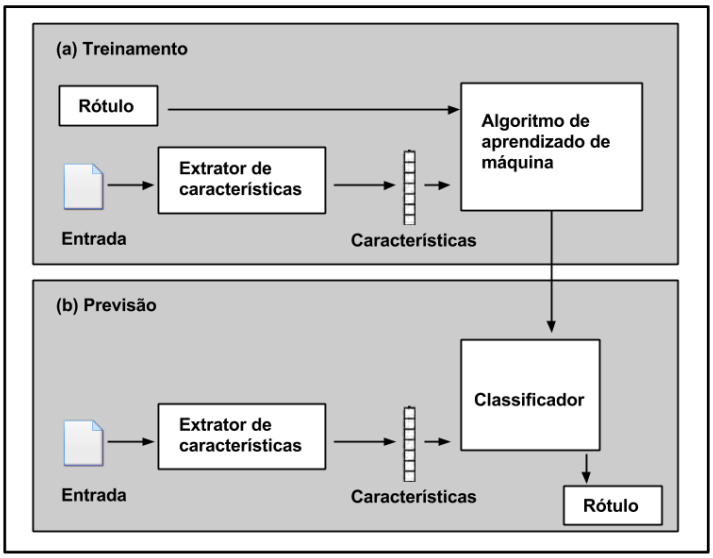

Figura 2: *Framework* de Classificação Utilizado em Aprendizado Supervisionado.
Fonte: Adaptado de Bird et al. (2009)

entrada e devem ser usados para classificá-las. Pares de vetores e rótulos são alimentados nos algoritmos de aprendizado para gerar um modelo de classificação. Durante a fase de previsão, o mesmo extrator é utilizado para converter entradas ainda não observadas em vetores de *features*. Estes vetores são então utilizados pelo modelo, que gera previsões dos rótulos (BIRD et al., 2009).

Em um problema de classificação, o objetivo é encontrar uma função hipótese, conhecida como fronteira de decisão (NG, 2014). No caso da Figura 3 essa função deve separar os dados de maneira ótima. A fronteira de decisão pode ser obtida com o uso de um modelo de Regressão Logística (NG, 2014).

A nova função hipótese deve produzir valores no intervalo $0 \ldots 1$. Dessa forma cada elemento $y^{(i)} \in y$ irá assumir valores $y^{(i)} \in 0, 1, 2, 3, \ldots$ que representam as classes do problema. Neste exemplo ilustrativo há apenas duas classes, 0 "bola vermelha" e 1 "bola azul". Para prever (classificar) se determinado exemplo corresponde a uma bola vermelha, a função hipótese $h_\theta(x)$ deve produzir um valor que seja maior ou igual a um determinado limiar, caso contrário o exemplo será classificado como uma bola azul (NG, 2014).

A função $h_\theta(x^{(i)})$ tem o objetivo de estimar qual a probabilidade do exemplo $x^{(i)}$ pertencer a classe positiva do problema, ou seja $h_\theta(x^{(i)}) = P(y = 1|x^{(i)}; \theta)$, a probabilidade de $y = 1$, dado $x^{(i)}$, parametrizado por θ, em que $P(y = 1|x^{(i)}; \theta) + P(y = 0|x^{(i)}; \theta) = 1$.

Para que a função hipótese produza valores $0 <= h_\theta(x) <= 1$ é utilizada uma função

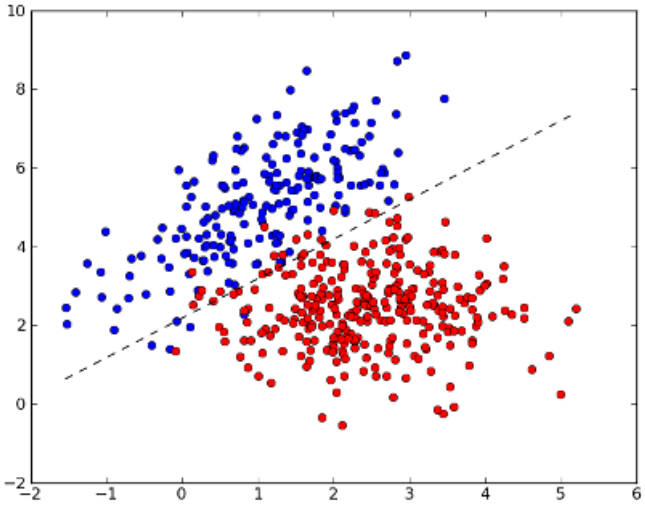

Figura 3: Fronteira de Decisão para Regressão Logística com Duas Variáveis.
Fonte: Adaptado de NG (2014)

sigmóide $g(z)$ para limitar os valores da hipótese, dada pela equação (9). A função $h_\theta(x)$ pode então ser redefinida como mostra a equação (10). A função erro $J(\theta)$ também é redefinida pela equação (11).

$$g(z) = \frac{1}{1+e^{-z}} \tag{9}$$

$$h_\theta(x) = g(\theta^T x) \tag{10}$$

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} log(h_\theta(x^{(i)})) + (1-y^{(i)}) log(1-h_\theta(x^{(i)}))\right] \tag{11}$$

2.1.1.3 O ALGORITMO *NAÏVE-BAYES*

O *Naïve-Bayes* é um algoritmo de classificação baseado no teorema de Bayes (KIM et al., 2006). Este algoritmo pertence a classe de modelos generativos e possui várias representações diferentes. A representação apresentada aqui é a de Bernoulli, que utiliza *features* binárias. Tem-se então um modelo $p(x|y)$. O *Naïve-Bayes* possui a prerrogativa de que os valores x_i são condicionalmente independentes dado y. Situação ilustrada pela equação (12).

Mesmo a prerrogativa do Naïve-Bayes sendo muito forte, ele ainda se sai bem em muitos problemas de classificação (HAN et al., 2011).

$$p(x_1,...,x_n|y) = \prod_{i=1}^{n} p(x_i|y) \tag{12}$$

Com o modelo parametrizado por $\phi_{i|y=1} = p(x^{(i)} = 1|y = 1), \phi_{i|y=0} = p(x^{(i)} = 1|y = 0)$, e $\phi_y = p(y = 1)$. Dado um conjunto de treinamento $\{(x^{(1)}, y^{(1)}), ..., (x^{(m)}, y^{(m)})\}$, a probabilidade conjunta dos dados é dada pela equação (13).

$$\mathscr{L}(\phi_y, \phi_{j|y=0}, \phi_{j|y=1}) = \prod_{i=1}^{m} p(x^{(i)}|y^{(i)}) \tag{13}$$

Tendo maximizado esta probabilidade com relação aos parâmetros, tem-se as seguintes estimativas definidas nas equações (14), (15) e (16).

$$\phi_{j|y=1} = \frac{\sum_{i=1}^{m} 1\{x_j^{(i)} = 1 \wedge y^{(i)} = 1\}}{\sum_{i=1}^{m} 1\{y^{(i)} = 1\}} \tag{14}$$

$$\phi_{j|y=0} = \frac{\sum_{i=1}^{m} 1\{x_j^{(i)} = 1 \wedge y^{(i)} = 0\}}{\sum_{i=1}^{m} 1\{y^{(i)} = 0\}} \tag{15}$$

$$\phi_y = \frac{\sum_{i=1}^{m} 1\{y^{(i)} = 1\}}{m} \tag{16}$$

Tendo todos estes parâmetros, uma previsão pode ser feita dado um vetor de *features* x pela equação (17). A classe estimada do exemplo é aquela que obtiver a maior probabilidade *a posteriori*.

$$p(y = 1|x) = \frac{p(x|y = 1)p(y = 1)}{p(x)} \tag{17}$$

$$= \frac{(\prod_{i=1}^{n} p(x_i|y = 1))p(y = 1)}{(\prod_{i=1}^{n} p(x_i|y = 1))p(y = 1) + (\prod_{i=1}^{n} p(x_i|y = 0))p(y = 0)}$$

2.1.1.4 O ALGORITMO *SUPPORT VECTOR MACHINES*

O SVM é um algoritmo baseado no conceito de hiperplanos que definem uma fronteira de decisão (HASTIE et al., 2009). O objetivo é encontrar um hiperplano que resulte na menor margem (uma margem rígida) entre os exemplos das duas classes do problema, como ilustra a Figura 4. Os vetores mais próximos às margens são chamados de vetores de suporte.

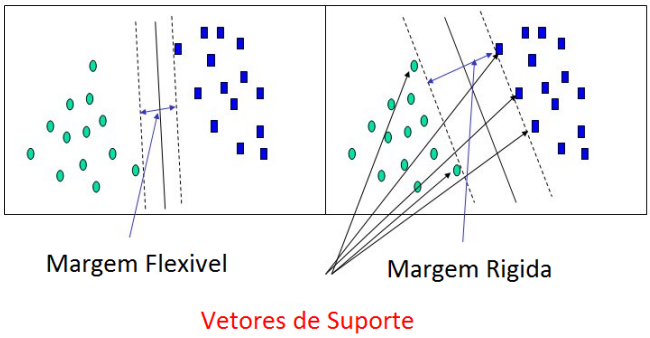

Figura 4: Margem SVM.
Fonte: Adaptado de Han et al. (2011)

No caso não linear o SVM utiliza funções de *kernel* para mapear os exemplos de entrada para um espaço de características (HAN et al., 2011). Esta situação é ilustrada pela Figura 5. Neste novo espaço, os exemplos são linearmente separáveis, portanto ao invés de construir uma função complexa, no espaço de características, o algoritmo precisa encontrar apenas o hiperplano que melhor separe os dados.

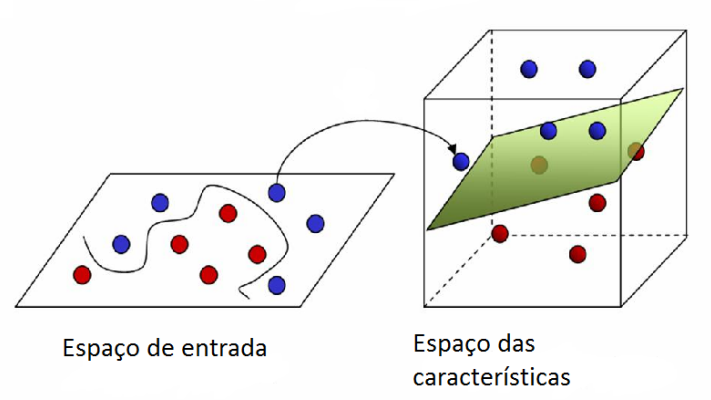

Figura 5: Mapeamento Utilizando *Kernels*.
Fonte: Adaptado de Han et al. (2011)

Uma das possíveis representações do hiperplano ótimo, conhecida como hiperplano

canônico é dada por $|\theta_0 + \theta^T x| = 1$ (HAN et al., 2011). A distância entre um exemplo x e um hiperplano (θ_0, θ) é dada pela equação (18).

$$distância = \frac{|\theta_0 + \theta^T x|}{||\theta||} \tag{18}$$

A distância para os vetores de suporte é dada pela seguinte igualdade na equação (19).

$$\frac{1}{||\theta||} = \frac{|\theta_0 + \theta^T x|}{||\theta||} \tag{19}$$

Como a margem M é duas vezes a distância para os exemplos mais próximos, tem-se a situação ilustrada pela equação (20).

$$M = \frac{2}{||\theta||} \tag{20}$$

O problema de maximizar M é equivalente ao problema de minimizar a função $J(\theta)$ na equação (21), sujeita a algumas restrições. As restrições modelam o fato de o hiperplano classificar corretamente todos os exemplos de treinamento $x^{(i)}$.

$$J(\theta) = \frac{1}{2}||\theta||^2 + C\sum_{i=1}^{m} \varepsilon_i \tag{21}$$

$$sujeito\ a\ y^{(i)}(\theta^T x^{(i)}) \geq 1 - \varepsilon_i, i = 1, ..., m$$
$$\varepsilon_i \geq 0, i = 1, ..., m$$

Os exemplos podem ter margens menores do que 1. Se um exemplo tiver margem (funcional) $1 - \varepsilon_i$ (com $\varepsilon > 0$), a função objetivo sofre uma punição, sendo acrescida por $C\varepsilon_i$. Para que o algoritmo seja menos suscetível a *outliers* é utilizada regularização. O parâmetro C controla a ponderação relativa entre os objetivos de fazer $||\theta||^2$ pequeno (o que faz a margem aumentar) e assegurar que a maioria dos exemplos tenha margem funcional de pelo menos 1.

As funções de *kernel* medem o quão similares (distantes) dois exemplos de treinamento são entre si. A equação (22) ilustra uma função de *kernel* Gaussiana. Essa medida de similaridade é aplicada a todos os exemplos de treinamento. Para cada exemplo de treinamento é calculada a distância do exemplo de treinamento para todos os outros exemplos, resultando em um mapeamento de um espaço n dimensional (o número de *features*) para um espaço

possívelmente m dimensional (o número de exemplos de treinamento). Outras funções de *kernel* que podem ser utilizadas são a linear, polinomial e sigmóide (HAN et al., 2011).

$$K(x,z) = \exp\left(-\frac{||x-z||^2}{2\sigma^2}\right) \tag{22}$$

2.1.1.5 *UNDERFITTING* VERSUS *OVERFITTING*

Um problema que merece atenção quando se está projetando modelos preditivos é o caso em que os modelos se adequam bem demais aos dados de treinamento e mesmo assim acabam falhando em generalizar bem para novos exemplos (HAN et al., 2011). Se forem utilizadas muitas *features*, a hipótese aprendida pode ficar muito complexa (em que $J(\theta) \approx 0$). Esse problema é conhecido como *"overfitting"* ou sobreajuste (NG, 2014). Se forem utilizadas poucas *features* ou *features* de baixa qualidade os modelos falharão em se adequar até mesmo aos dados de treinamento, nesse caso o problema é o inverso, problema conhecido como *"underfitting"* ou subajuste (NG, 2014). Na Figura 6 são ilustradas três situações: *underfitting*, ajuste ideal e *overfitting*.

