UNIVERSIDADE DO ESTADO DE SANTA CATARINA – UDESC
CENTRO DE EDUCAÇÃO SUPERIOR DO ALTO VALE DO ITAJAI – CEAVI
DEPARTAMENTO DE SISTEMAS DE INFORMAÇÃO

FELIPE APPIO

FERRAMENTA PARA DETECTAR COMENTÁRIOS OFENSIVOS NO FACEBOOK

IBIRAMA – SC
2015

FELIPE APPIO

FERRAMENTA PARA DETECTAR COMENTÁRIOS OFENSIVOS NO FACEBOOK

Trabalho de Conclusão apresentado ao Curso de Sistemas de Informação, da Universidade do Estado de Santa Catarina, como requisito parcial para obtenção do grau de Bacharel em Sistemas de Informação.

Orientador: Carlos Alberto Barth
Coorientador: Ricardo Grunitzki

IBIRAMA – SC

2015

FELIPE APPIO

FERRAMENTA PARA DETECTAR COMENTÁRIOS OFENSIVOS NO FACEBOOK

Trabalho de Conclusão apresentado ao Curso de Sistemas de Informação, da Universidade do Estado de Santa Catarina, como requisito parcial para obtenção do grau de Bacharel em Sistemas de Informação.

Banca Examinadora

Orientador (a) _____
Prof. Msc Carlos Alberto Barth

Membros: _____
Prof. Msc Pablo Schoeffel

Membros: _____
Prof. Msc Gustavo Luis Pasqualini

A minha família e a todos os meus amigos que sempre me apoiaram e compreenderam minha ausência nessa trilha.

AGRADECIMENTOS

Aos meus pais, Antonio Leoberto e Nilva, que sempre me incentivaram e acreditaram em mim.

Aos meus irmãos, Alisson e Khadine, que além de terem contribuído durante essa jornada, sempre serviram como mentores para mim.

A Ricardo Grunitzki, meu coorientador, que sempre me ajudou, teve paciência, e também serviu como uma espécie de mentor para a minha pessoa.

A meu orientador, Carlos Alberto Barth, que assim como o Ricardo, sempre teve paciência e me guiou durante essa trajetória.

A Fernando dos Santos que foi uma peça fundamental para a idealização desse trabalho, e também sempre me ajudou.

A Geraldo Menegazzo Varela e família que foram as primeiras pessoas que me ajudaram quando eu vim para Ibirama.

A todos os professores do Centro de Educação Superior do Alto Vale do Itajaí (CEAVI), que acompanharam essa jornada de 4 anos transmitindo seus conhecimentos e experiências.

A Maciel Hogenn e Douglas Felipe Hoss, que estiveram ao meu lado sempre que eu precisei de ajuda.

A Darlan Borges, meu amigo do peito, que também contribuiu para a conclusão deste trabalho.

A todos os integrantes dos XumbaWambas, que de algum modo fazem parte do minha vida, e que compreenderam a minha ausência durante esse período.

A todas as pessoas que contribuíram de alguma forma para a conclusão deste trabalho.

Especialmente a Alice, por ter fornecido ajuda em algumas atividades que foram necessárias para a conclusão desse trabalho, bem como por ter me apoiado durante essa caminhada.

"Upset the established order and everything becomes chaos."

Unknown

RESUMO

Este trabalho apresenta a proposta de uma ferramenta *Web* que realizará mineração de opiniões - positivo, negativo ou neutro - na rede social *online Facebook*. Pode-se conceituar rede social como sendo um grupo de pessoas que interagem por intermédio de qualquer mídia de comunicação. Análise de sentimento, mineração de opiniões ou análise de subjetividade é um campo de estudo dentro de inteligência artificial que busca extrair informações subjetivas como opiniões, sentimentos, escritos em linguagem natural. Uma das técnicas bem difundidas em análise de sentimento é a técnica *Naive Bayes*, que é uma técnica de aprendizado de máquina supervisionado presente em uma biblioteca de mineração de texto, nomeada de *NLTK-Trainer*. A importância monitorar tais opiniões reside no fato de que organizações que fazem uso de redes socais necessitam monitorar seus produtos/serviços visando traçar estratégias com base nesses sentimentos. Para extrair dados do *Facebook* foi utilizado uma biblioteca nomeada de *Django-Facebook*, que é uma implementação da *Facebook Graph* API. Por fim, com a finalidade de avaliar a ferramenta proposta, foram coletados 90 comentários igualmente distribuídos em positivos, negativos ou neutros. Ao final observou-se que a ferramenta definiu as postagens em positiva, negativa ou neutra com grande percentual de acurácia, obtendo uma taxa média de acerto de 72% nesta classificação.

Palavras-chave: Análise de Sentimento. *Multinomial Naive Bayes*. *Facebook*.

LISTA DE ILUSTRAÇÕES

Figura 1 - Etapas de mineração de dados. 23
Figura 2 - Mecanismo de obtenção de *tokens* de acesso. 30
Figura 3 - Classificação em aprendizagem supervisionada. 32
Figura 4 - Estratificação do conjunto de dados. 34
Figura 5 - Treinamento e avaliação dos modelos. 35
Figura 6 - Diagramas de casos de uso. 46
Figura 7 - Diagrama de Pacotes 49
Figura 8 - Diagrama de classes da camada de acesso aos dados. 50
Figura 9 - Diagrama de classe das entidades *NBClassifierLoader* e *SpellingReplacer*. 51
Figura 10 - Diagrama de atividades do treinamento do classificador. 52
Figura 11 - Protótipo da página principal da ferramenta. 53
Figura 12 - Protótipo da página que apresenta as postagens textuais presentes na linha do tempo do usuário. 54
Figura 13 - Protótipo da página que apresenta os resultados da classificação gerada pela ferramenta. 55
Figura 14 - Estrutura do projeto 56
Figura 15 - Integração com o *Facebook* utilizando a *Javascript* SDK. 58
Figura 16 - Função *Javascript checkLoginState()*. 58
Figura 17 - Classe abstrata *DAO*. 59
Figura 18 - Classe abstrata *DAOFactory*. 59
Figura 19 - Classe concreta *FacebookDAOFactory*. 60
Figura 20 - Implementação do método *getFeed(self, firstIndex, lastIndex)* da entidade *GenericDAOFacebook*. 60
Figura 21 - Método *getFeedPage(self, content)* da entidade *GenericDAOFacebook*, que adiciona as publicações das páginas gerenciadas pelo usuário. 61
Figura 22 - Implementação do método *getCommentsFeed(self, firstIndex, lastIndex, id)* da entidade *GenericDAOFacebook*. 62
Figura 23 - Página principal da ferramenta. 70
Figura 24 - Caixa de *login* do *Facebook*. 71
Figura 25 - Caixa de diálogo que apresenta ao usuário as permissões necessárias para acessar os dados no *Facebook*. 72

Figura 26 - Página que apresenta as postagens textuais que estão na linha do tempo do usuário. 73
Figura 27 - Página que apresenta os comentários de uma determinada postagem. 74

LISTA DE QUADROS

Quadro 1 - Representação *bag-of-words* dos documentos $D1$ e $D2$..... 24
Quadro 2 - Matiz de confusão. 33
Quadro 3 - Conjunto de documentos com palavras e suas respectivas classes. 37
Quadro 4 - Quadro comparativo dos trabalhos correlatos..... 43
Quadro 5 - Regras de negócio do sistema. 45
Quadro 6 - Requisitos funcionais do sistema. 45
Quadro 7 - Requisitos não funcionais do sistema. 46
Quadro 8 - Cenário do UC001 (Conectar com o *Facebook*). 47
Quadro 9 - Cenário do UC002 (visualizar postagens). 47
Quadro 10 - Cenário do UC003 (classificar comentários). 48
Quadro 11 - Cenário do UC004 (filtrar comentários rotulados). 48
Quadro 12 - Métricas dos classificadores avaliados. 67
Quadro 13 - Métricas do *MultinomialNB* com e sem pré-processamento. 69
Quadro 14 - Resultados gerados pela ferramenta para as classes positivo, negativo e neutro..... 75

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Programming Interface*
BSD	*Berkeley Software Distribution*
EA	*Enterprise Architect*
HTTP	*Hyper Text Transfer Protocol*
IBM	*International Business Machines*
MAXENT	*Maximum Entropy*
NLP	*Natural Language Processing*
NLTK	*Natural Language Toolkit*
SDK	*Software Development Toolkit*
SVM	*Support Vector Machines*
UML	*Unified Modeling Language*

SUMÁRIO

1 INTRODUÇÃO 13
1.1 PROBLEMA 14
1.2 OBJETIVOS 15
1.2.1 Objetivo geral 15
1.2.2 Objetivos específicos 15
1.3 JUSTIFICATIVA 16
1.4 HIPÓTESE 17
1.5 METODOLOGIA 17
1.5.1 Análise e Desenvolvimento da Ferramenta 18
1.6 ESTRUTURA DO TRABALHO 19
2 FUNDAMENTAÇÃO TEÓRICA 21
2.1 WEB MINING 21
2.2 MINERAÇÃO DE TEXTO E ANÁLISE DE SENTIMENTO 21
2.2.1 Coleta de Dados 23
2.2.2 Pré-Processamento 23
2.2.2.1 Extração de Características 24
2.2.3 Mineração e Interpretação 27
2.3 FACEBOOK GRAPH API 28
2.4 DJANGO-FACEBOOK 30
2.5 NLTK-TRAINER E APRENDIZAGEM DE MÁQUINA 31
2.5.1 Treinamento e avaliação dos classificadores 32
2.5.2 Naive Bayes 35
2.5.2.1 Multinomial Naive Bayes e Classificação Textual 37
3 TRABALHOS CORRELATOS 39
3.1 PROTÓTIPO PARA MINERAÇÃO DE SENTIMENTOS EM REDES SOCIAIS: ESTUDOS DE CASOS SELECIONADOS USANDO O TWITTER 39
3.2 REAL-TIME SENTIMENT ANALYSIS IN SOCIAL MEDIA STREAMS: THE 2013 CONFEDERATION CUP CASE 40
3.3 TWITTER SENTIMENT CLASSIFICATION USING DISTANT SUPERVISION 41

3.4 COMPARATIVO DOS TRABALHOS CORRELATOS 41
4 DESENVOLVIMENTO 44
4.1 ESPECIFICAÇÃO 44
4.1.1 Contextualização 44
4.1.2 Diagramas de Caso de Uso 46
4.1.3 Diagramas de Classe 48
4.1.4 Diagrama de Atividades 51
4.1.5 Prototipação 53
4.2 IMPLEMENTAÇÃO 55
4.2.1 Tecnologias Utilizadas 55
4.2.2 Estrutura do Projeto 56
4.2.3 Integração com o Facebook 57
4.2.4 Coleta dos dados 58
4.2.5 Classificação 63
4.2.5.1 Base de dados utilizada durante o treinamento 63
4.2.5.2 Treinamento/Escolha 64
4.3 OPERACIONALIDADE 69
4.4 RESULTADOS EXPERIMENTAIS 75
5 CONCLUSÃO 78
5.1 CONTRIBUIÇÕES 78
5.2 PERSPECTIVAS FUTURAS 78
REFERÊNCIAS 97
Apêndice x

1 INTRODUÇÃO

Rede social *online* pode ser conceituada como um grupo de pessoas que interagem por meio de qualquer mídia de comunicação. Pode-se definir, também, como sendo um sistema *Web* que fornece a possibilidade de construir perfis públicos; organizar uma lista de usuários com o qual é possível compartilhar conexões; visualizar e percorrer suas listas de conexões (BENEVENUTO; ALMEIDA e SILVA, 2011). Com base nessas definições há várias redes sociais *online* disponíveis como, por exemplo, o *Facebook*[1], *Twitter* [2], *Google* +[3].

Ao final de 2013 a rede social *online Facebook* atingiu 1.23 bilhões de usuários (KISS, 2014), o que implica na geração massiva de conteúdos nesta rede. No *Facebook* pode-se encontrar páginas de artistas como Luciano Huck[4], de empresas como Coca-Cola[5], e de outras figuras públicas como Barack Obama[6]. Tais páginas podem ter muitos seguidores, como por exemplo a página da Coca-Cola, que no dia 05 de abril de 2015 possuía mais de 89 milhões de seguidores. De modo geral, quanto maior for o número de seguidores de uma página, maior é a repercussão dos comentários textuais publicados na mesma. Isso pode ser um problema se o conteúdo publicado contiver informações negativas, pois conforme Becker e Tumitan (2013), as organizações baseiam suas estratégias de negócio com base na opinião dos clientes sobre um determinado produto ou serviço.

Desse modo, em páginas com milhares de seguidores, torna-se difícil monitorar os comentários textuais compartilhados nas postagens dessas páginas, principalmente comentários negativos. Visto que as organizações ou indivíduos podem monitorar a opinião dos seguidores para algum objetivo específico (BECKER; TUMITAN, 2013), torna-se evidente que comentários com semântica negativa (ruim) poderiam ser tratados de forma prioritária, sendo necessário haver algum modo de filtrá-los.

O processo automatizado de extrair informações textuais com um propósito específico de textos não estruturados é conhecido como mineração de textos – que vem

[1] Disponível em: <https://www.facebook.com/?_rdr>.
[2] Disponível em: <https://twitter.com/?lang=pt>.
[3] Disponível em: <https://plus.google.com/?hl=pt_BR>.
[4] Disponível em: < https://www.facebook.com/LucianoHuck?fref=ts>.
[5] Disponível em: < https://www.facebook.com/cocacolabr?fref=ts>.
[6] Disponível em: < https://www.facebook.com/barackobama?fref=ts>.

do inglês *text mining* (WITTEN et al. 1999). Ainda dentro de mineração de texto, há uma área chamada de análise de sentimento que objetiva classificar opiniões expressas em textos (COLLETA et al, 2014) - geralmente em positivo (bom), negativo (ruim) ou neutro (objetivo) (BECKER; TUMITAN, 2013). Essa classificação pode ocorrer por intermédio de duas abordagens: utilizando algoritmos baseados em aprendizagem de máquina, ou através de ontologias (IRFAN, 2015). Este trabalho realiza a classificação de textos por intermédio de um classificador baseado em aprendizado de máquina.

Aprendizagem de máquina - termo que vem do inglês, *Machine learning* (ML) - é uma área de inteligência artificial que estuda e desenvolve algoritmos que aprendem a partir de observações. Dentro de ML há aprendizagem supervisionada – que será objeto desse trabalho, pois será utilizado a técnica de aprendizagem estatística *Multinomial Naive Bayes* para classificação textual (RUSSEL; NORVIG, 2009).

Este trabalho propõe uma ferramenta cuja finalidade é apontar postagens textuais (comentários nas publicações) ofensivas, publicadas na linha do tempo de um perfil no *Facebook* (usuário da ferramenta). Futuramente essa ferramenta poderá ser integrada com outras redes sociais. A ferramenta classificará sentenças escritas em português brasileiro em três classes: positivo; negativo ou neutro. Posteriormente será efetuado um experimento com a finalidade de validar a ferramenta.

Este trabalho se justifica pelo fato de colaborar com usuários que possuem milhares de publicações (advindas de outros usuários) no seu perfil da rede social *Facebook*, que não conseguem monitorar todas as postagens textuais em virtude do grande volume.

1.1 PROBLEMA

Nos últimos anos o *Facebook* vem crescendo, e em 2013 atingiu mais de 1 bilhão de usuários ativos (KISS, 2014). Em consequência disso, uma vasta quantidade de comentários tem sido publicada por toda a rede. Páginas de figuras públicas ou de grandes empresas que tipicamente recebem grande volume de comentários. Isso torna difícil para que um usuário acompanhe todas as publicações em sua página ou perfil, pois constantemente seus seguidores estão publicando novos comentários na rede.

Conforme Becker e Tumitan (2013), as organizações (empresas, partidos políticos, etc.) têm criado páginas em redes sociais como o *Facebook* com o intuito de promover seus produtos e/ou serviços, bem como observar o que os clientes pensam a respeito disso, pois suas estratégias devem ser tomadas com base nisso. O problema é que em páginas que possuem muitos comentários de terceiros, pode ser humanamente inviável o acompanhamento desses comentários, dependendo do volume de dados a ser analisado.

Nesse sentido, uma ferramenta que possa detectar comentários ofensivos (negativos) ou favoráveis (positivos) em uma página ou perfil de um usuário que possui muitas publicações no *Facebook*, pode ser utilizado por essas organizações, visando melhorar o monitoramento da opinião de seus clientes.

1.2 OBJETIVOS

Esta Seção apresenta os objetivos que se deseja alcançar ao final do desenvolvimento deste trabalho.

1.2.1 Objetivo geral

Desenvolver uma ferramenta que, por intermédio de técnicas de aprendizagem de máquina, possa atribuir significado (negativo, positivo ou neutro) a postagens feitas na rede social *Facebook* em língua portuguesa do Brasil.

1.2.2 Objetivos específicos

a) Identificar comentários ofensivos que fazem uso de adjetivos e verbos negativos cujo objeto é um único usuário, sem levar em consideração o contexto (publicação);
b) Apontar postagem de comentários ofensivos que não fazem uso de adjetivos negativos, porém, ainda assim, expressam conteúdo pejorativo a um usuário ou de organizações;

c) Validar a ferramenta por intermédio de dados reais capturados da rede social *Facebook* e rotulados empiricamente em positivo (bom), negativo (ruim) ou neutro (objetivo).

1.3 JUSTIFICATIVA

A explosão das redes sociais tornou possível para indivíduos e organizações publicarem opiniões, ideias sobre produtos e serviços por intermédio de comentários nestas páginas. Consequentemente, surgiram oportunidades de monitoramento de opiniões tanto de forma qualitativa, quanto quantitativa sobre um determinado produto ou serviço. A importância de se medir opiniões está no fato de que o planejamento estratégico de organizações (empresas, partidos políticos, etc.) levam em conta este tipo de informação para traçar objetivos e metas (BECKER; TUMITAN, 2013).

Por conta dessa crescente explosão, organizações ou indivíduos vem criando páginas e perfis em redes sociais como o *Facebook, Twitter* (BECKER e TUMITAN, 2013). Segundo Statista (2015), o *Facebook* é a maior rede social *online*, e por conta disso, muitas empresas, artistas e políticos, tem criado páginas e perfis nessa rede com o intuito de comunicar-se com outras pessoas e divulgar seus produtos e serviços. Ocorre que, devido ao grande número de seguidores e consequentemente de publicações nestas páginas, torna-se difícil o monitoramento de comentários.

Desse modo, uma ferramenta que auxiliasse de forma automática o acompanhamento de comentários em suas publicações, classificando-os em positivo (bom), negativo (ruim) ou neutro (objetivo) e fornecendo-as aos usuários, seria de extrema valia. Essas informações poderiam ser avaliadas de acordo com as prioridades estabelecidas pelos usuários.

1.4 HIPÓTESE

Por intermédio das API - *Application Interface Programming*[7] - disponibilizada pelo *Facebook*, torna-se possível extrair dados dessa rede social. Com esses dados, podem-se aplicar técnicas de aprendizagem de máquina e efetuar classificação textual, rotulando-os em positivo (bom), negativo (ruim), neutro (objetivo).

1.5 METODOLOGIA

No que concerne à abordagem para a coleta de dados, optou-se por utilizar pesquisa qualitativa e quantitativa, conforme segue:

a) Qualitativa porque se faz necessário realizar um processo de sequenciamento de atividades de coleta e categorização dos dados, bem como fazer as interpretações sobre os mesmos. Embora o tamanho da amostragem dos dados, e os pressupostos teóricos que embasaram a investigação influenciem na abordagem qualitativa, seu uso é necessário no contexto desse trabalho (GIL, 2010).
b) Quantitativa pelo fato de haver a necessidade de quantificar os dados classificados assertiva e erradamente, e posteriormente medir os respectivos percentuais de erros e acertos, conforme argumenta Gil (2010).

Para realizar o desenvolvimento deste trabalho se faz necessário uma revisão bibliográfica sobre mineração de texto; análise de sentimento; aprendizagem de máquina e trabalhos correlatos. O levantamento bibliográfico foi feito por intermédio de uma pesquisa exploratória com o intuito de tornar o problema mais explícito (GIL, 2010).

De modo a entender o funcionamento de um classificador de aprendizado de máquina, e identificar o mais adequado ao contexto deste trabalho, optou-se por utilizar uma pesquisa exploratória, pois conforme postula Gil (2010), a pesquisa exploratória

[7] Interface de programação de aplicação (tradução do autor).

visa proporcionar um melhor entendimento ao autor sobre o tema a ser pesquisado visando tornar o conhecimento mais explícito.

A pesquisa experimental consiste em tomar um objeto de estudo, analisar e classificar as variáveis que, direta ou indiretamente influenciam seu comportamento, definir formas para controlar e observar os efeitos causados pelas variáveis sobre objeto de estudo (GIL, 2010).

Nesse sentido, será adotada a pesquisa experimental, pois as variáveis que influenciam diretamente no comportamento da ferramenta, são: captura de textos advindos da rede social *Facebook*, pré-processamento dos textos; inferências sobre os textos pré-processados. Caso essa metodologia não seja válida, novas abordagens serão utilizadas.

Com a finalidade de testar a hipótese, optou-se por utilizar uma pesquisa descritiva, pois esta é responsável por descrever as características de um fenômeno, proporcionando, assim, uma nova visão do problema (GIL, 2010).

1.5.1 Análise e Desenvolvimento da Ferramenta

Nesta Seção será descrita a metodologia adotada para análise e desenvolvimento deste trabalho, desde a especificação até a implementação.

Primeiramente é necessário fazer um estudo exploratório sobre os principais algoritmos de classificação utilizados em análise de sentimento, com o intuito de observar qual classificador apresenta melhores resultados em trabalhos semelhantes a este. O próximo passo será definir uma biblioteca de aprendizagem de máquina que tenha essas técnicas implementadas, para que então seja possível treinar e analisar os classificadores e, consequentemente, definir qual é mais adequado ao contexto desse trabalho.

Posteriormente, se faz necessário entender o funcionamento do *Facebook Graph* API (FACEBOOK, 2015c), para que seja possível obter os dados (publicações) na rede. Ainda nesta etapa, é necessário definir uma implementação da *Graph* API.

Após as etapas anteriores serem completadas, será realizada a especificação da aplicação por intermédio da *Unified Modelling Language* (UML). Será elaborado o

diagrama de caso de uso, de atividade, de classe e de sequência no decorrer deste trabalho. A ferramenta escolhida para dar suporte ao processo de modelagem será o *Enterprise Architect* [8](EA).

Na etapa de desenvolvimento será utilizado o ambiente de desenvolvimento integrado (em inglês: *integrated development environment* - IDE -) *Eclipse lunar* (versão 4.4)[9], utilizando como sistema operacional o *Linux Ubuntu* 14.04 versão de 64 *bits*[10].

Visando realizar a validação dos objetivos desse trabalho, será feito um conjunto de testes para verificar os resultados das classificações geradas pela ferramenta. Para tal, a ferramenta será executada na versão mais atual do *Google Chrome*[11] utilizando como alvo o perfil do *Facebook* do Juvenal Antena[12].

Os resultados dos rótulos dos comentários gerados pela ferramenta serão avaliados tanto de forma quantitativa, quanto qualitativamente.

1.6 ESTRUTURA DO TRABALHO

O trabalho está dividido em 5 capítulos. O capítulo (2) aborda o embasamento teórico necessários para o desenvolvimento desse trabalho. Em seguida a Seção (2.1) fundamenta os conceitos a respeito de *Web Mining*. A Seção (2.2) explana os preceitos básicos de mineração de texto e análise de sentimentos. O *Facebook Graph* API é abordado na Seção (2.3). Na Seção (2.4) são abordadas questões relacionadas ao *Django framework* e ao *Django-Facebook*. Logo em seguida apresenta-se o embasamento teórico pertinente a aprendizagem de máquina (2.5) e a biblioteca utilizada.