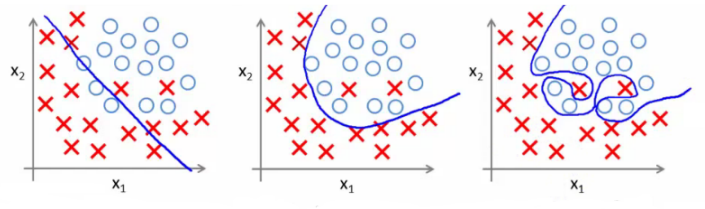

Figura 6: *Underfitting*, Ajuste Ideal e *Overfitting*.
Fonte: Adaptado de NG (2014)

Para corrigir o problema do *underfitting* e *overfitting* e obter um ajuste ideal para o modelo, pode-se utilizar dois artifícios: redução de dimensionalidade e regularização (HAN et al., 2011).

- **Redução de dimensionalidade:** nesse caso é feita a seleção de um subconjunto das *features* originais, de modo a diminuir a dimensão dos vetores de *features*. Esse processo reduz o número de variáveis aleatórias ou atributos considerados (HAN et al., 2011). Essa solução pode ser ruim, pois podem se perder informações com bom potencial preditivo (NG, 2014). Há métodos automáticos que fazem a escolha das *features* a serem mantidas, de modo a prejudicar o modelo o mínimo possível, como por exemplo o PCA (*Principal*

Component Analysis). O PCA transforma ou projeta os dados originais para um espaço menor (HAN et al., 2011). Utilizar o PCA para resolver problemas de *underfitting* e *overfitting* não é considerado uma boa prática (NG, 2014). Geralmente o PCA é utilizado para fazer compressão de dados, de modo a aumentar a velocidade dos algoritmos de aprendizado de máquina (NG, 2014). Outro método utilizado para realizar redução de dimensionalidade muito usado em recuperação de informações e que é muito similar ao PCA é o LSA [3] (*Latent Semantic Analysis*). O LSA realiza redução de dimensionalidade linear, ele opera sobre os vetores de *features* diretamente ao invés de operar em uma matriz de covariância como o PCA.

- **Regularização:** neste caso todas as *features* são mantidas, mas reduz-se a magnitude dos parâmetros θ adicionando-se um termo de regularização ao erro, forçando dessa forma a penalização dos parâmetros θ de modo que o erro continue baixo (NG, 2014). Essa abordagem funciona bem para situações em que o número de *features* é muito grande. Assim, cada uma das *features* contribui um pouco com a tarefa de prever y. Para o caso da Regressão Linear e Regressão Logística, esse efeito é obtido ao adicionar o termo de regularização r da equação (23) ao erro. O parâmetro de regularização λ controla o quanto os parâmetros θ serão penalizados. Quanto maior o λ mais os parâmetros θ serão penalizados. O parâmetro de regularização λ não deve ser nem muito pequeno, nem muito grande para evitar o *overfitting* e o *underfitting*, respectivamente. O parâmetro C do SVM pode ser interpretado como $\frac{1}{\lambda}$ (NG, 2014).

$$r = \frac{\lambda}{2m} \sum_{j=1}^{n} \theta_j^2 \tag{23}$$

2.1.1.6 DETECTANDO *UNDERFITTING* E *OVERFITTING*

Considera-se uma boa prática dividir os dados de treinamento em três subconjuntos: treino, validação e teste. Uma boa proporção geralmente consiste em separar 60% dos dados para treinamento, 20% para validação e 20% para teste. Dessa forma é possível realizar seleção de modelos e simular como os modelos se comportam na presença de dados ainda não observados (NG, 2014).

A medida do erro nos dados de treino *versus* a medida do erro nos dados de validação ou teste pode ser utilizada para detectar se o algoritmo está sofrendo de *underfitting* ou *overfitting* (NG, 2014). Quando o erro nos dados de treino for alto e o erro nos dados de

[3] http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.TruncatedSVD.html

validação for próximo ao erro nos dados de treino, significa que a hipótese aprendida é muito simples e o modelo sofre de *underfitting*. Quando o erro nos dados de treino for baixo e o erro nos dados de validação for muito maior que o erro nos dados de treino, significa que o algoritmo pode estar sofrendo de *overfitting*, ou seja a hipótese aprendida se ajusta muito bem aos dados de treino, mas falha em generalizar bem para dados ainda não observados.

2.1.1.7 ANÁLISE DE ERRO

Há vária maneiras de se analisar a capacidade preditiva de um modelo em termos de seus erros e acertos, entre elas: acurácia e medida F (HAN et al., 2011). A acurácia mede apenas qual a percentagem de exemplos classificados corretamente de acordo com seus rótulos. A medida F mede a capacidade preditiva do modelo em termos de precisão e cobertura (ou *recall*).

A precisão e o *recall* são calculados em termos dos falsos positivos, falsos negativos, verdadeiros positivos e verdadeiros negativos gerados pelo modelo. Essa situação pode ser descrita pela matriz de confusão no Quadro 1. Essa matriz representa a quantidade de exemplos do conjunto de validação que foram classificados corretamente como positivos ou negativos e erroneamente como positivos ou negativos (HAN et al., 2011).

Quadro 1: Matriz de Confusão.

		Valor Real		
		Positivo	Negativo	Total
Valor Estimado	Positivo	VP	FP	$VP+FP$
	Negativo	FN	VN	$FN+VN$
	Total	$VP+FN$	$FP+VN$	N

Fonte: Adaptado de (HAN et al., 2011)

Segundo Han et al. (2011) a precisão refere-se a fração de exemplos classificados como positivos que são realmente positivos. Ela mede a exatidão do classificador, uma alta precisão significa menos falsos positivos, enquanto que uma precisão baixa significa mais falsos positivos, equação (24).

$$Precisão = \frac{VP}{VP+FP} \tag{24}$$

Segundo Han et al. (2011) o *recall* se refere a fração de exemplos que são realmente positivos e que foram classificados corretamente como tal. Ele mede a completude ou sensibilidade do classificador, um alto *recall* significa menos falsos negativos, enquanto que

um baixo *recall* significa mais falsos negativos, equação (25).

$$Recall = \frac{VP}{VP + FN} \tag{25}$$

A precisão e o *recall* do modelo podem ser controlados por um limiar na função hipótese (NG, 2014). Há um *tradeoff* nesse caso. Se o limiar for muito alto, quer dizer então que algo é classificado como positivo somente se houver muita confiança, dessa forma tem-se uma alta precisão e um baixo *recall*. Em contrapartida, se o objetivo for evitar muitos falsos negativos, pode-se diminuir o limiar da função hipótese para obter-se um alto *recall* e uma baixa precisão (NG, 2014).

Considera-se quanto maior a precisão e o *recall* do modelo, melhor será sua capacidade preditiva. Uma métrica de comparação entre precisão e *recall* é o *score* F, equação (26). Essa medida representa a média harmônica entre precisão e *recall* (HAN et al., 2011).

$$F = 2\frac{Precisão.Recall}{Precisão + Recall} \tag{26}$$

2.1.2 APRENDIZADO DE MÁQUINA NÃO SUPERVISIONADO

A tarefa de agrupamento contrasta com a tarefa de classificação em aprendizado supervisionado, no qual o objetivo é: adequar uma hipótese a um conjunto de dados rotulados. Em aprendizado não supervisionado o objetivo é: determinar uma estrutura nos dados a partir de um conjunto de dados não rotulados, isto é, determinar como os dados se agrupam em agrupamentos (*clusters*) semelhantes e coesos (NG, 2014). Os algoritmos de agrupamento agrupam os dados baseando-se nas características de similaridade entre os dados (HAN et al., 2011).

O algoritmo de agrupamento denominado K-Means é descrito a seguir.

2.1.2.1 O ALGORITMO *K-MEANS*

O algoritmo de *clusterização* K-Means pode ser definido pelo seguinte pseudo código (HAN et al., 2011).

Algoritmo 1: K-Means

Entradas:

K (número de *clusters*);

Exemplos de treinamento $\{x^{(1)}, x^{(2)}, ..., x^{(m)}\}$;

Inicializar aleatoriamente K centróides $\{\mu_1, \mu_2, ..., \mu_k\}$;

Repetir {

para $i = 1$ até m

$c^{(i)} := indice(1...K)$ do centróide mais próximo de $x^{(i)}$;

para $k = 1$ até K

$\mu_k :=$ média dos pontos atribuídos ao *cluster* k;

}

Inicialmente, o algoritmo elege K pontos aleatoriamente como sendo os centróides dos *clusters*. O algoritmo então percorre cada exemplo de treinamento e, dependendo do centróide que estiver mais próximo, atribui ao exemplo o índice (1 ... K) do respectivo centróide. Posteriormente, o algoritmo toma cada centróide e o desloca para a média dos pontos atribuídos a ele. O algoritmo computa o erro para a configuração de centróides obtida, dado pela equação (27), também chamada de distorção, que representa a média das distâncias quadráticas entre os exemplos de treinamento e os centróides dos respectivos *clusters* (NG, 2014). O algoritmo repete esse processo até a convergência, quando não há alteração significativa nos centróides.

$$J(c^{(1)}, ..., c^{(m)}, \mu^{(1)}, ..., \mu^{(k)}) = \frac{1}{m} \sum_{i=1}^{m} \left| x^{(i)} - \mu_{c^{(i)}} \right|^2 \tag{27}$$

Como o algoritmo inicia elegendo K centróides aleatoriamente ele estará sujeito a ótimos locais. Se os ótimos locais forem um problema para a aplicação, múltiplas inicializações aleatórias podem ser feitas para se obter os *clusters* que resultem no menor erro. Se o mesmo resultado for observado muitas vezes para um determinado número de inicializações, esse pode ser o indicativo de um ótimo global.

Outro detalhe importante com relação ao *K-Means* é o de como determinar o número de centróides K. Não há uma solução ideal, uma possível abordagem é plotar o erro em função do K e observar o ponto em que houver uma queda brusca no erro (NG, 2014). Essa abordagem não irá funcionar se a queda do erro em relação ao aumento do K ocorrer de maneira suave.

2.1.2.2 COEFICIENTE DE SILHUETA

Um método intrínseco para avaliar a qualidade dos *clusters* é avaliar o quão bem os *clusters* estão separados e o quão compactos os *clusters* são (HAN et al., 2011).

A métrica para se medir a qualidade dos *clusters* gerados é o coeficiente de silhueta (HAN et al., 2011). Ele é calculado utilizando a distância média intra-cluster para cada exemplo (a) e a distância média para o *cluster* mais próximo para cada exemplo (b). O coeficiente de silhueta para um determinado exemplo é definido pela equação (28). Para medir a qualidade dos *clusters* gerados, pode-se utilizar o valor médio do coeficiente de silhueta para todos os dados (HAN et al., 2011).

$$s = \frac{(b-a)}{max(a,b)} \tag{28}$$

2.2 PROCESSAMENTO DE LINGUAGEM NATURAL

Em um nível prático, todos precisam de ajuda para navegar no universo de informação que é a web. Motores de busca tem sido cruciais para o crescimento e popularidade da web, pois em seus primórdios não havia indexação tão completa de conteúdo. Para um computador extrair informações relevantes de textos livres na web faz-se necessário o emprego de várias tarefas de processamento de linguagem natural (PLN), como por exemplo, extração de informação, inferência e sumarização (BIRD et al., 2009).

No contexto desse trabalho o PLN é utilizado com o objetivo de manipular grandes coleções de texto. São utilizados conceitos de PLN para representar, extrair e fazer o pré-processamento dos dados que são utilizados nos algoritmos de aprendizado de máquina.

A seguir é apresentado um *pipeline* de pré-processamento de textos. Esse *pipeline* é aplicado anteriormente à fase de extração de *features* do texto, de modo que o texto seja normalizado (BAEZA-YATES; RIBEIRO-NETO, 2011). Essa normalização é necessária porque o texto bruto contém muitas informações que não agregam nenhum potencial preditivo aos modelos, sendo assim caracterizadas apenas como ruido nos dados.

Posteriormente, são apresentadas formas de extrair e codificar *features* de texto em vetores de numéricos de *features*, é apresentada a representação por n-gramas. Por último são apresentadas formas de fazer a ponderação dos termos (n-gramas) dos vetores de *features*.

2.2.1 *PIPELINE* DE PRÉ-PROCESSAMENTO DE TEXTO

Segundo Bird et al. (2009) cada documento de texto utilizado nos algoritmos de aprendizado de máquina é codificado na forma de um vetor de *features*. Para obter esses vetores é necessário inicialmente normalizar o texto do documento. Esse processo é ilustrado pelo diagrama da Figura 7.

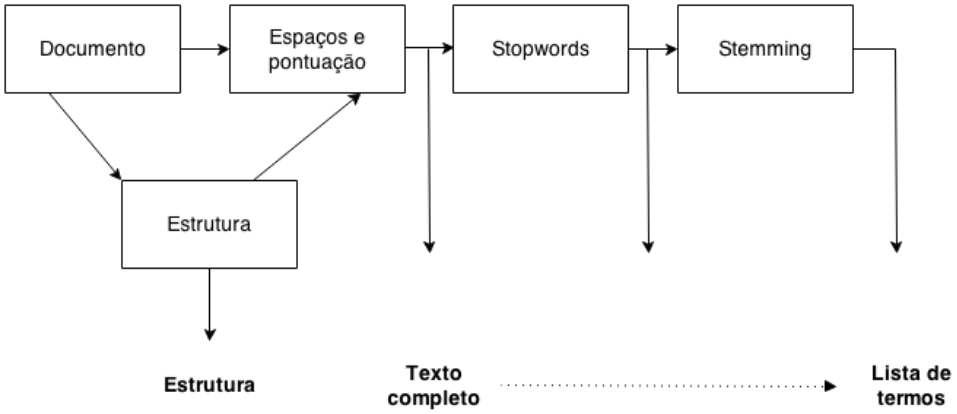

Figura 7: *Pipeline* de Pré-processamento de Texto.
Fonte: Adaptado de Kaestner (2014)

O texto do documento em sua forma bruta possui uma estrutura determinada pela ordem das palavras e elementos presentes. Após a normalização do documento ele irá perder essa estrutura, tornando-se uma lista de termos (KAESTNER, 2014).

Inicialmente, o texto passa por um processo de *tokenização*, em que os *tokens* são obtidos por meio de algum critério de separação de palavras presentes no texto. O critério de separação mais comum é o espaço em branco e finais de linha (BAEZA-YATES; RIBEIRO-NETO, 2011).

Uma vez em posse da lista de *tokens* extraídos, o *pipeline* passa à fase de remoção de pontuação, números e quaisquer outros caracteres não textuais. Nessa fase é também realizada a normalização de caixa para maiúsculas ou minúsculas para evitar redundância (BAEZA-YATES; RIBEIRO-NETO, 2011).

Posteriormente, o *pipeline* realiza a remoção de *stopwords* da lista de *tokens*. As *stopwords* são as palavras que não agregam nenhuma conotação semântica ao texto, como por exemplo “o”, “a”, “os”, “as”, “aquele”, “aquela”, “aquilo” (BAEZA-YATES; RIBEIRO-NETO, 2011).

Por fim, o *pipeline* realiza a compressão dos *tokens* por meio de um processo chamado

stemming (BIRD et al., 2009). O *stemming* elimina os sufixos das palavras, comprimindo-as para a raiz. Por exemplo, as palavras “trabalho”, “trabalhar” e “trabalhou” são comprimidas para a raiz “trabalha”, eliminando assim as redundâncias. No final, obtém-se uma sequência de elementos textuais que têm uma semântica significativa para o problema (BAEZA-YATES; RIBEIRO-NETO, 2011).

2.2.2 VETORIZAÇÃO DAS *FEATURES* TEXTUAIS

Uma coleção de m documentos pode ser representada por meio de uma tabela de m linhas por n colunas. Uma entrada na tabela corresponde ao peso de um determinado termo no documento, zero significa que o termo não tem nenhuma significância no documento ou que ele simplesmente não existe no documento. Esse tipo de representação é muitas vezes chamado de *bag-of-words* (KO, 2012).

Após a normalização de todos os documentos $d_i \in \{d_1, ..., d_m\}$ é computado um vocabulário V, em que cada termo $t_j \in \{t_1, \ldots, t_n\}$ do documento d_i aparece no vocabulário V apenas uma única vez.

O Quadro 2 apresenta um exemplo de uma coleção de três documentos. Após terem sido normalizados esses documentos podem ser representados pelas listas dos respectivos *tokens*, como mostra o Quadro 3. Note que as palavras “venceu”, “vencerá” e “pesquisas” foram comprimidas pelo processo de *stemming*. A representação *bag-of-words* dessa coleção de documentos é dada pelo Quadro 4. O vocabulário computado é representado pelo cabeçalho da tabela e cada entrada na tabela representa a presença ou ausência do termo do vocabulário no texto.

Quadro 2: Coleção de Três Documentos.

Documento	Texto
a	Ela venceu a ultima eleição !!!
b	O candidato lidera as pesquisas.
c	Ela não vencerá :)

Fonte: Autoria própria.

No exemplo anterior a representação *bag-of-words* dos documentos utiliza n-gramas como os termos do vocabulário. Mais especificamente, são utilizados uni-gramas, ou seja, apenas uma palavra. Também poderiam ser utilizados bi-gramas, tri-gramas ou uma combinação destes. Em uma representação por bi-gramas por exemplo, a lista de *tokens* de um determinado documento é uma lista de pares de termos que aparecem juntos, como por exemplo “ela venceu” no documento a.

Quadro 3: Coleção de Três Documentos Normalizados.

Documento	Lista de *tokens*
a	$[ela, vence, ultima, eleição]$
b	$[candidato, lidera, pesquisa]$
c	$[ela, não, vence]$

Fonte: Autoria própria.

Quadro 4: Representação *Bag-of-words*.

Documento	ela	vence	ultima	eleição	candidato	lidera	pesquisa	não
a	1	1	1	1	0	0	0	0
b	0	0	0	0	1	1	1	0
c	1	1	0	0	0	0	0	1

Fonte: Autoria própria.

Outra maneira de codificar os elementos textuais na representação *bag-of-words* é por meio de um *part-of-speech-tagger* (BIRD et al., 2009). O *part-of-speech-tagger* cuida do processo de classificar palavras em suas categorias léxicas e rotulá-las de acordo. Nesse caso, é utilizada uma coleção de *tags* para rotular os termos de cada documento, em um processo muito semelhante ao rotulamento por n-gramas descrito anteriormente.

2.2.3 MEDIDAS DE PONDERAÇÃO

Na seção anterior foi abordada uma medida de ponderação binária (ou seja se o n-grama está presente ou não no documento) para os valores das *features* extraídas do texto. A seguir é apresentado uma outra medida de ponderação para as *features* baseada na frequência dos n-gramas.

A frequência normalizada de um termo em um documento geralmente é um bom indicativo do tópico do documento (RAJARAMAN; ULLMAN, 2011). A equação (29) representa a frequência do termo *i* no documento *j* normalizada pela frequência do termo mais frequente (RAJARAMAN; ULLMAN, 2011).

$$TF_{i,j} = \frac{f_{i,j}}{MAX_k f_{k,j}} \tag{29}$$

A estatística tf-idf (*term frequency–inverse document frequency*) tem o objetivo de medir o quão importante uma palavra é para um documento em uma coleção de documentos (RAJARAMAN; ULLMAN, 2011). Termos que aparecem em muitos documentos diferentes são menos indicativos de tópico. A equação (30) representa o *logaritmo* na base 2 da razão

entre o número de documentos e em quantos documentos o termo *i* aparece, um valor indicativo do poder de discriminação do termo.

$$IDF_i = log_2(\frac{N}{n_i}) \tag{30}$$

O valor tf-idf aumenta proporcionalmente com o número de vezes que a palavra aparece no documento, mas é afetado pela frequência da palavra na coleção de documentos como um todo, o que ajuda a controlar o fato de que algumas palavras são mais comuns do que outras, geralmente sendo uma boa métrica para eliminar *stopwords* (RAJARAMAN; ULLMAN, 2011). O score tf-idf para o termo *i* no documento *j* é definido então pelo produto $TF_{i,j} \times IDF_i$. Os termos com as maiores pontuações tf-idf são geralmente os termos que melhor caracterizam o tópico do documento.

2.3 TRABALHOS CORRELATOS

O Twitter é um serviço popular de mídia social que permite a seus usuários postarem mensagens curtas, ou *tweets*, que podem conter até 140 caracteres (JAVA et al., 2007). Usuários do Twitter podem anotar suas postagens com uma ou mais *hashtags* (#) para indicar o assunto da mensagem. *Tweets* que possuem *hashtags* são tratados de maneira especial pelo Twitter, possibilitando que fluxos de *tweets* contendo as mesmas *hashtags* sejam lidos em sequência pelos usuários (BECKER et al., 2011).

O Twitter utiliza um algoritmo proprietário para gerar um ranking das *hashtags* mais populares (BECKER et al., 2011). *Tweets* com essas *hashtags* podem conter termos e frases que representam algum comportamento de tendência. Esses tópicos mais populares podem vir a se referir a algum evento que esteja acontecendo (ex: # copaDoMundo) ou a algum *meme* [4].

A troca de informações no Twitter pode ser representada por um grafo direcionado $G = (U, E)$, em que usuários U estão conectados uns aos outros através de links E. Um link $E_{i,j}$ de um usuário u_i para outro usuário u_j significa que u_i é um seguidor de u_j, e u_j é um amigo de u_i. O total de amigos de u_i é seu grau de saída e o total de seguidores de u_j é o seu grau de entrada (ZHANG et al., 2013). Cada *tweet* de um usuário será enviado para todos os seus seguidores, essas ligações unidirecionais são então a unidade básica de difusão das informações.

Usuários do Twitter também podem se comunicar uns com os outros por meio dos mecanismos de *retweet* e menção. O *retweet* funciona como o encaminhamento tradicional de

[4] http://en.wikipedia.org/wiki/Internet_meme

e-mails e é identificado pelo uso de RT @nomeUsuario. A menção é o uso de @nomeUsuario em um *tweet* para se fazer referência ao mesmo (ZHANG et al., 2013).

Em um estudo feito por Tonkin et al. (2012), indicou-se que em tópicos orientados a eventos, como por exemplo no caso de desastres naturais, as informações são mais difundidas pelo mecanismo de *retweet*, onde a maioria das informações são comentários posteriores ao evento ou provenientes de agências de notícias.

Com relação a tarefa de sumarização de eventos, o uso de meta-informações presentes nos *tweets*, como entidades nomeadas, geolocalização e sentimento podem ser utilizadas para se analizar e explorar eventos (DOU et al., 2012). O sistema Tweetinfo proposto por Marcus et al. (2011), permite que os usuários naveguem por uma grande coleção de *tweets* utilizando uma visualização por linha do tempo para exibir picos de alta atividade de *tweets* relacionados a um evento. Usuários podem ainda analisar sub-eventos e explorá-los por meio de geolocalização, sentimento e *urls*[5] mais populares (MARCUS et al., 2011).

Sakaki et al. (2010) utilizaram o Twitter como uma rede de sensores para detectar eventos em tempo real, mais especificamente desastres naturais como terremotos, tufões e tsunamis. Foram coletados *tweets* no Japão que continham termos relacionados a esse tipo de evento, como por exemplo “terremoto” ou “tremor”. Foi construído um classificador baseado em SVM para detectar um evento alvo e foram utilizadas *features* como palavras-chave em um *tweet*, número de palavras, e seu contexto.

Go et al. (2009) introduzem uma nova abordagem para classificar automaticamente o sentimento de *tweets*. Os *tweets* são classificados como positivos ou negativos, baseado em um determinado termo de busca. Os dados de treinamento consistem em *tweets* com *emoticons* que funcionam como rótulos ruidosos. Esse tipo de dado é abundante e fácil de ser obtido automaticamente. Foram utilizados algoritmos de treinamento como SVM, *Naïve-bayes* e Máxima Entropia, utilizando n-gramas como *features* para os exemplos de treinamento.

Becker et al. (2011) argumentam que o *microblogging* em mídias sociais emergiu como um poderoso mecanismo para detectar eventos do mundo real. Os autores utilizam um algoritmo de agrupamento *online* e incremental para distinguir *tweets* sobre eventos do mundo real de *tweets* não relacionados a eventos. Foram utilizadas como *features* o número de *retweets*, o volume de tráfego, a coesão entre termos, entre outras *features*. Essa abordagem analisa um fluxo de *tweets* em tempo real, considerando cada *tweet* e atribuindo-o a um determinado *cluster* dependendo da similaridade do *tweet* para os *clusters* obtidos até então, dessa forma os *clusters* estão constantemente se modificando com o passar do tempo. Posteriormente, é utilizado um

[5] *Uniform Resource Locators*

modelo de classificação que dada uma representação de *features* de um *cluster*, decide se o *cluster* e seus *tweets* contém informações sobre o evento.

3 DESENVOLVIMENTO

No desenvolvimento da ferramenta foi utilizada a linguagem de programação Python na versão 3.4 [1]. Foram utilizadas algumas bibliotecas e API's específicas, dentre elas:

- *Twitter Search API*[2]: utilizada para obter *tweets* e suas respectivas meta-informações diretamente da base do Twitter.
- NLTK (*Natural Language Toolkit*) [3]: utilizada para realizar o processo de *stemming* em textos de língua portuguesa.
- *Scikit-learn* [4]: utilizada para realizar a extração de *features* textuais e treinamento de algoritmos de aprendizado de máquina.

O código fonte da ferramenta está disponível para *download* e visualização diretamente na plataforma de projetos *open source* github [5].