Feito isso, os trabalhos correlatos são apresentados, no capítulo (3). A especificação e implementação da ferramenta são abordados no capítulo (4), mais

[8] Disponível em: < http://www.sparxsystems.com.au/>.
[9] Disponível em: < http://www.eclipse.org/downloads/download.php?file=/technology/epp/downloads/release/luna/R/eclipse-standard-luna-R-linux-gtk-x86_64.tar.gz>.
[10] Disponível em: < http://releases.ubuntu.com/14.04/>.
[11] Disponível em: < http://www.google.com/chrome/>.
[12] Disponível em: < https://www.facebook.com/profile.php?id=100009652808830>.

precisamente nas seções (4.1) e (4.2) respectivamente. Por último, no capítulo (0) apresentam-se as conclusões e os trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Nesse capítulo será abordada a fundamentação teórica necessária para o desenvolvimento deste trabalho.

2.1 WEB MINING

De acordo com Cooley; Mobasher e Srivastava (1997), *Web mining* pode ser definido como sendo a análise e descoberta de informações advindas da *World Wide Web*. Conforme Carrilha Júnior (2007) há três abordagens para minerar dados no ambiente *Web*:

a) *Web Content Mining* – ou mineração do conteúdo *Web* - preocupa-se em analisar o conteúdo dentro das páginas HTML, isto é, textos; imagens; etc.
b) *Web Usage Mining* – preocupa-se em identificar padrões de acesso dos usuários, por exemplo, do ponto de vista do *link*, quais partes dos sites precisam ser alteradas com base nos acessos dos usuários. Por último,
c) *Web Structure Mining* – responsável por estudar os relacionamentos entre os *hiperlinks*. Nesse sentido, avalia-se a quantidade de ligações que uma página possui com outra, podendo-se inferir qual página possui mais referências, isto é, as mais divulgadas.

Esse trabalho encontra-se dentro da subárea *Web Content Mining*, visto que o conteúdo a ser processado está presente no ambiente *Web*, mais precisamente na rede social *Facebook*.

2.2 MINERAÇÃO DE TEXTO E ANÁLISE DE SENTIMENTO

Conforme Witten et al. (1999), mineração de textos - *text mining* - é o processo de analisar textos não estruturados de forma automática, procurando por padrões, de modo a extrair informações com um propósito específico. Essas informações podem ser opiniões ou emoções expressas em textos. Para tal, há uma

subárea nomeada de análise de sentimentos, também conhecida como mineração de opinião ou análise de subjetividade que, busca justamente classificá-las - geralmente em positivo, negativo ou neutro (BECKER; TUMITAN, 2013).

Segundo Hu; Liu (2004), Turney; Pang e Lee (2002), Melville; Gryc e Lawrence (2009), He et al. (2008), Balahur et al. (2010), pode-se extrair sentimentos dos mais diversos textos (seja escrito de forma formal ou informal), como por exemplo, textos de críticas de filmes, produtos, automóveis, noticiários e blogues.

Há três maneiras de fazer isso, utilizando algoritmos de aprendizagem de máquina (supervisionada ou não), ou técnicas de *Natural Language* Processing[13] (NLP) (TABOADA et al. 2011). Além disso, é possível combinar técnicas de pré-processamento (técnicas de NLP) com técnicas de aprendizagem de máquina.

A área de NLP trata da extração de informações como “por que”, “quando”, “como”, "o que" expressa em textos. Suas técnicas fazem uso de conceitos de linguística, gramática e pragmática (KAO; POTEET, 2007).

Conforme Aranha (2007), as etapas de análise de sentimento acontecem da seguinte maneira:

a) A coleta de dados textuais é a primeira etapa da mineração de textos, cujos dados são obtidos de diversas fontes, como por exemplo redes sociais;
b) Em seguida vem a etapa de pré-processamento – técnicas de processamento de linguagem natural - onde os dados sofrem tratamento em sua estrutura e organização;
c) O desenvolvimento de cálculos, inferências e extração de conhecimento sobre esses dados é chamado de mineração, que vem logo após a indexação;
d) A última etapa é a análise dos dados, isto é, a interpretação dos dados. Pode ser feita com o uso de medidas quantitativas ou por um especialista no assunto.

Essas fases de mineração de dados são ilustradas na Figura 1 e explanadas nas seções a seguir.

[13] Termo que vem do inglês e pode ser definido como processamento de linguagem natural.

Figura 1 - Etapas de mineração de dados.

Fonte: Produção própria do autor.

2.2.1 Coleta de Dados

Conforme Carrilha Júnior (2007), por coleta entende-se o ato de buscar e recuperar dados com o intuito de formar uma base textual da qual se pretende extrair informações. Há vários meios de se obter esses dados textuais. Arquivos encontrados em discos rígidos; dados advindos da *Web* ou dados advindos dos mais diversos bancos de dados.

O escopo desse trabalho restringe-se à coleta de dados que tem sua origem na *Web*; mais precisamente dados da rede social *online Facebook* por intermédio da biblioteca *Django-Facebook*[14] - que é explanada na Seção (2.4) - segundo os mecanismos definidos pela *Facebook Graph* API (vide Seção 2.3).

2.2.2 Pré-Processamento

Conforme Carrilha Júnior (2007), o pré-processamento é a etapa que se segue logo após a coleta dos dados. O pré-processamento é necessário para realizar a mineração de dados, pois ele aumenta a qualidade dos dados em questão. Embora para outros autores, como para Santos e Ladeira (2014), o pré-processamento nem sempre tem uma melhora significativa dentro de mineração de textos.

Para Irfan et al. (2015) há dois métodos básicos de pré-processamento: a seleção de características e a extração de características. Esses métodos serão detalhados na seção a seguir.

[14] Uma implementação do *Facebook Graph* API para *Python* compatível com o *Django* (SCHELLENBACH, 2015).

2.2.2.1 Extração de Características

Conforme Scikit-learn (2014a), o processo de extração de características - *feature extraction* (FE) - é o processo de transformar dados arbitrários, como imagens ou textos, em dados numéricos que possam ser utilizadas pelos algoritmos de aprendizagem de máquina.

O modelo *bag-of-words*[15] é um dos modelos mais simples utilizados em extração de características textuais, pois não é necessário saber a ordem de uma determinada palavra no texto, basta saber que ela está presente e a quantidade de vezes que ela aparece (PERKINS, 2014). Para Raschka (2014), o processo de extrair um vocabulário V (todas as palavras que aparecem, desconsiderando as repetições) de 1 ou mais documentos de textos D_i e contar suas respectivas frequências é chamado de vetorização. Desse modo, o modelo *bag-of-words* de um documento $D_1 =$ "$Cada\ estado\ tem\ suas\ próprias\ leis$" e outro documento $D_2 =$ "$Cada\ país\ tem\ suas\ próprias\ culturas$" será mapeado conforme Quadro 1:

Quadro 1 - Representação *bag-of-words* dos documentos D_1 e D_2

	Cada	estado	tem	suas	próprias	leis	país	culturas
x_{D1}	1	1	1	1	1	1	0	0
x_{D2}	1	0	1	1	1	0	1	1
$\sum$	2	1	2	2	2	1	1	1

Fonte: Produção própria do autor.

De acordo com Raschka (2014), o modelo *bag-of-words* pode conter valores binários - representando se a palavra está presente no documento ou não - ou as respectivas frequências das palavras. O que determina esses valores é o classificador utilizado. Em virtude desse trabalho utilizar o *Multinomial Naive Bayes* (vide Seção 2.5.2), será utilizado o modelo *bag-of-words* contendo as frequências das palavras.

Segundo Irfan et al. (2015), é possível categorizar a extração de características em análise morfológica, análise sintática e análise semântica. O método de análise morfológica consiste em *tokenization*[16]e redução do léxico (CARRILHA JÚNIOR,

[15] Termo que vem do inglês e pode ser traduzido para saco de palavras.

[16] Não há tradução para português.

2007). Em virtude desse trabalho utilizar somente a técnica de análise morfológica (vide Seção 2.5) os outros métodos não serão abordados.

Tokenization é a primeira etapa a ser realizada durante o pré-processamento, a sua finalidade é extrair as unidades mínimas de um texto. Um *token*[17] geralmente é uma palavra que pode estar relacionado com outras palavras, caracteres ou símbolos (CARRILHA JÚNIOR, 2007). Desse modo, após se aplicar *tokenization* na sentença: "Chorão foi um dos maiores cantores brasileiros.", obter-se-ia como resultado: [Chorão] [foi] [um] [dos] [maiores] [cantores] [brasileiros] [.], neste caso cada *token* seria representado por um elemento entre colchetes.

Apesar de esse processo parecer relativamente simples, é necessário ter cuidado ao realizá-lo, pois alguns termos, quando separados mudam completamente o sentido da frase. O ponto final, por exemplo, além de finalizar a sentença, pode ser utilizado para abreviar números. Para resolver esse problema podem-se identificar símbolos da internet, abreviações, etc. e combiná-las em um único *token* de modo a conservar o sentido das palavras (CARRILHA JÚNIOR, 2007). A palavra "guarda-roupa", por exemplo, poderia ser agrupada em um único *token*: [guarda-roupa].

Feito a *tokenization* é necessário fazer a redução do léxico, pois conforme foi citado em Carrilha Júnior (2007), cada palavra vira um *token* que é mapeado para uma dimensão, gerando assim um vasto número de dimensões, dependendo do texto. Para amenizar esse problema, então, deve-se diminuir a quantidade de palavras em um texto, atentando para não prejudicar a semântica das sentenças (CARRILHA JÚNIOR, 2007).

Há várias abordagens para amenizar esse problema; contudo, esse trabalho abordará somente três delas: remoção de *stopwords*[18]; normalização; seleção de características (CARRILHA JÚNIOR, 2007).

Conforme Carrilha Júnior (2007), a etapa de remoção de *stopwords* consiste em remover artigos, preposições, conjunções, pontuação e pronomes de uma língua, pois geralmente não possuem um significado relevante para o contexto de mineração de textos. Essas palavras são classificadas como *stopwords* – termo que vem do inglês e não possui tradução para português.

[17] Termo sem tradução para português.
[18] Sem tradução para português.

Um conjunto de *stopwords* compõe uma *stoplist*[19], isto é, uma lista com os *tokens* pouco relevantes. As *stopslits* podem ser definidas por um especialista no assunto, ou por intermédio de alguma técnica de NLP, como por exemplo a técnica de medida de entropia do termo, ou a medida de divergência (SAIF, 2014). Conforme Saif (2014), não há diferença significativa entre o a remoção de *stoplists* estáticas ou dinâmicas. Portanto esse trabalho utilizará a abordagem estática, pois conforme Saif (2014), é computacionalmente menos custoso utilizar *stoplists* estáticas. Em virtude disso, só será explanado o mecanismo de remoção de *stoplist* estática (Seção 4.2.5.2).

De acordo com Carrilha Júnior (2007), normalização é a etapa de identificação e agrupamento de palavras que possuem um relacionamento semântico, como por exemplo, os sinônimos. A normalização pode acontecer pelos processos de:

a) *Stemming* – é o processo de converter palavras para sua forma original. Desse modo, termos no plural, verbos em diferentes tempos verbais tornam-se palavras no infinitivo. Segundo Raschka (2014), esse processo pode gerar palavras que não existem.
b) *Lemmatization* – é o processo de converter palavras que foram derivadas de outras palavras em sua respectiva palavra de origem, respeitando a gramática. Pode-se citar como exemplo "as palavras música, músicas e musical compartilham a mesma palavra de origem, música." (SANTOS, 2010, p. 30).

Durante a etapa de normalização, pode-se utilizar tanto o método *lemmatization*, quanto *stemming*, ou ambos. *Lemmatization* é computacionalmente mais custoso do que *stemming*, embora não gere um impacto significativo (em relação a *stemming*) no desempenho de classificadores textuais. Além disso, a combinação dos dois métodos não tem grande impacto no desempenho de classificação textual (RASCHKA, 2014). Portanto, durante a etapa de pré-processamento foi escolhido utilizar somente *stemming* neste trabalho.