3.1 DESCRIÇÃO DA FERRAMENTA

O usuário da ferramenta, a partir de algum contexto de interesse, entrará com alguma consulta por palavras-chave para coletar *tweets* que serão armazenados em uma base de dados. Os *tweets* coletados são *clusterizados* e posteriormente analisados. O processo de análise gera informações de sumarização sobre o texto das mensagens.

O usuário também entrará com consultas por palavras-chave para coletar dados de treinamento para treinar um classificador de sentimentos, e o usuário deve escolher um classificador: *Naive-Bayes* ou SVM. Esse classificador é utilizado sobre os *tweets clusterizados* para auxiliar no processo de geração de informações de sumarização.

[1] https://www.python.org/
[2] https://dev.twitter.com/docs/using-search
[3] http://www.nltk.org/
[4] http://scikit-learn.org/stable/
[5] https://github.com/atrevisan/twitter_analysis

Para realizar a *clusterização* e o treinamento do algoritmo de classificação de sentimentos, a ferramenta deve auxiliar o usuário no processo de extração de *features* textuais para compor um vocabulário. As *features* textuais são n-gramas. O processo de extração de n-gramas implementado na ferramenta necessita que o usuário escolha: uma combinação de tipos de n-gramas (uni-gramas, bi-gramas e tri-gramas), um limite máximo para a quantidade de n-gramas selecionados e um número mínimo e máximo de *tweets* em que os n-gramas devem estar presentes. A ferramenta então seleciona automaticamente os n-gramas e exibe a quantidade de n-gramas obtidos. O usuário também deve selecionar algumas opções de pré-processamento: aplicar *stemming* e eliminar *stopwords*. O processo de eliminação de *stopwords* examina uma lista com cerca de 258 *stopwords* mais comuns na língua portuguesa. Para realizar a vetorização dos *tweets* em vetores numéricos de *features* há duas opções de escolha: frequência absoluta e *tf-idf*. Caso seja selecionado *tf-idf* ainda é possível marcar a opção para normalizar por *idf*. Independentemente do método de vetorização escolhido, também é possível marcar uma opção para binarizar os valores para as *features*, nesse caso é computado apenas se o *n-grama* está presente ou não no *tweet*.

Para gerar as informações de sumarização, o usuário pode selecionar as seguintes opções no menu principal da ferramenta:

- Nuvem de palavras: gera uma nuvem de palavras com os n-gramas de maior pontuação, quanto maior a pontuação maior o tamanho da fonte.
- Nuvem de palavras por *cluster*: gera uma nuvem de palavras para o *cluster* selecionado.
- Análise de sentimentos: gera um gráfico de pizza que mostra a proporção de sentimento positivo e negativo nos *tweets*.
- Análise de sentimentos por *cluster*: gera um gráfico de pizza com a proporção de sentimento positivo e negativo para o *cluster* selecionado.
- Análise de n-gramas: o usuário pode escolher um n-grama e gerar informações sobre ele no decorrer do tempo: frequência, *retweets*, *tweets* positivos e *tweets* negativos.
- Análise de n-gramas por *cluster*: o usuário pode escolher um n-grama e um *cluster* e gerar informações sobre o n-grama no decorrer do tempo para o *cluster* selecionado.
- Análise de *hashtags*: o usuário pode escolher uma *hashtag* e gerar informações sobre ela no decorrer do tempo: frequência, *retweets*, *tweets* positivos e *tweets* negativos.
- Análise de *hashtags* por *cluster*: o usuário pode escolher uma *hashtag* e um *cluster* e gerar informações sobre a *hashtag* no decorrer do tempo para o *cluster* selecionado.

3.2 COLETA, PRÉ-PROCESSAMENTO E ANÁLISE DE *TWEETS*

A seguir, na Figura 8 é apresentada a ideia geral por trás do processo envolvendo a coleta, pré-processamento e análise dos *tweets*.

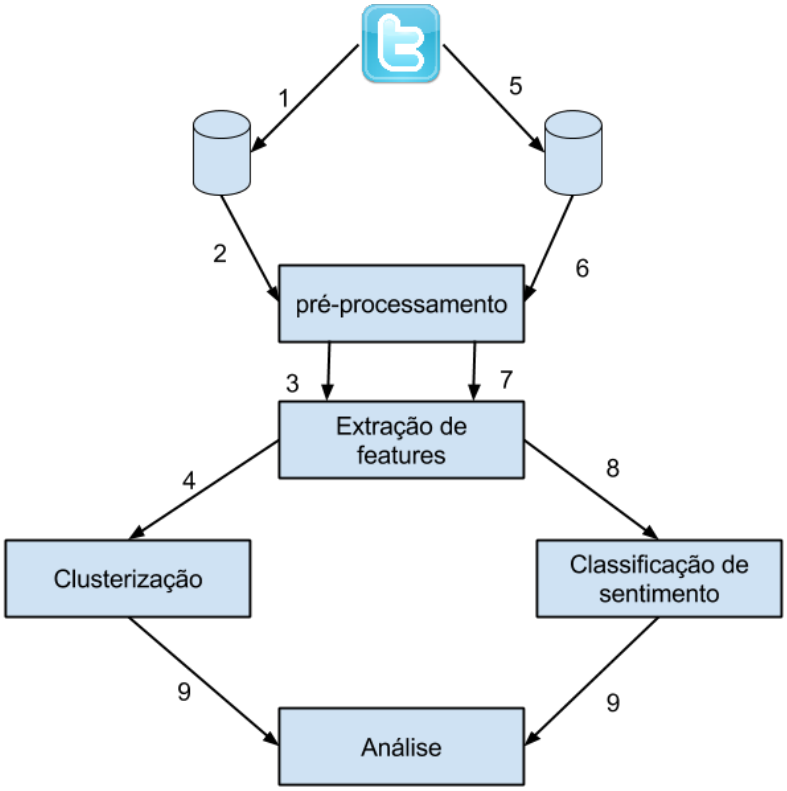

Figura 8: Coleta, Pré-processamento e Análise de *Tweets*.
Fonte: Autoria própria

1. Os *tweets* relacionados ao evento alvo de interesse são coletados e armazenados em uma base de dados. Para cada *tweet* são armazenadas suas meta-informações associadas, data e hora de publicação, número de *retweets*, texto da mensagem e a latitude e longitude do usuário que publicou o *tweet* (quando disponível).

2. Os *tweets* irão passar um a um pelo *pipeline* de pré-processamento definido na seção 2.2.1. São realizados ainda alguns pré-processamentos adicionais, baseando-se na abordagem apresentada em Go et al. (2009):

   - Palavras são normalizadas para minúsculas. Por exemplo, “Palavra”, “palavrA” e “PALAVRA” são convertidas para “palavra”.

- Nomes de usuários são substituídos pela classe de equivalência “AT_USER”. Por exemplo, “@nome” é convertido para “AT_USER”.
- *Links* são substituídos pela classe de equivalência “URL”. Por exemplo, a url “http://tinyurl.com/cvvg9a” é substituída por “URL”.
- As classes de equivalência são adicionadas à lista de *stopwords*. O processo de eliminação de *stopwords* examina cada *tweet*, um a um, se o *tweet* possuir palavras que estejam na lista de *stopwords*, essas palavras são então eliminadas do *tweet*.
- Letras repetidas são substituídas por no máximo duas repetições. Por exemplo, o *token* “fooome!!” é substituído por “foome!!”.
- *Retweets* são removidos. *Retweets* são cópias de *tweets* de outros usuários que são postados em uma conta diferente. Dessa forma evita-se colocar peso extra a qualquer *tweet* particular.
- *Hashtags* são convertidas para a representação da palavra sem a *hashtag*. Por exemplo, a *hashtag* “#operaçãoBetaLab” é convertida para “operaçãoBetaLab”.

3. As *features* utilizadas são n-gramas. Os n-gramas são grupos de palavras que aparecem juntas com frequência no texto dos *tweets*. Por exemplo, “greve”, “greve professores” e “greve professores estaduais” são exemplos de uni-gramas, bi-gramas e tri-gramas respectivamente. É computado um vocabulário com diferentes combinações de uni-gramas, bi-gramas e tri-gramas. Esse vocabulário é utilizado para fazer a representação *bag-of-words* dos *tweets*, como explicado na seção 2.2.2. É utilizada então alguma das ponderações apresentadas nas seções 2.2.2 e 2.2.3 para os elementos dos vetores de *features*, binária, frequência absoluta ou *tf-idf*.

4. O algoritmo de *clusterização K-Means* é aplicado sobre os vetores de *features* obtidos no passo anterior. São então obtidos os centróides dos *clusters* de *tweets*. Para se determinar a qual *cluster* cada *tweet* pertence, é calculada a distância euclidiana entre *tweet* e centróides (HAN et al., 2011). Cada *tweet* é atribuído ao centróide que estiver mais próximo. É então avaliada a qualidade dos *clusters* com o coeficiente de silhueta, seção 2.1.2.2. O valor para o coeficiente de silhueta varia entre -1 e 1, dessa forma é possível avaliar a qualidade das *features* selecionadas, podendo então escolher combinações e quantidades diferentes de n-gramas para as *features*, e também valores diferentes para a constante K do algoritmo *K-Means*. Por fim, para cada *tweet* é então salvo um *label* indicando a qual *cluster* ele pertence, dessa forma é possível saber a qual *cluster* os *tweets* pertencem na fase de análise.

5. São coletados e armazenados os *tweets* utilizados como exemplos de treinamento para o classificador de sentimentos. Para a tarefa de classificação de sentimento é utilizada a abordagem de "supervisão distante", apresentada em (GO et al., 2009). Nessa abordagem os dados de treinamento consistem em *tweets* com *emoticons*, que funcionam como rótulos ruidosos. Dentre os dados coletados, 80% são utilizados como dados de treinamento e 20% são utilizados para teste.

6. O pré-processamento dos *tweets* prossegue como na fase 2. Há múltiplos *emoticons* que podem expressar emoção positiva e emoção negativa. Por exemplo, ":)" e ": −)" ambos expressam emoção positiva, enquanto que ": (" e ": −(" expressam ambos emoção negativa. Como existem múltiplos *emoticons* que podem expressar uma emoção positiva ou negativa, eles são então normalizados, situação apresentada no Quadro 5. Adicionalmente, *tweets* ambíguos são removidos. Se um *tweet* possuir *emoticons* positivos e negativos ao mesmo tempo ele é eliminado do conjunto de treinamento, dessa forma evita-se ruído excessivo nos dados de treinamento.

7. As *features* utilizadas no classificador de sentimentos são extraídas da mesma maneira descrita no passo 3, relacionado a tarefa de *clusterização*. Tanto os exemplos de treinamento quanto os exemplos de teste são codificados baseados no mesmo vocabulário extraido.

8. Para o treinamento do classificador de sentimentos há duas opções de algoritmos: *Naive-Bayes* e SVM. Após terem sido treinados os algoritmos são comparados, dessa forma pode-se avaliar qual o melhor modelo obtido para classificar o sentimento dos *tweets* relacionados ao evento alvo de interesse. Os algoritmos são avaliados em termos de precisão, *recall* e medida F sobre os exemplos de teste, seção 2.1.1.7. Para o SVM é possível ainda alterar o parâmetro C, dessa forma é possível avaliar a melhor combinação entre algoritmo e vocabulário de *features*.

9. Por último, em posse dos *clusters* dos *tweets* do evento alvo de interesse e do classificador de sentimento, é feita a análise dos *tweets* do evento alvo. A análise consiste em gerar uma nuvem de palavras para os n-gramas de maior pontuação, determinar o proporção de *tweets* com sentimento positivo/negativo, analisar a distribuição dos *n-gramas* de maior pontuação no decorrer do tempo, analisar a distribuição das *hashtags* mais populares no decorrer do tempo e exibir todas essas informações separadamente por *cluster*.

Quadro 5: Lista de *Emoticons*.

Emoticons mapeados para :)	*Emoticons* mapeados para :(
:)	:(
:-)	:-(
:)	: (
:D	
=)	

Fonte: Adaptado de Go et al. (2009).

3.3 ANÁLISE E PROJETO DA FERRAMENTA

Nas sub-seções seguintes são apresentadas informações relevantes relacionadas a análise e projeto da ferramenta desenvolvida: requisitos funcionais, requisitos não funcionais, o diagrama de casos de uso do sistema e o diagrama de classes do sistema.

3.3.1 REQUISITOS FUNCIONAIS

- RF01: a ferramenta deve permitir que o usuário entre com uma busca por palavras-chave para realizar o *download* de *tweets* para um arquivo csv [6].
- RF02: a ferramenta deve permitir que o usuário escolha combinações de diferentes tipos de n-gramas para serem extraídos dos *tweets* coletados: uni-gramas, bi-gramas e tri-gramas.
- RF03: a ferramenta deve permitir que o usuário escolha diferentes tipos de ponderações para vetorizar os *tweets* em vetores numéricos de *features*: *tf-idf*, freqência absoluta e binário.
- RF04: a ferramenta deve fornecer opções para reduzir a quantidade de *features* selecionadas: limite máximo, número mínimo e máximo de documentos que devem conter a *feature*, *stemming* das *features* e remoção de *stopwords*.
- RF05: a ferramenta deve permitir que o usuário entre com um número de componentes para aplicar um método de redução de *features*.
- RF06: a ferramenta deve permitir que o usuário selecione um arquivo csv contendo *tweets* e entre com a quantidade de *clusters* desejados para aplicar o algoritmo *K-Means* sobre os *tweets* e posteriormente exibir o coeficiente de silhueta para os *clusters* obtidos.

[6] *Comma Seperated Values*

- RF07: a ferramenta deve permitir que o usuário carregue dois arquivos csv: um contendo *tweets* positivos e o outro *tweets* negativos, em que 80% dos dados são utilizados para treinar um classificador de sentimento.
- RF08: a ferramenta deve permitir que o usuário escolha um algoritmo de classificação para treinar o classificador de sentimento: *Naive-Bayes* ou SVM.
- RF09: a ferramenta deve exibir um relatório do processo de treinamento do classificador com a análise de erro feita sobre os 20% dos dados não utilizados no treinamento.
- RF10: a ferramenta deve exibir uma nuvem de palavras com as *features* de maior pontuação, obtidas durante o processo de *clustering*.
- RF11: a ferramenta deve exibir uma nuvem de palavras com as *features* de maior pontuação, obtidas durante o processo de *clustering* para um determinado *cluster* selecionado.
- RF12: a ferramenta deve exibir um gráfico de pizza com a proporção de *tweets* positivos/negativos.
- RF13: a ferramenta deve exibir um gráfico de pizza com a proporção de *tweets* positivos/negativos para um determinado *cluster* selecionado.
- RF14: a ferramenta deve permitir que o usuário escolha um n-grama dentre os n-gramas de maior pontuação e exiba informações (frequência, quantidade de *retweets* relacionados, quantidade de *tweets* positivos relacionados e quantidade de *tweets* negativos relacionados) sobre o n-grama no decorrer do tempo (mês, dia e hora).
- RF15: a ferramenta deve permitir que o usuário escolha um n-grama dentre os n-gramas de maior pontuação e exiba informações (frequência, quantidade de *retweets* relacionados, quantidade de *tweets* positivos relacionados e quantidade de *tweets* negativos relacionados) sobre o n-grama no decorrer do tempo (mês, dia e hora) para um determinado *cluster* selecionado.
- RF16: a ferramenta deve permitir que o usuário escolha uma *hashtag* dentre as *hashtags* mais populares e exiba informações (frequência, quantidade de *retweets* relacionados, quantidade de *tweets* positivos relacionados e quantidade de *tweets* negativos relacionados) sobre a *hashtag* no decorrer do tempo (mês, dia e hora).
- RF17: a ferramenta deve permitir que o usuário escolha uma *hashtag* dentre as *hashtags* mais populares e exiba informações (frequência, quantidade de *retweets*

relacionados, quantidade de *tweets* positivos relacionados e quantidade de *tweets* negativos relacionados) sobre a *hashtag* no decorrer do tempo (mês, dia e hora) para um determinado *cluster* selecionado.

3.3.2 REQUISITOS NÃO FUNCIONAIS

- RNF01: a ferramenta deve ser desenvolvida em Python.
- RNF02: a ferramenta deve integrar com o Twitter através da API pública fornecida.
- RNF03: as tarefas relacionadas a aprendizado de máquina devem ser implementadas com a biblioteca *Scikit-learn*.
- RNF04: a funcionalidade de *stemming* deve ser realizada pela biblioteca NLTK.

3.3.3 DIAGRAMA DE CASOS DE USO

Na Figura 9 é representado o diagrama de casos de uso do sistema, são representadas as principais interações entre o usuário da ferramenta e o sistema e as principais interações entre o sistema e atores externos.

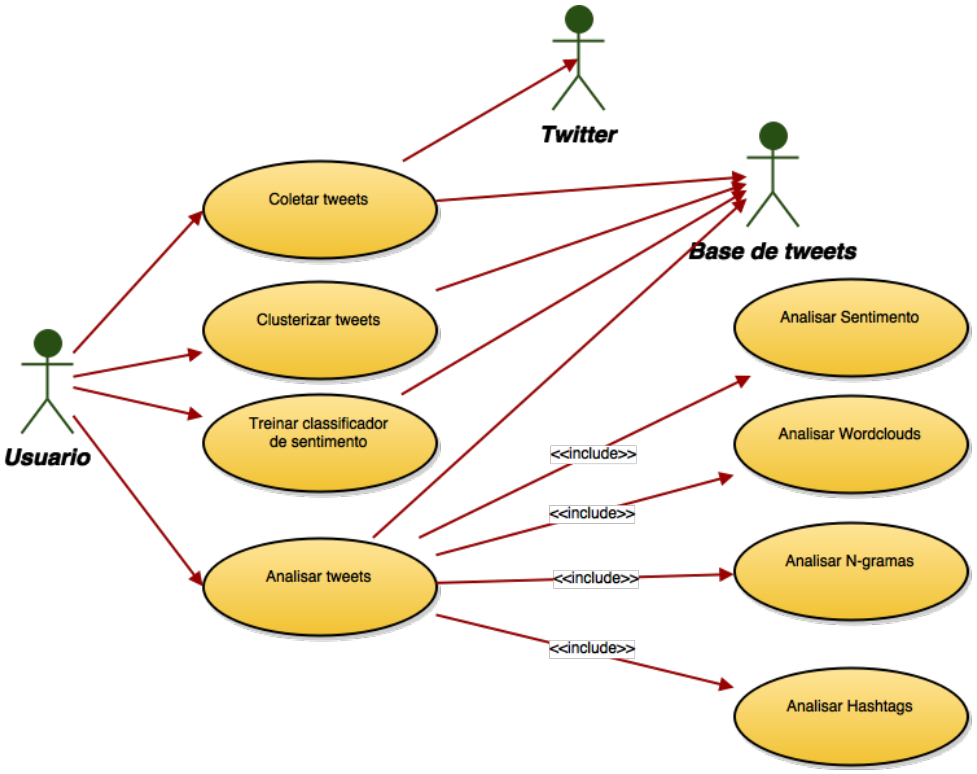

Figura 9: Casos de Uso.

Fonte: Autoria própria

3.3.4 DIAGRAMA DE CLASSES

Na Figura 10 é representado o diagrama de classes do sistema. Esse diagrama representa de forma simplificada as principais classes envolvidas em todo o processo de coleta, pré-processamento, extração de *features* e aplicação de algoritmos de aprendizagem.

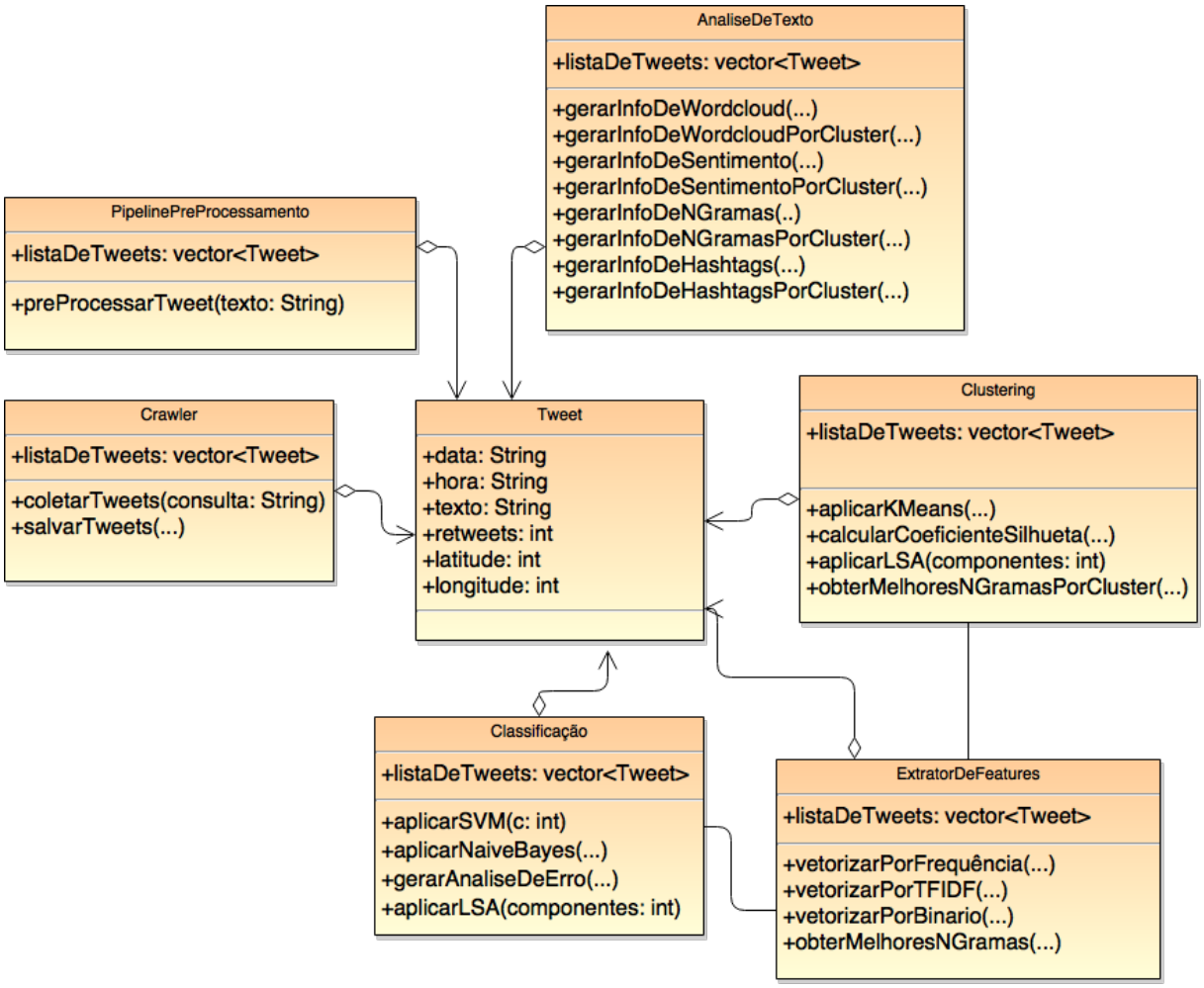

Figura 10: Diagrama de Classes.

Fonte: Autoria própria

3.4 FUNCIONALIDADES DA FERRAMENTA

A seguir são descritas as principais funcionalidades da ferramenta e também são levantados alguns pontos importantes com relação ao processo de desenvolvimento.

3.4.1 COLETA DE *TWEETS*

A funcionalidade de coleta de *tweets* pode ser visualizada pela Figura 11. É utilizada a API *Search* do Twitter para coletar *tweets* que ocorreram no passado. A API *Search* busca por todos os *tweets* que correspondam à consulta e que ocorreram até uma semana no passado. A API na sua versão gratuita possui uma limitação de recuperar apenas 180 *tweets* a cada 15 minutos.

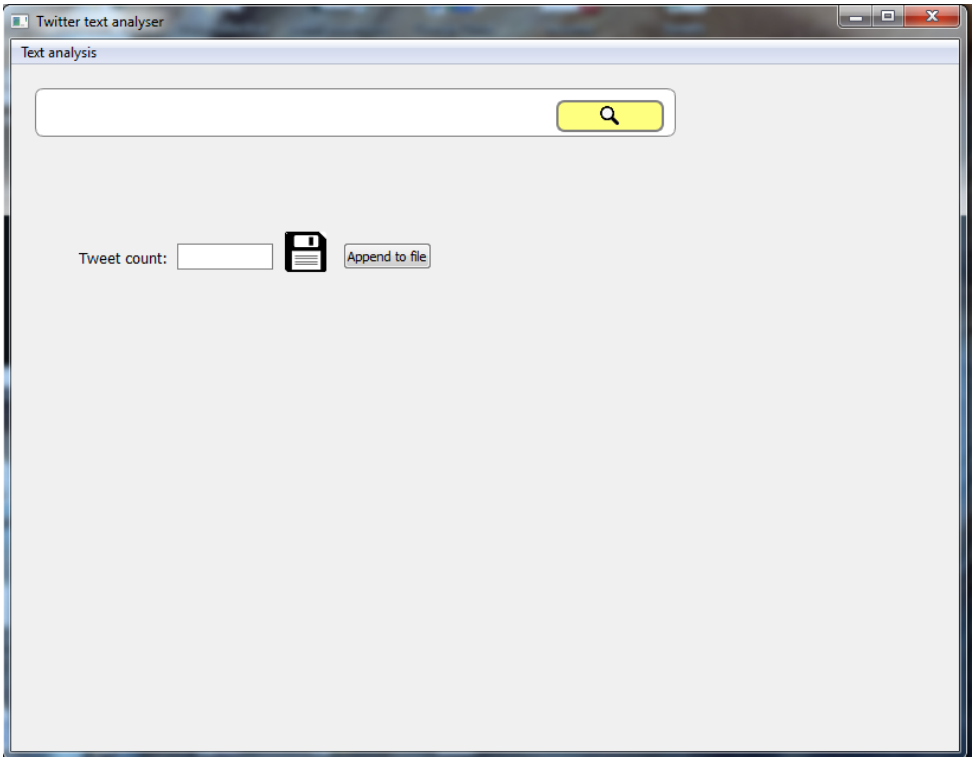

Figura 11: Coleta de *Tweets*.