Segundo Perkins (2014) uma estratégia utilizada para melhorar o processo de extração de características é utilizar *collocations*[20]. *Collocations* são duas ou mais

[19] Sem tradução para português.
[20] Pode ser traduzido para português como colocações (tradução do autor).

palavras que tendem a aparecer juntas, como por exemplo, 'Estados Unidos' (PERKINS, 2014). O termo *ngrams*[21] descreve uma colocação de ordem n.

Desse modo, utilizando *bigrams* (*ngram* de ordem 2), a sentença 'Estados Unidos' após o processo de *tokenization* torna-se ['Estados Unidos'] e não ['Estados'] ['Unidos'] (RASCHKA, 2014). Para Pang et al. (2002) e Dave; Lawrence e Pennock (2003), em análise de sentimento *unigrams*, *bigrams* e *trigrams* aumentam o desempenho do classificador. Portanto, esse trabalho utiliza somente *ngram* de rodem 1 até 3.

Para Brown (1992), *ngrams* são gerado com base em algoritmos de modelagem de linguagem probabilísticos. Em virtude desse trabalho considerar somente *ngrams* de ordem 1 até 3 - conforme citado no parágrafo acima - , não será entrado em detalhes em relação a geração de *ngrams*. Mais informações sobre *ngrams* podem ser obtidas em Brown (1992).

2.2.3 Mineração e Interpretação

Esta etapa é responsável por extrair novos conhecimentos partir de técnicas de aprendizagem de máquina (ML). Dentro de mineração de dados há a mineração de textos, que tem por objetivo extrair conhecimentos oriundos de bases textuais gravados em linguagem natural. Inicialmente deve-se decidir que tipo de informação deseja-se extrair da massa textual. Algoritmos de classificação e agrupamento podem ser aplicados nessa etapa (SOARES, 2008). Como o escopo desse trabalho restringe-se a classificação, não será abordada a questão de agrupamento.

A interpretação, análise ou também nomeada de pós-processamento diz respeito ao tratamento do conhecimento obtido na mineração. Pode ser realizada por um especialista no assunto, bem como por métodos quantitativos e qualitativos (SOARES, 2008).

Essa etapa é responsável por efetivamente visualizar e interpretar os dados obtidos até a mineração. Para realizar a interpretação dos dados, há várias métricas na

[21] Sem tradução para português.

literatura (SOARES, 2008). Contudo, para realizar o desenvolvimento deste trabalho será tomada como critério a acurácia (que será explanada na 2.5.1).

2.3 FACEBOOK GRAPH API

O *Facebook* fornece uma API baseada em *Hyper Text Transfer Protocol*[22] (HTTP) chamada de *Facebook Graph* API, cujo objetivo é fornecer mecanismos para ler e escrever dados na linha do tempo de usuários da rede social *Facebook* (FACEBOOK, 2015a). Em função disso, qualquer linguagem de programação que tenha uma biblioteca HTTP pode manipular os dados através da *Graph* API. Segundo Facebook (2015a), há várias bibliotecas para a *Graph* API. Entretanto, esse trabalho utiliza a biblioteca *Django-Facebook* que será explanada na Seção (2.4).

Independentemente de qual implementação da *Graph* API for utilizada, só há duas maneiras de manipular dados do *Facebook*. Uma utilizada para ler, e outra para escrever. O mecanismo de leitura dos dados acontece por intermédio do método GET - utilizado em requisições simples, geralmente para leitura de dados - do protocolo HTTP. Já o segundo mecanismo, de escrita de dados, acontece por intermédio do método POST - geralmente utilizado em requisições mais complexas que necessitam enviar dados - do protocolo HTTP (FACEBOOK, 2015b).

Segundo Facebook (2015b), todas as requisições efetuadas para *Facebook* devem seguir o conceito de grafo social, que representa as informações da seguinte forma:

a) Os nodos representam um usuário, uma foto, comentário, uma página;
b) As bordas configuram as interconexões entre eles, como por exemplo, os comentários de uma foto, as fotos de uma página, etc;
c) Os campos representam informações sobre os nodos, como por exemplo o nome de um usuário, a data de nascimento, etc.

Conforme foi citado acima, o protocolo HTTP funciona no modelo cliente-servidor por intermédio das requisições e repostas, e portanto, é necessário que haja um

[22] Um protocolo projetado para permitir comunicação entre cliente e servidor por intermédio de requisições e respostas entre o cliente e o servidor (W3C, 2015).

endereço (servidor) de destino para as requisições. Segundo Facebook (2015b), quase todas as requisições (exceto as em que o nodo do tipo vídeo) são feitas para o servidor de endereço: *graph.facebook.com*. As demais, do tipo vídeo são feitas para o endereço *graph-video.com*.

De acordo com Facebok (2015b), algumas requisições podem ser efetuadas sem um *token* de acesso. Contudo, grande parte das requisições precisam de um *token* de acesso[23]. Além disso, as requisições efetuadas pela ferramenta proposta nesse trabalho necessitam de permissões adicionais. Segundo Facebook (2015c), para obter *token* de acesso com permissões adicionais se faz necessário enviar o aplicativo ao *Facebook* para que seja realizado uma inspeção no mesmo. Além disso, é necessário ter um aplicativo cadastrado no *Facebook*, pois é necessário ter um código de acesso (FACEBOOK, 2015c). Portanto, a ferramenta proposta será enviada para inspeção realizada pelo *Facebook*.

Um *token* de acesso é um conjunto de dados alfanuméricos que é obtido quando o *Facebook* confirma a conexão do usuário (realizada a partir da ferramenta). O *token* de acesso possui informações como permissões que o aplicativo[24] possui para acessar informações do usuário, tempo de vida (validade) do *token* de acesso e informações sobre o aplicativo (código identificador do aplicativo).

Há quatro tipos de *tokens* de acesso, o *token* de acesso do usuário, do aplicativo, de página e outro chamado de *token* de acesso do cliente (FACEBOOK, 2015c). Em virtude desse trabalho utilizar somente o *token* de acesso do usuário e o *token* de acesso de páginas, só serão abordados os dois. Informações adicionais podem ser encontradas em Facebook (2015c).

De acordo com Facebook (2015c), o *token* de acesso do usuário é utilizado quando o aplicativo necessita ler ou escrever dados na linha do tempo de um usuário em específico, utilizando o nome deste usuário. Embora diferentes plataformas gerem os *tokens* de acesso por intermédio de diferentes API's, todas seguem o mesmo princípio (a Figura 1 demonstra o mecanismo):

1) Um cliente requisita um *token* de acesso;

[23] Em virtude desse trabalho utilizar o termo *token* no contexto de pré-processamento e no contexto do *Facebook*, a expressão *token* de acesso será utilizada para descrever o *token* no contexto do *Facebook*.
[24] É necessário criar um aplicativo juntamente com o *Facebook* com o intuito de realizar requisições que necessitam de permissões adicionais (FACEBOOK, 2015a).

2) Usuário autentica e aprova as permissões;
3) O *token* de acesso é retornado ao cliente (aplicativo).

Figura 2 - Mecanismo de obtenção de *tokens* de acesso.

Fonte: Produção própria do autor, 2015, com base em Facebook (2015c).

Há dois tempos de validade para os *tokens* de acesso. Os curtos e os longos. O primeiro, tem seu tempo de vida determinado entre 1 e 2 horas. Já o segundo (longo), possui validade de aproximadamente 60 dias a contar a partir do dia em que foi realizada a conexão. Cabe ressaltar que os *tokens* de acesso obtidos por intermédio de páginas *Web* possui validade curta, porém podem ser expandidos, se necessário (FACEBOOK, 2015c).

Conforme foi citado em Facebook (2015c) e Facebook (2015f), com a finalidade de obter um *token* de acesso é necessário autenticar ao *Facebook* (usuário da ferramenta). Essa autenticação pode ocorrer de duas formas: por intermédio de bibliotecas oficiais (recomendadas) ou por intermédio de bibliotecas de terceiros. Em função disso, foi optado em utilizar a *Javascript software development toolkit*[25] (SDK), pois esta é uma biblioteca oficial do *Facebook*.

2.4 DJANGO-FACEBOOK

Django-Facebook é uma biblioteca escrita em *Python*[26] que implementa a *Facebook Graph* API e é compatível com o *framework Django* (SCHELLENBACH, 2015). O *Django* tem como principal foco o desenvolvimento de aplicações *Web* de forma rápida e fácil (DJANGO, 2015). Segundo Brown (2015), O *Django* começou a ser desenvolvido em meados 2006, e sua versão mais atual é a 1.7 - que será utilizada nesse trabalho.

[25] Em inglês: conjunto de ferramentas de desenvolvimento de *software*.
[26] Disponível em: < https://www.python.org/>.

Foi optado em utilizar o *Django-Facebook* para manipular os dados advindos do *Facebook*, pois a biblioteca escolhida para fazer classificação textual está escrita em *Python* (será explanada na Seção 2.5) e a única implementação da *Facebook Graph* API disponível em *Python* é a *Django-Facebook* (FACEBOOK, 2015a).

2.5 NLTK-TRAINER E APRENDIZAGEM DE MÁQUINA

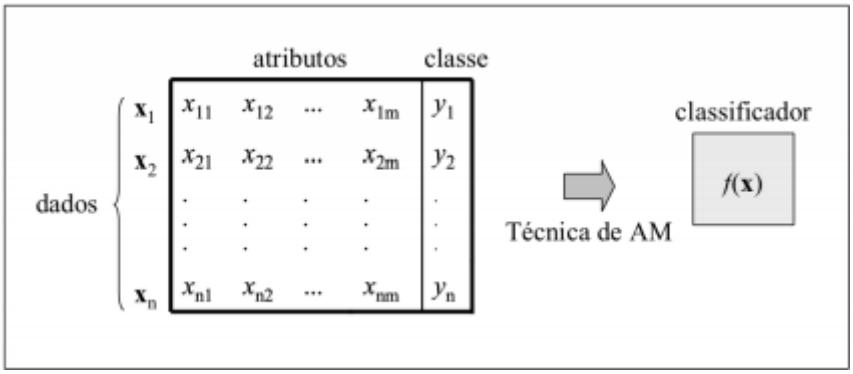
ML é uma área da inteligência artificial dedicada ao desenvolvimento de algoritmos que extraem conceitos (conhecimento) a partir de uma amostra de dados de entrada. Um dos conceitos que regem o ML é o conceito de indução, onde a partir de um conjunto de exemplos gera-se uma função que se aproxima da função que gerou os exemplos. Há dois tipos de aprendizagem de máquina indutivo, o primeiro é a aprendizagem supervisionada – o qual será objeto desse trabalho – e o outro é chamado de aprendizagem não supervisionada. (RUSSEL; NORVIG, 2009)

Conforme Russel e Norvig (2009), aprendizagem de máquina não supervisionada trata da extração de conhecimento a partir de um conjunto de entrada, sem haver um conjunto de saída para o mesmo. Por outro lado, quando há conhecimento do conjunto de saída para seu respectivo conjunto de entrada, entende-se por aprendizagem supervisionada, isto é, a partir de um conjunto X de entrada e outro conjunto Y de saída, gera-se uma função hipótese que se aproxima da função geradora dos conjuntos X e Y. Esse processo de aprendizagem indutivo é chamado de treinamento.

De acordo com Russel e Norvig (2009), geralmente os algoritmos de aprendizagem de máquina são utilizados para classificar dados. Por classificação entende-se como o ato de determinar a qual ordem ou rótulo um dado pertence.

A Figura 3 representa um classificador supervisionado, onde há um conjunto x_n dados representados por x_{nm} atributos. As variáveis y_i representam as classes. A partir dos exemplos e as suas respectivas classes, o algoritmo de ML extrai um classificador (LORENA; CARVALHO, 2007).

Figura 3 - Classificação em aprendizagem supervisionada.