Fonte: Autoria própria

A ferramenta exibe para o usuário a quantidade de *tweets* recuperados até então. Para finalizar o processo de coleta o usuário pode clicar no botão de salvar para salvar os *tweets* para um arquivo csv. Também é possível incluir os *tweets* coletados em um arquivo já existente, essa funcionalidade é útil para se coletar *tweets* por um período estendido de tempo. Para cada *tweet* são salvos a sua data e horário de publicação, seu texto, a quantidade de *retweets* e a latitude e longitude do usuário que o publicou (quando disponível).

3.4.2 CLUSTERIZAÇÃO DE *TWEETS*

Posteriormente a coleta dos *tweets* que se quer analisar, é realizado o processo de clusterização sobre esses *tweets*, como pode ser visualizado pela Figura 12. Esse processo irá

descobrir a qual *cluster* cada *tweet* pertence.

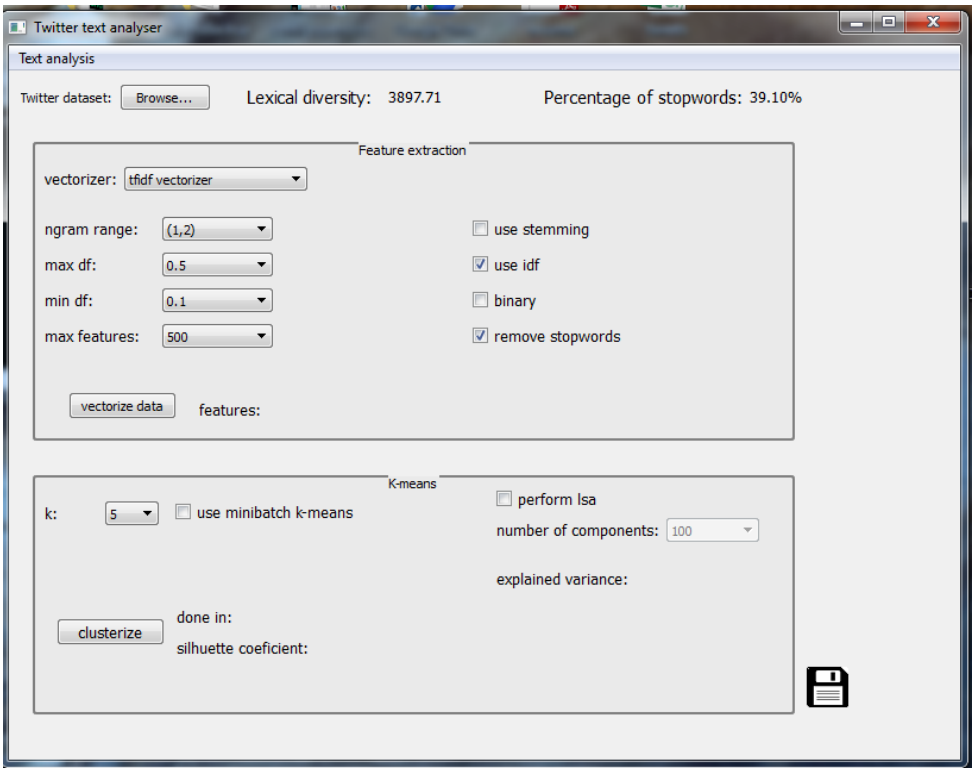

Figura 12: Clusterização de *Tweets*.

Fonte: Autoria própria

Inicialmente o usuário deve navegar até o diretório em que foi salvo o arquivo csv contendo os *tweets*. Após carregado o arquivo csv, é exibida a diversidade léxica, isso é, a razão entre o número de ocorrências e o número de ocorrências únicas de cada palavra na coleção de *tweets*. Também é exibido qual a proporção de palavras no texto é composta de *stopwords*. Essas informações podem ser relevantes para ajustar os parâmetros do processo de extração de *features*.

Antes de aplicar o algoritmo K-Means sobre os *tweets* é necessário extrair as *features* que irão compor o vocabulário que será utilizado para vetorizar os *tweets*. Antes de extrair as *features* é aplicado o *pipeline* de pré-processamento.

O usuário deve escolher qual método de vetorização será utilizado para representar os exemplos de treinamento: tf-idf ou a frequência absoluta. Se for escolhido tf-idf o usuário pode marcar a opção use idf para normalizar o valor calculado pela frequência inversa. Também podem ser marcadas as opções de utilizar *stemming* para remover sufixos, remover *stopwords* ou binarizar os valores das *features*, nesse caso será computado apenas se a *feature* está presente (1) ou não (0).

O usuário pode escolher entre uni-gramas, bi-gramas, tri-gramas ou uma combinação deles para compor as *features*. As opções min df e max df servem para limitar os n-gramas escolhidos entre um número mínimo e máximo de documentos (em números absolutos ou percentagem). A opção max *features* serve para colocar um limite superior no número de *features* selecionadas. Após vetorizar os dados é exibido o número de *features* selecionadas. O usuário pode modificar esses parâmetros e vetorizar os dados várias vezes seguidas até obter o número desejado de *features*. Em posse das *features* que irão compor o vocabulário da representação *bag-of-words* segue-se com a vetorização dos *tweets*.

Para realizar a clusterização dos *tweets* e aplicar o algoritmo K-Means o usuário deve selecionar o valor desejado do parâmetro K, isso é, o número de *clusters*. Pode-se marcar a opção de utilizar o K-Means em modo *mini batch* [7]. O *K-Means* em modo *mini batch* é uma variante do *K-Means* que usa subconjuntos dos dados de entrada selecionados aleatoriamente para reduzir o tempo de computação, enquanto tenta otimizar a mesma função objetivo.

A métrica para se medir a qualidade dos *clusters* gerados é o coeficiente de silhueta, que é exibido após a clusterização dos dados.

A ferramenta também conta com a opção de realizar redução de dimensionalidade sobre as *features* selecionadas, de modo que restem apenas as *features* mais informativas. O método empregado é o LSA. O usuário deve escolher o número de componentes desejados, um número estritamente menor que a quantidade de *features*. É exibida a variância de cada exemplo de treinamento transformada por uma projeção para cada componente. Por fim os dados clusterizados podem ser salvos para um arquivo csv com a adição de que para cada *tweet* é incluído o índice do *cluster* a qual ele pertence.

3.4.3 CLASSIFICAÇÃO DE SENTIMENTO

A funcionalidade de classificação de sentimento é ilustrada pela Figura 13. Inicialmente o usuário deve selecionar um arquivo csv contendo *tweets* com sentimento positivo e outro contendo *tweets* com sentimento negativo. Tendo carregado esses dois arquivos o usuário pode proceder à fase de extração de *features* da mesma maneira como descrito na seção anterior.

Tendo extraído as *features*, o usuário pode passar à fase de treinamento do classificador. Os dados são divididos em 80% para treinamento e 20% para teste. Há duas opções de classificadores, SVM com *kernel* linear e *Naïve Bayes* Multinomial. Para o SVM, o usuário

[7] http://scikit-learn.org/stable/modules/clustering.html

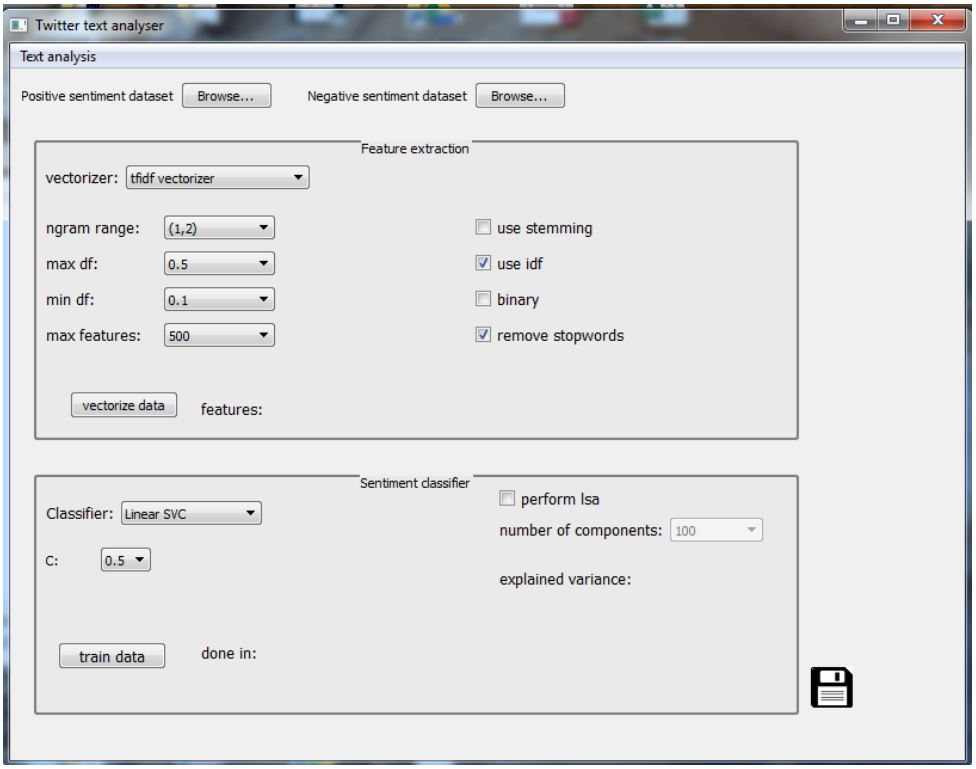

Figura 13: Classificação de Sentimento.

Fonte: Autoria própria

pode ajustar o parâmetro de regularização C. No caso do *Naïve Bayes* a implementação multinomial difere da apresentada na seção 2.1.1.3 pois ela aceita vários tipos de valores para as *features* e não apenas valores binários.

Antes de treinar o classificador, também há a possibilidade de aplicar redução de dimensionalidade com o LSA.

Após realizar o treinamento do algoritmo é exibido um relatório com os valores de precisão, *recall* e medida F para as duas classes do problema.

Por fim, o usuário pode salvar o modelo que acabou de ser treinado para que ele seja utilizado no processo de análise dos *tweets*. O classificador é salvo na forma de um arquivo com a extensão clf.

3.4.4 ANÁLISE DE *WORDCLOUDS*

Após os *tweets* terem sido clusterizados e um modelo de classificação de sentimentos ter sido gerado, o usuário pode passar à fase de análise dos *tweets*. Um *wordcloud* contendo os n-gramas com as maiores pontuações (tf-idf ou frequência dependendo do método de extração

de *features* utilizado no processo de clusterização) é exibido com o objetivo de sumarizar os principais tópicos dos *tweets* coletados. A Figura 14 exibe o *wordcloud* de uma coleção de em torno de 10.000 *tweets*.

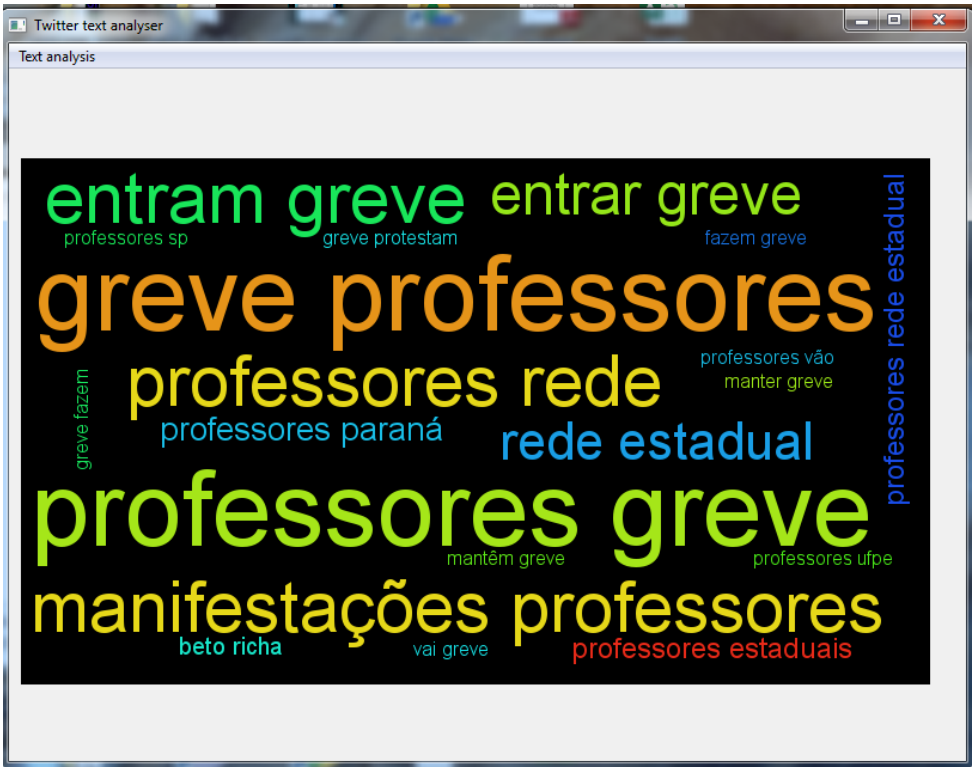

Figura 14: Exibição de *Wordcloud*.

Fonte: Autoria própria

Há também a funcionalidade de exibir um *wordcloud* por *cluster*, como mostra a Figura 15. Nesse caso é exibido uma lista com os índices de todos os *clusters* gerados, onde o usuário pode escolher um *cluster* para que seja gerado um *wordcloud* para os *tweets* do respectivo *cluster*. O processo que escolhe os melhores n-gramas para cada *cluster*, examina os valores das *features* para os centróides obtidos.

3.4.5 ANÁLISE DE SENTIMENTO

Outra informação útil que pode ser gerada sobre os *tweets* é a proporção de sentimento positivo/negativo nas mensagens, um bom indicativo da polarização das opiniões sobre o tópico de interesse. O modelo de classificação gerado é então utilizado para classificar os *tweets* coletados, como mostra a Figura 16.

Como cada *cluster* contém *tweets* semelhantes entre si, também é possível exibir a proporção de sentimento positivo/negativo por *cluster*. Dessa forma é possível ter uma visão

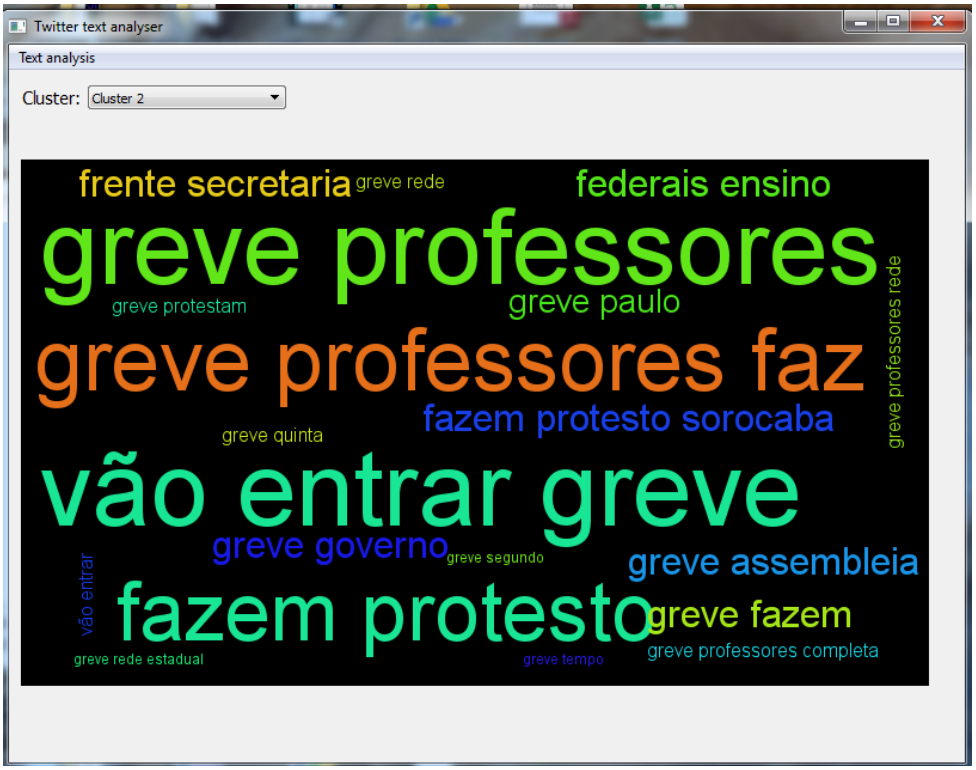

Figura 15: Exibição de *Wordcloud* por *Cluster*.

Fonte: Autoria própria

geral da polarização de opiniões distribuídas por diferentes tópicos.

3.4.6 ANÁLISE DE N-GRAMAS

Outra informação gerada pela ferramenta é a análise de n-gramas, ilustrada na Figura 17. Essa análise exibe uma relação entre os principais n-gramas com as seguintes informações: frequência do n-grama, quantidade de *retweets* relacionados ao n-grama, quantidade de *tweets* com sentimento positivo relacionados ao n-grama e a quantidade de *tweets* com sentimento negativo relacionados ao n-grama. A análise exibe as informações no decorrer dos meses do ano escolhido, dos dias do mês escolhido ou das horas do dia escolhido.

Por exemplo, se forem geradas informações sobre a frequência de um determinado n-grama no decorrer dos dias de um determinado mês, é gerada a frequência total para cada dia, a frequência média no decorrer das horas do dia e a frequência máxima no decorrer das horas do dia. Se a informação for gerada no decorrer dos meses do ano, é exibido o total no mês, a média entre os dias do mês e o máximo entre os dias do mês. Se as informações forem geradas no decorrer das horas do dia, o total é gerado por hora e a média e o máximo são gerados no decorrer dos minutos da hora.

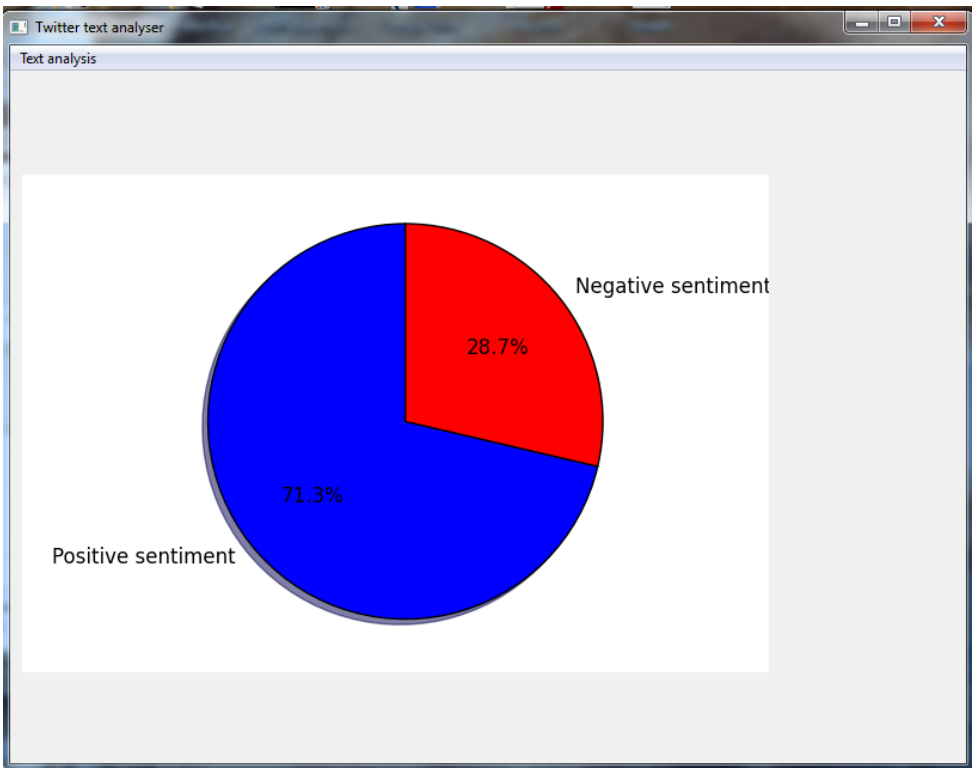

Figura 16: Proporção de Sentimento Positivo/Negativo.

Fonte: Autoria própria

Esse tipo de informação é útil para analisar no decorrer do tempo as mudanças de importância e a relação da palavra com a mudança de opinião das pessoas que a utilizaram, pois uma palavra pode conter conotação positiva em um momento e negativa em outro. Há também a possibilidade de realizar essa mesma análise por *cluster*, nesse caso funciona da mesma maneira como descrito anteriormente, o usuário escolhe um *cluster* e a análise é feita sobre os *tweets* daquele *cluster*.

3.4.7 ANÁLISE DE *HASHTAGS*

Assim como na análise sobre n-gramas, também é possível realizar uma análise sobre as *hashtags*, como mostra a Figura 18. As *hashtags* utilizadas são aquelas mais frequentes na coleção de *tweets*. Deve ser escolhida uma *hashtag*, o tipo de informação desejada e o período de tempo para o qual a informação vai ser gerada. Também é possível realizar a análise de *hashtags* por *cluster*.

Tanto para o exemplo da análise de n-gramas na Figura 17 quanto para a análise de *hashtags* na Figura 18, foram selecionadas a *hashtag* e o n-grama com a pontuação mais alta, ambos para o mês de maio. É possível observar que o período de tempo com a maior

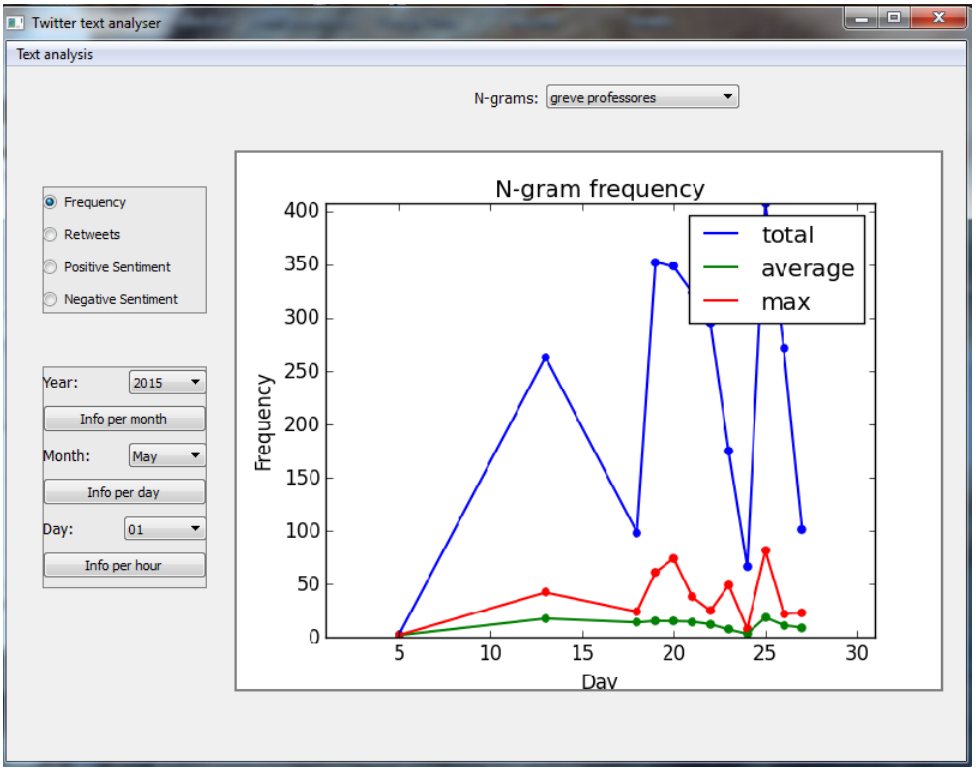

Figura 17: Análise de N-gramas.

Fonte: Autoria própria

concentração de menções a esses termos se dá entre os dias 20 e 25. Também é possível observar que o uso da *hashtag* "OperaçãoBetaLab" começa a se difundir antes do uso do n-grama "greve professores", em termos totais o uso do n-grama possui tendência de crescimento enquanto que o uso da *hashtag* possui tendência de queda.

3.5 VALIDAÇÃO DA FERRAMENTA

Para realizar a validação da ferramenta, foi analisado um evento que ganhou uma grande repercussão no Twitter brasileiro. O evento analisado foi referente a protestos de professores da rede pública de ensino do Paraná. Foram coletados cerca de 10.000 *tweets* no período de 30 de abril de 2015 e 27 de maio de 2015. A consulta por palavras-chave utilizada para coletar os *tweets* foi "manifestações professores".

A tarefa de clusterização foi aplicada quatro vezes, utilizando diferentes combinações e quantidades de n-gramas e diferentes métodos para vetorizar os dados. Nessas quatro vezes, os *tweets* foram clusterizados em dez *clusters* diferentes.

Para realizar o treinamento do classificador de sentimentos, foram coletados em torno

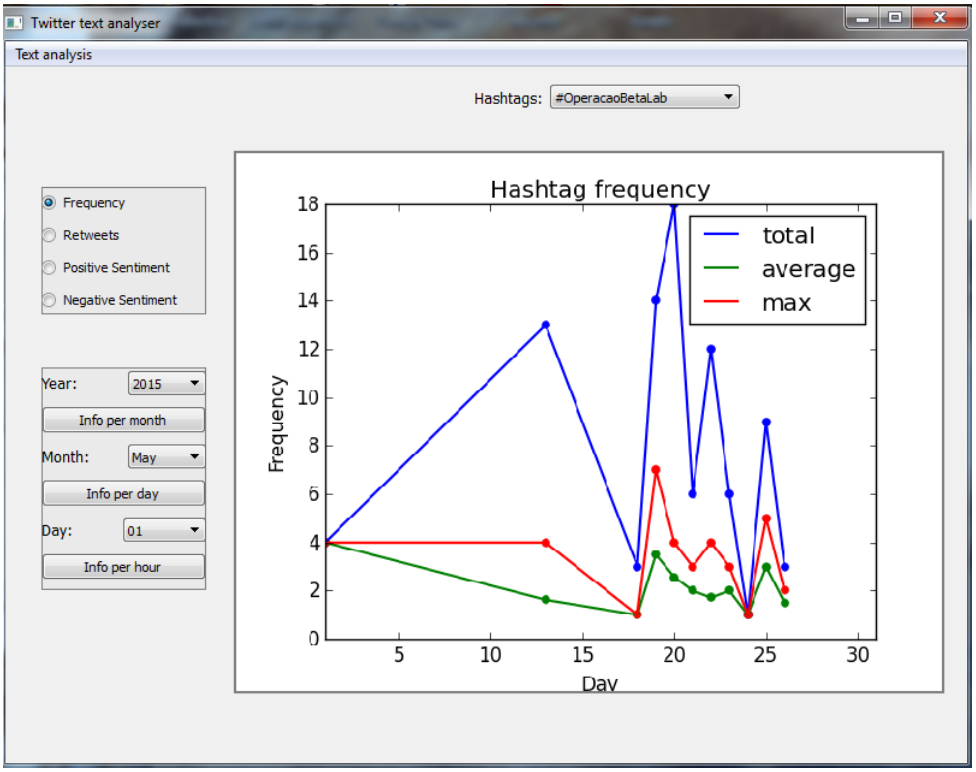

Figura 18: Análise de *Hashtags*.