Fonte: (LORENA; CARVALHO, 2007).

Em Perkins (2015) há um conjunto de *scripts* escritos em *Python* utilizando o *framework Natural Language Toolkit*[27] - em português: kit de ferramentas de linguagem natural - (NLTK), nomeados de *NLTK-Trainer*. Seu principal objetivo é treinar e avaliar classificadores da NLTK, sem implementar nada, isto é, apenas utilizando os *scripts*. O funcionamento desses *scripts* será abordado na próxima Seção.

Para Perkins (2014), NLTK é uma biblioteca escrita em *Python* utilizada para análise de texto e processamento de linguagem natural. Foi desenvolvida, inicialmente, para fins didáticos (ensino). Entretanto, ganhou notoriedade quando passou a ser utilizada em pesquisas, em função da sua facilidade de uso e aprendizagem.

Optou-se em utilizar esse conjunto de *scripts* escritos para o *NLTK* em função da sua facilidade de uso, aprendizagem, documentação, e pelo fato de ser o maior *framework* para processamento de linguagem natural e análise de texto (NLTK, 2015a).

2.5.1 Treinamento e avaliação dos classificadores

Conforme Hamilton (2012), com o intuito de comparar diferentes classificadores, se faz necessário utilizar um conjunto de dados rotulados para teste a fim de obter algumas métricas e posteriormente fazer a comparação empiricamente. Algumas das métricas para avaliação de classificadores geralmente utilizadas são: acurácia (*accuracy*), precisão (*precision*), *recall* e *média* harmônica (*F-measure*). Para entender essas métricas se faz necessário conhecer o conceito de alguns termos utilizados para fazer o cálculo das mesma. São eles: verdadeiro positivo (VP);

[27] Disponível em: < http://www.nltk.org/>.

verdadeiro negativo (VN); falso positivo (FP); falso negativo (FN). Para melhor elucidar esses conceitos, foi criado uma tabela - matriz de confusão - conforme o Quadro 2.

Quadro 2 - Matiz de confusão.

		Classe Prevista	
		Sim	Não
Classe Real	Sim	Verdadeiro Positivo - VP	Falso Negativo - FN
	Não	Falso Positivo - FP	Verdadeiro Negativo - VN

Fonte: Produção própria do autor.

Os VPs e os VNs representam os números de predições que foram corretamente previstos como positivo e negativos. Os FNs expressam o número de predições feitas para a classe "não" que não foram adequadas, isto é, pertenciam a classe "sim". Por último, os falsos positivos configuram as predições inadequadas para a classe "sim", ou seja, pertenciam a classe "não" (HAMILTON, 2012).

Segundo Hamilton (2012), acurácia é a medida que representa o número de predições corretas. Sua fórmula é dada por:

$$\frac{(VP+VN)}{(VP+VN+FP+FN)} \quad (1)$$

Quanto maior esta medida for, maior é a probabilidade do classificador atribuir rótulos adequadamente (PERKINS, 2014).

Conforme Hamilton (2012), *recall* ou taxa de VPs é proporção de casos positivos que foram corretamente classificados em relação ao total de positivos. É dada pela fórmula:

$$Recall = \frac{VP}{(VN+FN)} \quad (2)$$

Desse modo, o *recall* representa a capacidade de um classificador reconhecer todas as instâncias possíveis para uma determinada classe.

Para Hamilton (2012), a precisão mede a quantidade verdadeiros positivos que foram corretamente classificados. A fórmula é por:

$$\frac{VP}{(FP+VP)} \quad (3)$$

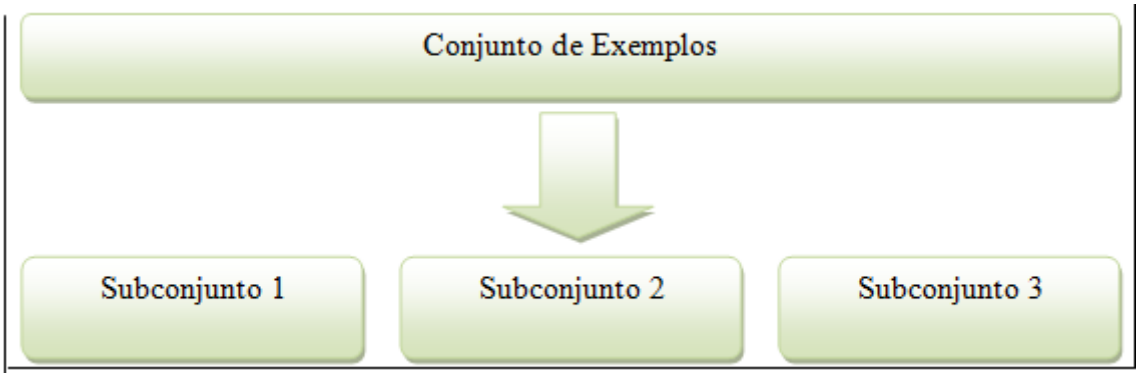
Por último, a média harmônica entre precisão e *recall* é dada por:

$$\frac{2rp}{(r+p)} \quad (4)$$

Onde r é *recall* e p é precisão. Para que as técnicas de aprendizagem de máquina classifiquem e avaliem corretamente os dados, deve-se atentar a superadaptação. Pelo fato de haver um grande conjunto de hipóteses possíveis, pode acontecer de a técnica classificar corretamente apenas os dados fornecidos como entrada, e não os outros dados, o que é chamado de superadaptação (RUSSEL; NORVIG, 2009).

De acordo com Russel e Norvig (2009), a validação cruzada é uma técnica estatística que diminui a superadaptação e pode ser aplicada em qualquer algoritmo de classificação. O pressuposto básico da validação cruzada "é estimar até que ponto cada hipótese irá prever dados não vistos" (RUSSEL; NORVIG, 2009). Um dos métodos de validação cruzada é conhecida como *k-fold*[28], que consiste em estratificar (dividir) os dados em k subconjuntos. Cada subconjunto possui $1/k$ dos dados para serem utilizados como testes, variando de acordo com cada subconjunto (RUSSEL; NORVIG, 2009), conforme **Erro! Fonte de referência não encontrada.**.

Figura 4 - Estratificação do conjunto de dados.

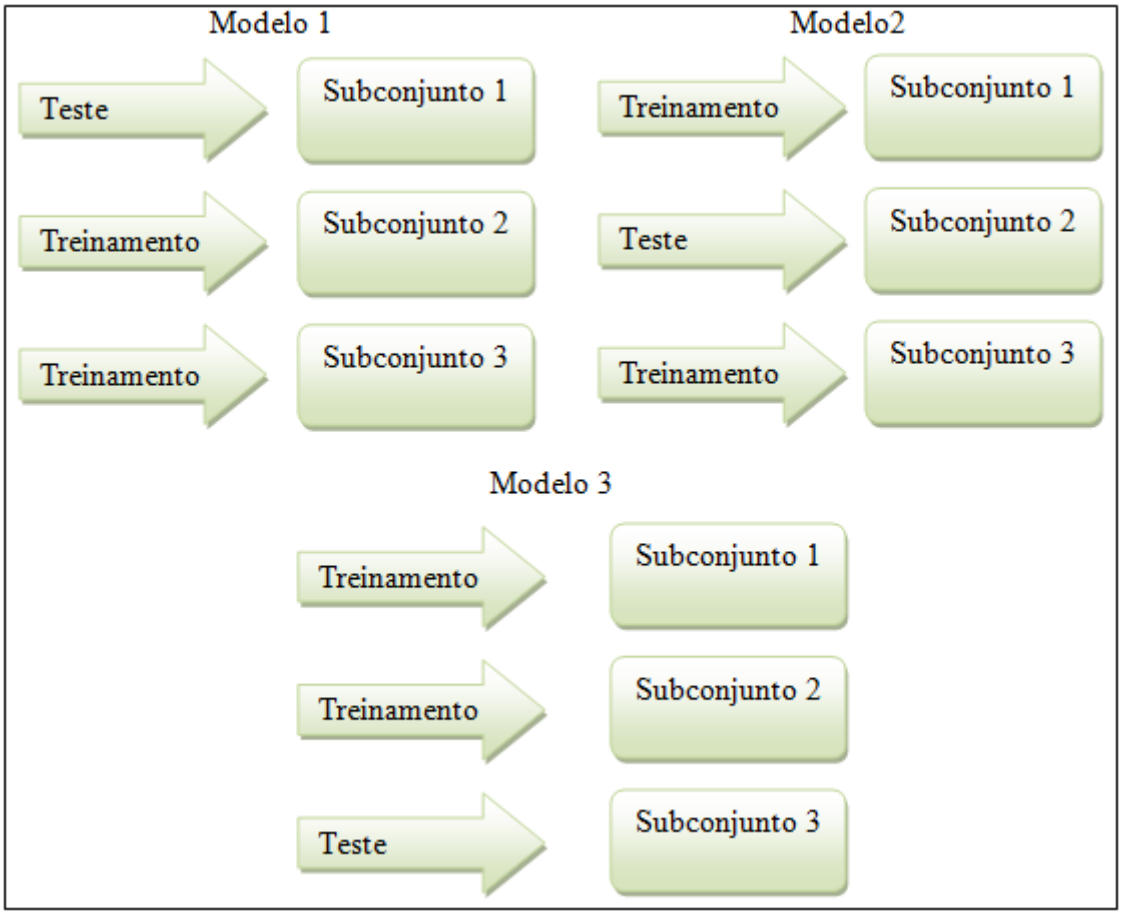


Fonte: Produção própria do autor.

Feito isso, deve-se treinar o classificador utilizando $k-1$ subconjuntos e utilizar o restante dos dados para o teste, gerando-se k modelos, conforme Figura 5.

[28] Sem tradução adequada para o português.

Figura 5 - Treinamento e avaliação dos modelos.



Fonte: Produção própria do autor.

Desse modo, geram-se modelos, onde ao final do treinamento, toma-se o modelo que obter as melhores métricas, isto é, o modelo com maior capacidade de generalização (PERKINS, 2014).

2.5.2 Naive Bayes

Conforme Russel e Norvig (2009), *Naive Bayes* é um método de aprendizado estatístico (supervisionado) que tem sido extensivamente estudado desde a década de 1950. Possui o nome *Naive Bayes* pois o primeiro termo, *naive* vem do inglês e significa ingênuo, e o segundo pelo fato de ser uma aplicação do teorema de *Bayes* (RUSSEL;

NORVIG, 2009). Informações adicionais sobre esse teorema podem ser obtidas em Russel e Norvig (2009).

Esse modelo de aprendizado é considerado ingênuo pois "supõe que os atributos são condicionalmente independentes uns dos outros, dada a classe" (RUSSEL; NORVIG, 2009, p. 696). Além disso, outra suposição que o *Naive Bayes* faz é que os dados são independente e identicamente distribuídos, ou seja, a probabilidade de um evento x acontecer não afeta a probabilidade de um evento y ocorrer (RASCHKA, 2014).

Essas suposições podem ser consideradas ingênuas porque alguns eventos são dependentes um dos outros, como por exemplo, séries temporais. Por outro lado, alguns outros eventos são condicionalmente independentes, como por exemplo, ao jogar uma moeda para cima 3 vezes seguidas, pode-se observar que na primeira jogada a moeda terá 50% de chance de cair com a coroa virada para cima, na segunda jogada a chance será a mesma e assim por diante (RASCHKA, 2014).

Conforme citado acima, é um método de aprendizado estatístico, e portanto, pode ser escrito conforme uma fórmula matemática:

$$\hat{y} = argmax_y \, P(y) \prod_{i=1}^{n} P(x_i \mid y) \quad (5)$$

Onde $P(y)$ é a probabilidade de ser uma classe y em relação as outras classes. $P(x_i \mid y)$ é a probabilidade condicional de o atributo x_i dado a classe y. $argmax$ é a função que maximiza a distribuição das probabilidades (SCIKIT-LEARN, 2014b).