Fonte: Autoria própria

de 5.000 *tweets* com sentimento positivo e 5.000 *tweets* com sentimento negativo, durante o dia de 28 de maio de 2015. Os *tweets* coletados como exemplos de treinamento para o classificador de sentimentos, são independentes dos *tweets* sobre o evento alvo. Para coletar os *tweets* positivos, foi utilizada a consulta ":)" e para coletar os *tweets* negativos, foi utilizada a consulta ":(". A API do Twitter recupera *tweets* que contenham os sinônimos dos *emoticons* utilizados nas consultas, apresentados no Quadro 5.

Foram treinados quatro classificadores diferentes para os dados de treinamento coletados. Da mesma maneira que na tarefa de clusterização, foram utilizadas diferentes combinações de *features*, algoritmos de classificação e métodos de vetorização para os exemplos de treinamento.

3.6 ANÁLISE DOS RESULTADOS

Para a clusterização dos *tweets*, foram obtidos os resultados apresentados no Quadro 6. No caso dos n-gramas, (2,2) significa apenas bi-gramas, (2,3) bi-gramas e tri-gramas e (3,3) apenas tri-gramas. A melhor clusterização dos dados foi obtida utilizando vetorização baseada

na frequência absoluta de tri-gramas. Uma possível explicação para esse resultado é que o uso de tri-gramas consegue capturar um maior contexto, facilitando a identificação de tópicos diferentes. Esses resultados foram obtidos realizando remoção de *stopwords* e normalização por *idf* quando aplicada a ponderação *tf-idf*.

Quadro 6: Resultados Obtidos na Clusterização.

#	vetorizador	n-gramas	*features*	coef. silhueta
1	*tf-idf*	(2,2)	117	0.42
2	*tf-idf*	(2,3)	177	0.50
3	*tf-idf*	(3,3)	60	0.80
4	frequência	(3,3)	60	0.93

Fonte: Autoria própria.

Para a classificação de sentimento, foi mantida a proporção de 50% de *tweets* positivos e 50% de *tweets* negativos para os dados de treinamento (80% dos dados) e teste (20% dos dados). Os resultados obtidos são apresentados no Quadro 7. No caso dos n-gramas, (1,3) significa o uso de uni-gramas, bi-gramas e tri-gramas. Para os testes realizados, foi utilizada a normalização por *idf* na ponderação *tf-idf* e remoção de *stopwords*. O uso de *stemming* e binarização das *features* obteve resultados piores em todos os cenários analisados. Como pode ser observado, os melhores resultados foram obtidos com o uso do SVM utilizando *tf-idf*. O parâmetro de regularização ajustado em 0.1 obteve uma melhora de 0.70 para 0.71 na medida F. Para valores muito pequenos do parâmetro de regularização a função de otimização busca por um hiperplano de maior margem, mesmo se o hiperplano classificar erroneamente alguns exemplos, o que pode explicar a ligeira melhora da medida F pois os dados de treinamento contém um alto nível de ruído. Os piores resultados foram obtidos com o uso do SVM utilizando a freqência absoluta como ponderação para as *features*. O *Naïve-Bayes* não sofreu nenhuma variação nos resultados com a utilização de diferentes ponderações.

Quadro 7: Resultados Obtidos na Classificação de Sentimento.

#	vetorizador	n-gramas	*features*	classificador	precisão	*recall*	medida F	C (SVM)
1	frequência	(1,3)	223	SVM	0.61	0.58	0.55	0.5
2	*tf-idf*	(1,3)	223	SVM	0.72	0.71	0.71	0.1
3	frequência	(1,3)	223	NB	0.70	0.70	0.70	-
4	*tf-idf*	(1,3)	223	NB	0.70	0.70	0.70	-

Fonte: Autoria própria.

Os quatro modelos de classificação apresentados no Quadro 7, foram postos a prova ao analisar o sentimento dos *tweets* e dos *clusters* de *tweets* do evento alvo, os resultados são apresentados no Quadro 8. A primeira e segunda coluna exibem a proporção de *tweets*

positivos/negativos, obtidos ao utilizar esses modelos. As outras colunas examinam a proporção de *tweets* positivos/negativos obtidas ao analisar o *cluster* 1 e o *cluster* 7. Os *clusters* foram obtidos, utilizando o modelo de número 4 no Quadro 6, que obteve o maior coeficiente de silhueta. Como pode ser observado, todos os classificadores obtiveram uma maior proporção de *tweets* positivos do que *tweets* negativos, tanto ao analisar o conjunto de todos os *tweets* quanto para os *cluster* analisados separadamente.

Quadro 8: Resultados Obtidos na Análise de Sentimento.

#	% pos	% neg	% pos *cluster* 1	% neg *cluster* 1	% pos *cluster* 7	% neg *cluster* 7
1	86.3	13.7	85.1	14.9	95.3	4.7
2	78.8	21.2	77.0	23.0	95.1	4.9
3	70.3	29.7	67.8	32.2	89.9	10.1
4	73.6	26.4	71.3	28.7	92.3	7.7

Fonte: Autoria própria.

4 CONCLUSÕES E TRABALHOS FUTUROS

Neste trabalho foi apresentada uma revisão de literatura sobre aprendizado de máquina e processamento de linguagem natural. Essas duas áreas do conhecimento foram empregadas no contexto da recuperação inteligente de informações. Foi apresentada uma ferramenta que auxilia o usuário no processo de coleta, pré-processamento e análise de dados provenientes do Twitter.

A ferramenta apresentada possui de maneira auto contida funcionalidades de coleta, pré-processamento, extração de *features* e treinamento de algoritmos de aprendizado, que são aplicados sobre dados textuais. Isso acaba facilitando o trabalho do usuário que pode contar com todas essas funcionalidades em uma única ferramenta, sem que seja necessário recorrer a diferentes pacotes de software e API's para realizar cada uma das tarefas separadamente.

No contexto da análise de eventos a ferramenta apresentada se mostra útil na tarefa de realizar a sumarização das informações relativas a um determinado evento alvo, com destaque para o processo de mineração de opiniões. Outra aplicação que poderia ser interessante para a ferramenta é a mineração de opiniões sobre um determinado produto, dessa forma uma empresa poderia ter uma noção da aceitação do seu produto no mercado, assim como uma sumarização das informações que estão sendo vinculadas ao produto. Dessa forma, uma ferramenta especializada como a apresentada nesse trabalho pode ser mais útil do que os mecanismos de busca mais genéricos que não realizam nenhum pré-processamento ou sumarização das informações desejadas.

Com relação ao treinamento do classificador de sentimento, melhorias podem ser realizadas. A adição de uma classe neutra em relação às classes positiva e negativa do problema, poderia aumentar ligeiramente a performance do modelo gerado, situação apresentada por Go et al. (2009). A utilização de dados de treinamento que possuam o mesmo tópico do evento alvo também poderia aumentar a performance dos classificadores, nesse caso há uma dificuldade maior em obter dados que possuam *emoticons* e o mesmo tópico do evento de interesse do que em obter dados genéricos, apenas com os *emoticons*.

Futuramente, outra informação de sumarização que poderia ser integrada à ferramenta é a distribuição geográfica de sentimento positivo/negativo. Dessa forma, o usuário teria a noção da polarização de opiniões por localização geográfica. Além disso, poderia também ser gerado um grafo sobre um mapa, relacionando usuários que realizaram postagens semelhantes, realizando uma filtragem por período de tempo, como ilustrado na análise de n-gramas e *hashtags*. Dessa forma, ficaria evidente como a distribuição geográfica de certos termos sofre alterações com o decorrer do tempo, sendo útil para analisar certos comportamentos de tendência em redes sociais.

Para o mecanismo de extração de *features*, futuramente poderia-se adicionar a funcionalidade de exibir os n-gramas selecionados pelo processo automático e possibilitar que o usuário inclua ou exclua certos n-gramas do conjunto de *features*, facilitando assim o processo de *tuning* dos algoritmos.

Uma alteração em relação a interface também poderia ser realizada futuramente. A ferramenta poderia ser portada para uma aplicação web, facilitando assim o acesso às informações geradas pela ferramenta, o usuário não teria a necessidade de instalar o Python e todas as dependências necessárias.

No cenário atual, com o grande crescimento das fontes de dados disponíveis para análise, sejam elas provenientes de mídias sociais ou não, trazem consigo um imenso horizonte de possibilidades para a aplicação de algoritmos de aprendizado de máquina e conceitos de PLN para realizar recuperação inteligente de informações. Esse tipo de aplicação se tornará cada vez mais comum e útil na resolução de problemas do mundo real e uma ferramenta que facilite esse processo de extração de informações como a apresentada neste trabalho irá se tornar uma peça importante no desenvolvimento de novas técnicas e conhecimentos que irão contribuir com o avanço da área.

REFERÊNCIAS

ASUR, S.; HUBERMAN, B. A. Predicting the future with social media. In: IEEE. **Web Intelligence and Intelligent Agent Technology (WI-IAT), 2010 IEEE/WIC/ACM International Conference on**. [S.l.], 2010. v. 1, p. 492–499.

BAEZA-YATES, R.; RIBEIRO-NETO, B. **Modern Information Retrieval: The Concepts and Technology behind Search**. [S.l.]: ACM Press Books, 2011.

BECKER, H.; NAAMAN, M.; GRAVANO, L. Beyond trending topics: Real-world event identification on twitter. **ICWSM**, v. 11, p. 438–441, 2011.

BIRD, S.; KLEIN, E.; LOPER, E. **Natural language processing with Python**. [S.l.]: "O'Reilly Media, Inc.", 2009.

BISHOP, C. M. et al. **Pattern recognition and machine learning**. [S.l.]: springer New York, 2006.

DOU, W. et al. Event detection in social media data. In: **IEEE VisWeek Workshop on Interactive Visual Text Analytics-Task Driven Analytics of Social Media Content**. [S.l.: s.n.], 2012. p. 971–980.

GO, A.; BHAYANI, R.; HUANG, L. Twitter sentiment classification using distant supervision. **CS224N Project Report, Stanford**, p. 1–12, 2009.

HAN, J.; KAMBER, M.; PEI, J. **Data mining: concepts and techniques**. [S.l.]: Elsevier, 2011.

HASTIE, T. et al. **The elements of statistical learning**. [S.l.]: Springer, 2009.

JAVA, A. et al. Why we twitter: understanding microblogging usage and communities. In: ACM. **Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 workshop on Web mining and social network analysis**. [S.l.], 2007. p. 56–65.

KAESTNER, C. **Notas de aula do curso sobre recuperação inteligente de informações**. 2014. Disponível em: <http://www.dainf.ct.utfpr.edu.br/ kaestner/Konstanz/iir.htm>. Acesso em: 28 de julho de 2014.

KIM, S.-B. et al. Some effective techniques for naive bayes text classification. **Knowledge and Data Engineering, IEEE Transactions on**, IEEE, v. 18, n. 11, p. 1457–1466, 2006.

KO, Y. A study of term weighting schemes using class information for text classification. In: ACM. **Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval**. [S.l.], 2012. p. 1029–1030.

MARCUS, A. et al. Twitinfo: aggregating and visualizing microblogs for event exploration. In: ACM. **Proceedings of the SIGCHI Conference on Human Factors in Computing Systems**. [S.l.], 2011. p. 227–236.

MITCHELL, T. M. Machine learning. 1997. **Burr Ridge, IL: McGraw Hill**, v. 45, 1997.

NG, A. **Notas de aula do curso sobre aprendizado de maquina**. 2014. Disponível em: <http://www.holehouse.org/mlclass/>. Acesso em: 28 de julho de 2014.

RAJARAMAN, A.; ULLMAN, J. D. **Mining of massive datasets**. [S.l.]: Cambridge University Press, 2011.

SAKAKI, T.; OKAZAKI, M.; MATSUO, Y. Earthquake shakes twitter users: real-time event detection by social sensors. In: ACM. **Proceedings of the 19th international conference on World wide web**. [S.l.], 2010. p. 851–860.

SAMUEL, A. L. Some studies in machine learning using the game of checkers. **IBM Journal of research and development**, IBM, v. 44, n. 1.2, p. 206–226, 2000.

TONKIN, E.; PFEIFFER, H. D.; TOURTE, G. Twitter, information sharing and the london riots? **Bulletin of the American Society for Information Science and Technology**, Wiley Online Library, v. 38, n. 2, p. 49–57, 2012.

ZHANG, P.; WANG, X.; LI, B. On predicting twitter trend: factors and models. In: ACM. **Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining**. [S.l.], 2013. p. 1427–1429.