Em outras palavras, o cálculo da hipótese mais provável (valor que maximiza a hipótese, também conhecido como máximo à priori) de um conjunto de atributos x pertencer a uma classe y é dada pela ponderação das probabilidades condicionais de x_i dado a classe y (SCIKIT-LEARN, 2014b). Para Russel e Norvig (2009 p.691) "a aprendizagem *bayesiana* simplesmente calcula a probabilidade de cada hipótese, considerando-se os dados e faz previsões de acordo com ela."

Conforme Raschka (2014), há diferentes implementações para o modelo *Naive Bayes*. Em função desse trabalho utilizar a implementação conhecida como *Multinomial Naive Bayes* - que será abordada na próxima Seção -, as outras técnicas não serão discutidas. Mais informações podem ser obtidas em Russel e Norvig (2009) e em Raschka (2014).

2.5.2.1 Multinomial Naive Bayes e Classificação Textual

O *Multinomial*[29] *Naive Bayes* é uma implementação para dados *multinomially*[30] distribuídos do algoritmo *Naive Bayes* muito utilizada para classificação textual, embora possa ser utilizada para outros problemas de classificação (MCCALLUM e NIGAM, 1998). Conforme Scikit-learn (2014b), a distribuição dos dados parametrizados é dada por $\hat{\theta} = (\theta_{y1}, \ldots, \theta_{yn})$ para cada classe y onde n é o tamanho das características (em problemas de classificação textual, é o tamanho do vocabulário), e θ_{yi} é a probabilidade $P(x_i \mid y)$ de uma característica i (palavra) aparecer na amostra pertencendo a classe y.

Segundo Scikit-learn (2014b), os parâmetros θ_y são estimados com base em uma versão suavizada da máxima a priori:

$$\hat{\theta}_y = \frac{N_{yi} + \alpha}{N_{y} + \alpha n} \tag{6}$$

Onde $N_{iy} = \sum_{x \in T} x_i$ é o número de vezes que i aparece em uma amostra da classe y em um conjunto de treinamento T, $N_y = \sum_{i=1}^{|T|} N_{yi}$ é o número total de características para a classe y.

Desse modo, com o objetivo de elucidar o funcionamento desse classificador, será apresentado um problema de classificação de textos da Ásia, onde as possíveis classes são Chinês e Japonês, conforme Quadro 3.

Quadro 3 - Conjunto de documentos com palavras e suas respectivas classes.

Uso	Documento	Palavras	Classe
Treinamento	1	Chinês Pequim Chinês	Chinês
Treinamento	2	Chinês Chinês Xangai	Chinês
Treinamento	3	Chinês Macau	Chinês
Treinamento	4	Tokio Japão Chinês	Japonês
Teste	5	Chinês Chinês Chinês Tokio Japão	?

Fonte: Produção própria do autor.

Seguindo a fórmula do *Naive Bayes* da Equação (6), primeiramente será determinado as probabilidades das classes (probabilidade à priori), conforme

[29] Termo que vem do inglês e pode ser traduzido para multinômio (tradução do autor).
[30] Sem tradução adequada para português.

MacCallum e Nigam (1998). No caso de $P(Chinês) = \frac{3}{4}$, pois o total de documentos com as classes é 4, sendo que 3 são chineses. Para o caso de $P(Japonês) = \frac{1}{4}$.

O próximo passo é calcular a probabilidade condicional de cada palavra x_i dada uma classe y. Para tal, calcula-se a frequência (quantidade de vezes) que uma dada palavra x_i aparece em um determinada classe y, adiciona-se 1 a esse valor, e divide o resultado dessa soma pelo total de palavras distintas (tamanho do vocabulário) presentes em todos os documentos juntos somados com o total de palavras presentes na classe y (MACCALLUM e NIGAM, 1998).

Desse modo, $P(Chinês \mid Chinês) = \frac{(5+1)}{(8+6)} = \frac{3}{7}$; $P(Tokyo \mid Japão) = \frac{(1+1)}{(3+6)} = \frac{2}{9}$; $P(Tokyo \mid Chinês) = \frac{(0+1)}{(8+6)} = \frac{1}{14}$; $P(Japão \mid Chinês) = \frac{(0+1)}{(8+6)} = \frac{1}{14}$; $P(Chinês \mid Japão) = \frac{(1+1)}{(3+6)} = \frac{2}{9}$; $P(Japão \mid Japão) = \frac{(1+1)}{(3+6)} = \frac{2}{9}$.

Feito isso, para cada classe é necessário computar as probabilidades condicionais obtidas, conforme a equação demonstrada na Equação (5). Assim, $P(Chinês \mid d5) = \left(\frac{3}{4}\right) \times \left(\frac{3}{7}\right)^3 \times \left(\frac{1}{14}\right)^2 = 0{,}0003$ e $P(Japão \mid d5) = \left(\frac{1}{4}\right) \times \left(\frac{2}{9}\right)^5 = 0{,}0001$. O último passo é determinar qual predição possui a maior probabilidade, isto é, a $argmax$ (MACCALLUM; NIGAM, 1998), que é a classe Chinês. Portanto, o rótulo previsto pelo *Multinomial Naive Bayes* é a classe chinesa.

3 TRABALHOS CORRELATOS

Nesta seção serão apresentadas as ferramentas identificadas na literatura que estão relacionadas ao trabalho proposto.

3.1 PROTÓTIPO PARA MINERAÇÃO DE SENTIMENTOS EM REDES SOCIAIS: ESTUDOS DE CASOS SELECIONADOS USANDO O TWITTER

Santos (2010) propôs um protótipo para desktop implementado em Java para realizar mineração de opinião (opinativo ou neutro) em textos expressos nas redes sociais, utilizando como alvo o *Twitter*. Conforme Santos (2010), para realizar a classificação de opiniões ou sentimentos expressos em comentários pelo canal do *Twitter*, os autores utilizam o método de aprendizagem máquina de vetores suporte[31] - *Support Vector Machine* (SVM) -, pelo fato de ser uma classificação binária (duas classes).

Segundo Santos (2010), para efetuar a classificação textual foi utilizado uma implementação do algoritmo[32] SVM em linguagem de programação C. Para realizar toda a coleta de dados e o processamento de linguagem natural, o Santos (2010) implementou uma ferramenta em Java.

Para validar o trabalho o autor submeteu a ferramenta a três casos de teste:

a) No primeiro caso de teste foram coletados 1885 *tweets*, sendo que destes, 350 foram utilizados para treinamento e 150 para teste. Após o treinamento foi obtido um modelo do SVM com acurácia de 80%. Em seguida, esse classificador avaliou o restante dos *tweets* (1385) e rotulou 1268 como sendo neutros e 108 opinativos. (SANTOS, 2010).

b) No segundo caso de teste, de um total de 109.213 *tweets*, 350 foram utilizados para treinamento e 150 para teste - pois essa amostra obteve uma acurácia satisfatória. Outro modelo do SVM foi gerado, este com 88% de acurácia. Após o treinamento, foi classificado 108.713 *tweets*,

[31] Informações adicionais sobre esse método podem ser obtidas em RUSSEL e NORVIG (2004).
[32] Disponível em: < http://svmlight.joachims.org/>

gerando 955 rotulados como opinativos e 107.758 como neutros. Em seguida, tomou-se o total de *tweets* opinativos (955) e foi treinado outro modelo do SVM com 300 *tweets* para treinamento (rotulados em positivo e negativo) e 100 para teste. Em seguida, 447 foram classificadas como positivos e 108 como negativos.

Em trabalhos futuros o autor deseja avaliar essa ferramenta em outros meios de coleta e mineração de opinião (SANTOS, 2010).

3.2 REAL-TIME SENTIMENT ANALYSIS IN SOCIAL MEDIA STREAMS: THE 2013 CONFEDERATION CUP CASE

Cavalin et al. (2014) desenvolveram a primeira ferramenta para análise de sentimento no *Twitter* em tempo real. O algoritmo utilizado para tal foi o *Naive Bayes*. A ferramenta foi executada na plataforma IBM - *International Business Machines* - *InfoSphere Streams*[33], e é capaz de rotular *tweets* em positivo, negativo ou neutro em grande escala, utilizando processamento distribuído.

Com a finalidade de treinar o *Naive Bayes,* foram coletados dados - *tweets*- durante 4 jogos amistosos entre Itália, Inglaterra, Bolívia e Chile. Ao final, foi obtido aproximadamente 11 milhões de *tweets*. Em seguida, os dados foram rotulados por 15 usuários através de uma plataforma *online* (construída para esse fim) e o total de dados rotulados foi de 2092. Durante o treinamento, optou-se em utilizar o método de validação cruzada *k-fold*, sendo que $k = 4$, e observou-se que a acurácia foi de aproximadamente 82% (CAVALIN et al. 2014).

Para validar a ferramenta, utilizou-se como estudo de caso a Copa das Confederações de 2013. Durante os testes, observou-se um pico de 60 mil de *tweets* por segundo, o que gera muitos dados a serem explorados, justificando a escolha do processamento distribuído. Após a análise dos dados, os autores puderam inferir, por exemplo, que o segundo gol que o Brasil fez contra o Chile aumentou os *tweets* em quase 50 mil (CAVALIN et al. 2014).

[33] Disponível em: < http://www-03.ibm.com/software/products/en/infosphere-streams/>.

Os autores sugerem como trabalho futuros, o teste da ferramenta em diferentes eventos, como por exemplo em campanhas de marketing. Outra perspectiva futura é avaliar métodos de aprendizagem não supervisionada, utilizando outras línguas (CAVALIN et al. 2014).

3.3 TWITTER SENTIMENT CLASSIFICATION USING DISTANT SUPERVISION

Go, Bhayani e Huang (2009), desenvolveram uma aplicação para análise de sentimentos (positivo ou negativo) no *Twitter* utilizando os classificadores *Naive Bayes*, *Maximum Entropy* (MAXENT)[34], SVM, com o intuito de avaliar qual é mais eficiente.

Antes de efetuar o treinamento dos classificadores, foi realizado pré-processamento. Mais precisamente, nomes de usuários foram removidos (palavras que inicial com o caractere '@', palavras com mais de duas letras repetidas foram normalizadas, *links* foram substituídos pelo *token* 'URL' (GO; BHAYANI E HUANG, 2009).

Os classificadores foram treinados com uma base de dados que foi obtida através de uma função heurística utilizando *emoticons*[35] - por isso aprendizagem com supervisão distante. *Unigrams, bigrams* foram levados em consideração. Ao final, percebeu-se que utilizando *unigram* e *bigrams*, os classificadores *Naive Bayes*, *MaxEnt* e SVM obtiveram acurácia de 81.3%, 80.5% e 82.2% respectivamente.

3.4 COMPARATIVO DOS TRABALHOS CORRELATOS

As ferramentas mencionadas neste capítulo possuem várias similaridades com a ferramenta proposta neste trabalho. Todas fazem análise de sentimentos (em diferentes classes) em redes sociais. A principal diferença da ferramenta proposta com os demais trabalhos: está disponível na *Web*, e portanto, não possui restrições de uso; além disso,

[34] Informações adicionais disponíveis em RUSSEL e NORVIG (2004).
[35] Termo que não tem tradução para português e descreve um conjunto de caracteres tipográficos, como por exemplo: ':-)', que expressam um determinado sentimento.

ela faz mineração de opinião na rede social *Facebook*, diferentemente dos outros trabalhos que analisam o *Twitter*.

Com o intuito de melhor elucidar as similaridades e diferenças dos trabalhos relacionados com esse trabalho, foi criado um quadro (Quadro 4) que fornece alguns critérios que distinguem os diferentes trabalhos. Os critérios utilizados são:

a) *K-fold*: Determina o número de amostras utilizadas durante o treinamento e avaliação do (s) classificador (es) - caso tenha sido utilizado a técnica de validação cruzada;

b) Algoritmo: Quais algoritmos foram utilizados para fazer a classificação;

c) Quantidade de classes rotuladas: Define o número de classes que a ferramenta é capaz de rotular;

d) Acurácia: Determina a taxa de acertos em relação ao total de predição da ferramenta;

e) Língua: Define a língua em que a ferramenta é capaz de rotular;

f) Pré-processamento: Se foi ou não utilizado pré-processamento durante o treinamento e avaliação;

g) *Collocations*: Define a quantidade de ngrams utilizados;

h) Tamanho de base de dados: Determina o tamanho da base de dados (*corpus*) utilizada durante o treinamento e avaliação dos classificadores.

Quadro 4 - Quadro comparativo dos trabalhos correlatos.

Aspecto	Santos (2010)	Cavalin et al. (2014)	Go; Bhayani e Huang (2009)	Trabalho Proposto
K-fold	Não utiliza	4	Não disponível	10
Algoritmo	SVM	*Naive Bayes*	*Naive Bayes*, Maxent e SVM	*Multinomial Naive Bayes*
Nº de classes	2	3	2	3
Acurácia	80% e 88%	82%	81%, 80.4% e 82.9% respectivamente	~85.92%
Língua	Inglês norte-americano	Português brasileiro	Inglês norte-americano	Português brasileiro
Pré-processamento	Sim	Sim	Sim	Não
Collocations	Não	*Unigrams, bigrams* e *trigrams*	*Unigrams* e *bigrams*	*Unigrams, bigrams e trigrams*
Nº de *corpus*	300 para treinamento e 150 para teste.	2092	1.600.000	9997
Método de rotulação de *Corpus*	Rotulados manualmente	Rotulados manualmente	Rotulados por intermédio de uma função heurística	Rotulados manualmente

Fonte: Produção própria do autor.

Conforme pode-se observar no Quadro 4, a ferramenta proposta neste trabalho possui a maior medida de acurácia, em relação as outras. Entretanto, não se pode afirmar que essa ferramenta possui maior capacidade de classificação, pois são bases de dados diferentes - exceto ao trabalho de Cavallin et al. (2014), pois a base de dados é semelhante (vide Seção 4.2.5.1) -. Desse modo, se a ferramenta proposta fosse rotular dados advindos com características semelhantes aos dados utilizados pelos autores, provavelmente sua acurácia seria menor.

Talvez o trabalho de Cavallin et al. (2014) possua menor acurácia em função da quantidade de amostras utilizadas na validação cruzada, ou talvez seja pelo tamanho da base de dados utilizada durante o treinamento - que é menor. Outro possível fator contribuinte para isto, seria o modelo do algoritmo *Naive Bayes* (que não foi fornecido), visto que a ferramenta proposta neste trabalho também utiliza o *Naive Bayes*.

4 DESENVOLVIMENTO

Nesta Seção será abordado o desenvolvimento da aplicação. Inicialmente é apresentada sua especificação. Em seguida apresenta-se a sua implementação, e por fim sua operacionalidade.

4.1 ESPECIFICAÇÃO

Nesta Seção será apresentada a especificação da ferramenta proposta neste trabalho. Serão relacionados os requisitos da aplicação, bem como os diagramas envolvidos.

4.1.1 Contextualização

A ferramenta proposta neste trabalho fará análise de sentimentos na rede social *Facebook* rotulando comentários textuais em positivo (bom), negativo (ruim), neutro (objetivo). A classificação de um dado comentário é determinada com base no uso de adjetivos, verbos, e o comentário deverá ser breve com no máximo 1200 caracteres, pois a base de dados utilizada para treinar a ferramenta possui sentenças curtas (vide Seção 4.2.5.1). Para tal, a ferramenta utilizará o classificador *Multinomial Naive Bayes*, que por sua vez, foi treinado utilizando a biblioteca *NLTK-Trainer*, conforme Seção (4.2.3). Por último, após a classificação, a ferramenta fornecerá essas informações ao usuário, oferecendo filtros pelos rótulos obtidos pelo classificador (positivo, negativo ou neutro).

Para alcançar os objetivos propostos nesse trabalho, o sistema deverá satisfazer os requisitos funcionais listados no Quadro 6, que foram extraídos com base nas regras de negócios, conforme Quadro 5.

Quadro 5 - Regras de negócio do sistema.

Identificação	Descrição
RN001	O usuário deverá informar seu respectivo e-mail e senha da conta no *Facebook*.
RN002	As publicações só serão mostradas ao usuário, caso elas contenham pelo menos 1 comentário.
RN003	Para realizar a classificação de forma adequada, os comentários deverão ser escritos em língua portuguesa do Brasil, com no máximo 1200 caracteres, e deverão possuir adjetivos e verbos.
RN004	Os filtros serão com base nas classes geradas pelo classificador (positivo, negativo ou neutro).
RN005	O usuário tem a necessidade de estabelecer um significado (bom, ruim ou neutro) para as postagens em sua página.

Fonte: Produção própria do autor.

Baseado nas regras de negócio foram levantados os requisitos funcionais (Quadro 6) que representam as funcionalidades da aplicação.

Quadro 6 - Requisitos funcionais do sistema.

Identificação	Descrição	Regras Envolvidas
RF001	O sistema deve realizar integração com o *Facebook*.	**RN001**
RF002	O sistema deve exibir as publicações textuais, e os respectivos *links* (endereço da publicação) presentes na linha do tempo do usuário.	**RN002**
RF003	O sistema deve fornecer possibilidade de realizar classificação de comentários textuais presentes nas postagens da linha do tempo no *Facebook*.	**RN003**
RF004	O sistema deve apresentar ao usuário os comentários e suas respectivas classes geradas após a classificação.	**RN005**
RF005	O sistema deve fornecer filtros para os comentários classificados.	**RN004**

Fonte: Produção própria do autor.

Por fim, foram levantados os requisitos não funcionais da aplicação. Estes requisitos não estão ligados diretamente as funcionalidades da aplicação, e sim a estrutura em que a aplicação foi desenvolvida. No Quadro 7 são listados os requisitos não funcionais para a aplicação proposta neste trabalho.

Quadro 7 - Requisitos não funcionais do sistema.

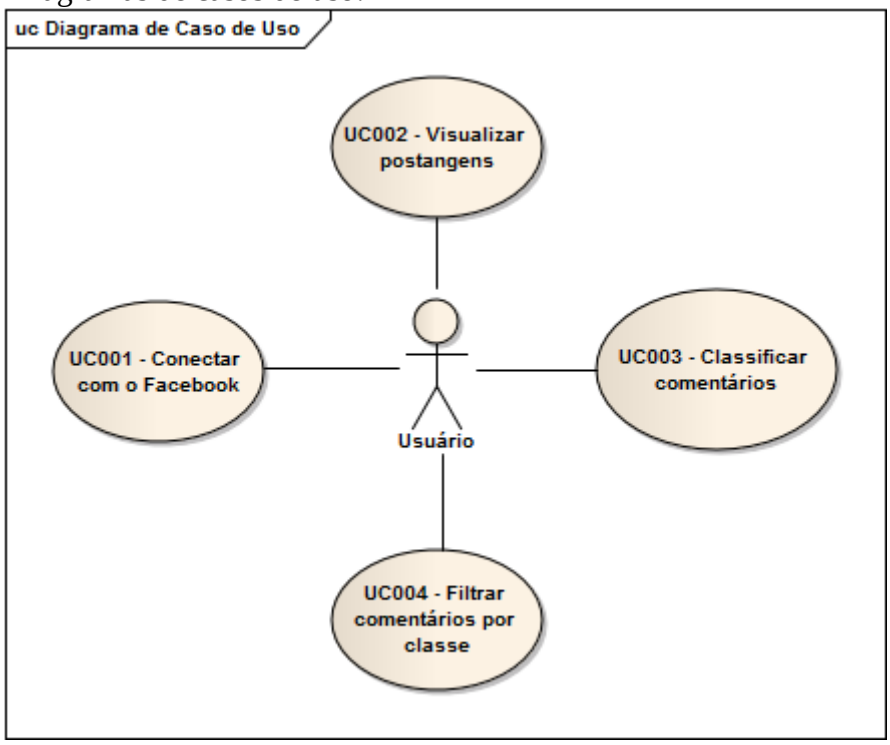
Identificação	Descrição
RNF001	O sistema deverá ser desenvolvido em linguagem de programação *Python* utilizando o *framework Django*.
RNF002	A classificação textual será realizada pelo classificador *Multinomial Naive* Bayes (disponível na biblioteca *NLTK-Trainer*), devidamente treinado.
RNF003	A comunicação cliente-servidor (usuário-*Facebook*) deverá ser realizada por intermédio da biblioteca *Django-Facebook*.
RNF004	A obtenção do *token* de acesso juntamente ao *Facebook* ocorrerá por intermédio da biblioteca *Javascript* SDK.

Fonte: Produção própria do autor.

4.1.2 Diagramas de Caso de Uso

Com o intuito de satisfazer os requisitos levantados para a ferramenta proposta neste trabalho, foram levantados alguns casos de uso, conforme Figura 6.

Figura 6 - Diagramas de casos de uso.



Fonte: Produção própria do autor.

Conforme observa-se na Figura 6, o ator está diretamente associado aos 4 casos de uso que serão explicado a seguir. O Quadro 8 apresenta o cenário do UC001.

Quadro 8 - Cenário do UC001 (Conectar com o *Facebook*).

UC001	**Conectar ao Facebook**
Descrição	Usuário autentica-se na rede social *Facebook*.
Rastreabilidade	RF001
Atores	Usuário
Pré-Condições	O usuário deve possuir uma conta no *Facebook*.
Fluxo Principal	1. Usuário clica no botão do *Facebook*. 2. A ferramenta apresenta uma caixa de diálogo com os campos email e senha do *Facebook*. 3. Usuário fornece email e senha e clica no botão "Entrar". 4. A ferramenta realiza uma requisição ao *Facebook* solicitando a validação dos dados e retorna o *token* de acesso, finalizando a autenticação.
Fluxo Alternativo	4a. Usuário e senha inválidos 4a_1. Usuário fornece outros valores para os campos: email e senha. 4a_2. Volta para o passo 4 do fluxo principal.

Fonte: Produção própria do autor.

O Quadro 9 apresenta o cenário especificado para o caso de uso UC002 Visualizar postagens que descreve a funcionalidade visualizar postagens.

Quadro 9 - Cenário do UC002 (visualizar postagens).

UC002	**Visualizar Postagens**
Descrição	Usuário visualiza as postagens textuais disponíveis na sua linha do tempo.
Rastreabilidade	RF002
Atores	Usuário
Pré-Condições	O usuário deve estar autenticado com o *Facebook*
Pós-Condições	O usuário visualiza postagens presentes na linha do tempo
Fluxo Principal	1. Usuário clica no botão referente as postagens. 2. A ferramenta apresenta os dados da linha do tempo ao usuário.
Fluxo Alternativo	2a. Não há postagens 2a_1. A ferramenta informa ao usuário que não há postagens disponíveis.

Fonte: Produção própria do autor.

O Quadro 10 aborda o cenário que foi especificado para o caso de uso UC003 que descreve o RF003.

Quadro 10 - Cenário do UC003 (classificar comentários).

UC003	Classificar Comentários
Descrição	Usuário seleciona opção referente a classificação dos comentários de uma postagem.
Rastreabilidade	RF003
Atores	Usuário
Pré-Condições	O usuário deve estar devidamente autenticado com o *Facebook*
Pós-Condições	O ator visualiza os comentários classificados.
Fluxo Principal	1. O usuário clica no botão referente a classificação de comentários. 2. A ferramenta classifica os comentários e apresenta os resultados.

Fonte: Produção própria do autor.

O cenário especificado para o caso de uso UC004 é apresentado no Quadro 11.

Quadro 11 - Cenário do UC004 (filtrar comentários rotulados).

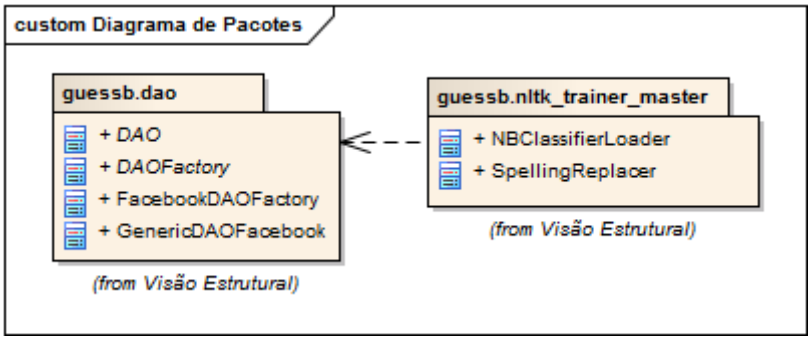
UC004	Filtrar Comentários Rotulados
Descrição	Usuário visualiza os comentários rotulados de acordo com o filtro selecionado.
Rastreabilidade	RF005
Atores	Usuário
Pré-Condições	Os comentários devem estar rotulados
Pós-Condições	Usuário visualiza os comentários rotulados filtrados por classe
Fluxo Principal	1. O usuário seleciona o filtro desejado. 2. A ferramenta filtra os comentários de acordo com a (s) classe (s) selecionada (s) e os apresenta.

Fonte: Produção própria do autor.

4.1.3 Diagramas de Classe

Para a aplicação proposta neste trabalho foi criado um diagrama de pacote de classes, o qual é mostrado na Figura 7. Desta forma, são mostradas as classes contidas em cada pacote, bem como a dependência entre eles. Nesta representação, os atributos e métodos foram abstraídos para uma melhor visualização.

Figura 7 - Diagrama de Pacotes.



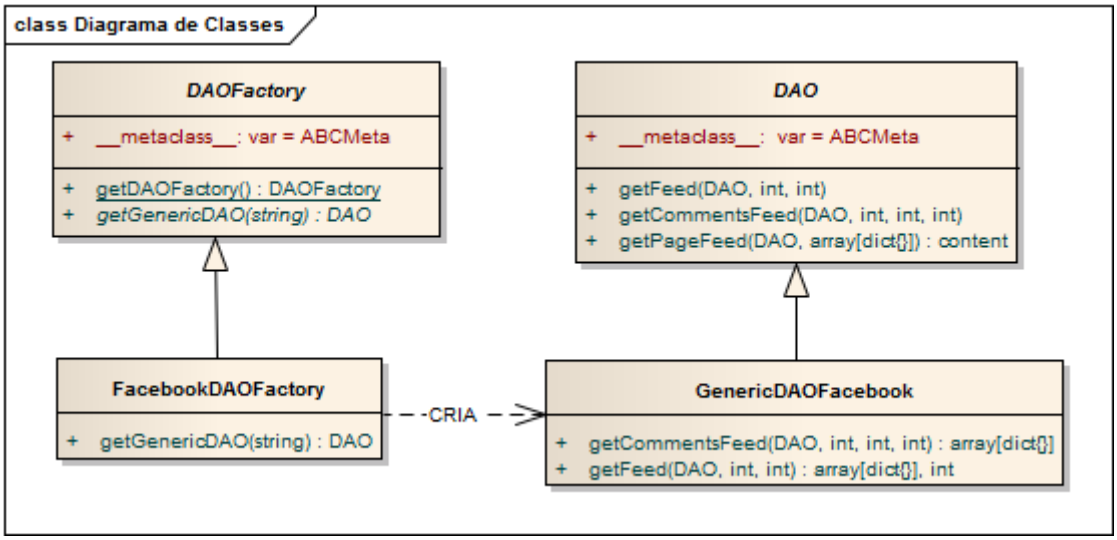
Fonte: Produção própria do autor.

Conforme pode-se observar nas classes da Figura 7, as classes estão separadas em pacotes de acordo com suas responsabilidades. Desse modo, as classes que estão dentro pacote *guessb.dao* têm a responsabilidade de acessar os dados do *Facebook* (ou de outras redes sociais). Isso é possível em função da camada de acesso aos dados ter sido desenvolvida utilizando o padrão de projeto *abstract factory*, que tem a responsabilidade de "fornecer uma interface para criação de famílias de objetos relacionados ou dependentes sem especificar suas classes concretas" (GAMMA et. al, 2000, p. 95). Em virtude disso, as classes de camada de acesso aos dados tornam-se modulares o suficiente a tal ponto que para adicionar outra fonte de dados (no contexto desse trabalho, uma nova rede social, como por exemplo o *Google+*), basta criar uma nova classe concreta e especificar na interface qual classe concreta deve ser utilizada.

As classes presentes no pacote *guessb.dao* estão representadas em um diagrama de classes expresso na Figura 8.

.

Figura 8 - Diagrama de classes da camada de acesso aos dados.



Fonte: Produção própria do autor.

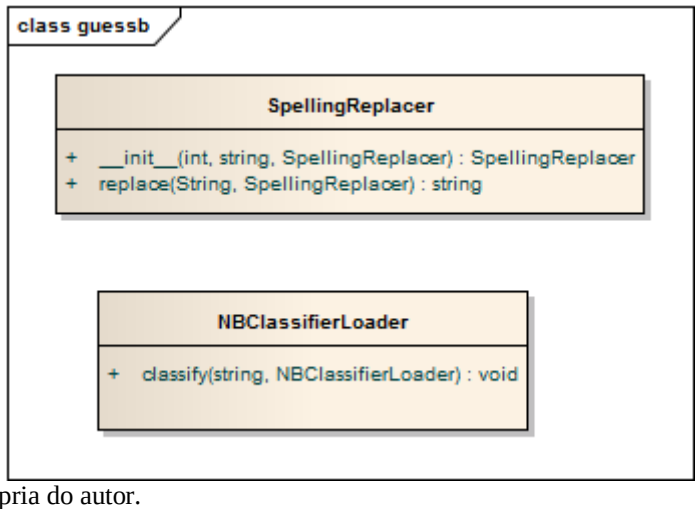
Conforme pode-se observar, há uma classe abstrata nomeada de *DAOFactory* que tem a responsabilidade de definir todos os métodos que deverão ser implementados (*getGenericDAO()*) pelas classes que a derivam. Além disso, há um método estático nomeado de *getGenericDAO(string)* que tem a responsabilidade de efetivamente criar a classe concreta (*FacebookDAOFactory)*. Por último, dentro há um atributo de escopo público *__metaclass__ = ABCMeta*, que define que essa classe poderá ter métodos abstratos, e caso tiver, será uma classe abstrata (PYTHON, 2015b).

A classe *FacebookDAOFactory* é a classe derivada da *DAOFactory* que tem a responsabilidade de instanciar a classe concreta *GenericDAOFacebook* através do método *getGenericDAO(string)*. Já a classe *GenericDAOFacebook* é uma especialização da classe abstrata *DAO* que tem dois método abstratos: *getFeed(DAO, int, int)*, onde o primeiro parâmetro é a instância, o segundo é o índice inicial dos comentários (utilizado na paginação), e o terceiro é o índice final. O segundo método, nomeado de *getCommentsFeed(DAO, int, int, int)* recebe como parâmetros a instância, o código identificador da publicação, o terceiro e o quarto representam os índices iniciais e finais, respectivamente.

O segundo pacote ilustrado na Figura 7, nomeado de *guessb.nltk_trainer_master* armazena as classes *NBClassifierLoader* e *SpellingReplacer* que tem a responsabilidade de carregar o classificador e corrigir a ortografia das

palavras, respectivamente. O diagrama das classes presentes neste pacote é apresentado na Figura 9.

Figura 9 - Diagrama de classe das entidades *NBClassifierLoader* e *SpellingReplacer*.



Fonte: Produção própria do autor.

Conforme ilustrado na Figura 9 na entidade *SpellingReplacer* (entidade responsável pela correção ortográfica das palavras) há o método construtor - *__init__(int, string, SpellingReplacer)* - que recebe como parâmetros, o número máximo de mudanças de caracteres necessários para corrigir uma palavra, de acordo com uma lista predefinida do dicionário *PyEnchant* (KELLY, 2011). O segundo parâmetro define qual idioma presente no dicionário será utilizado. Cabe ressaltar que essa classe foi criada para realizar correção ortográfica das palavras durante o pré-processamento, pois não havia nenhum mecanismo para fazer isso na biblioteca *NLTK-Trainer*.

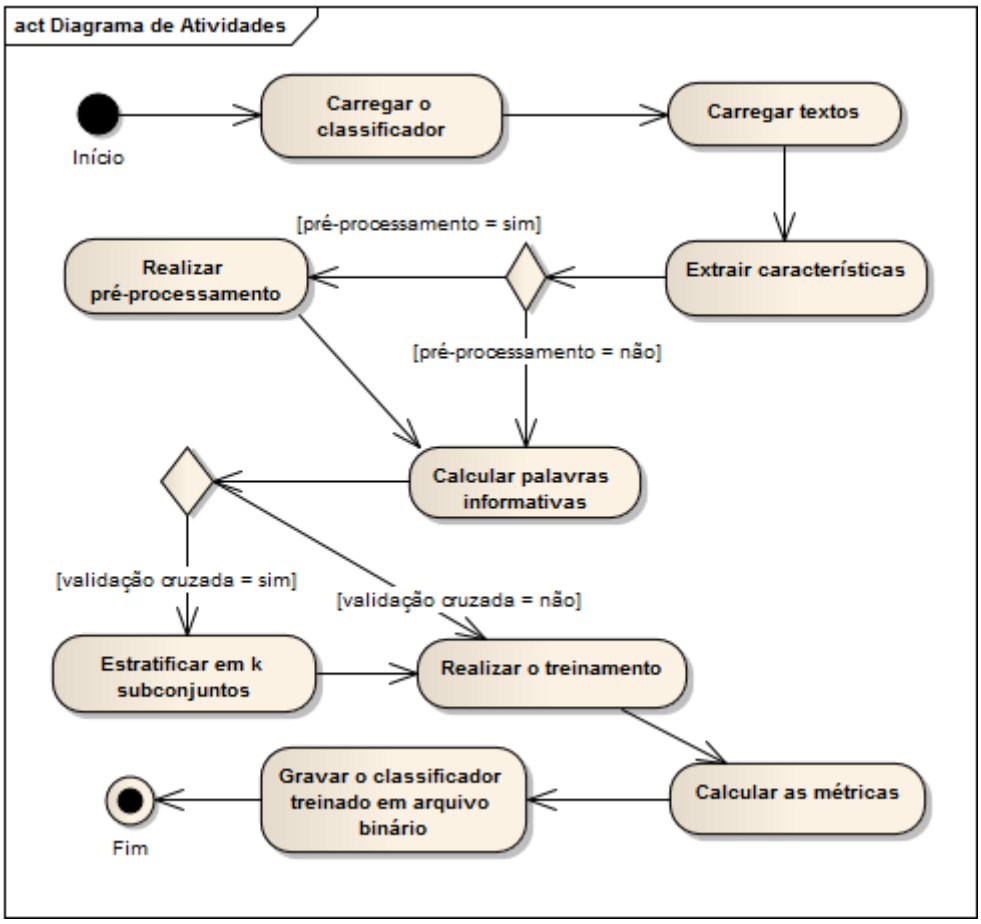
A segunda entidade (*NBClassifierLoader*) ilustrada na Figura 9, tem a responsabilidade de carregar o classificador em memória, passar como parâmetro o texto a ser classificado e retornar esse valor. Em outras palavras, essa classe funciona como um mediador entre a entidade *GenericDAOFacebook* e o classificador com os parâmetros ajustados de acordo com o treinamento.

4.1.4 Diagrama de Atividades

Com o intuito de explicar o funcionamento da biblioteca *NLTK-Trainer* e o treinamento do classificador, optou-se em utilizar o diagrama de atividades da UML, pois conforme Guedes (2009, p. 285) "a modelagem de atividades enfatiza a sequência e

condições para coordenar comportamentos de baixo nível. Dessa forma, o diagrama de atividades é o diagrama com maior ênfase ao nível de algoritmo da UML". Portanto, o treinamento do classificador será ilustrado no diagrama de atividades proposto na Figura 10.

Figura 10 - Diagrama de atividades do treinamento do classificador.



Fonte: Produção própria do autor.

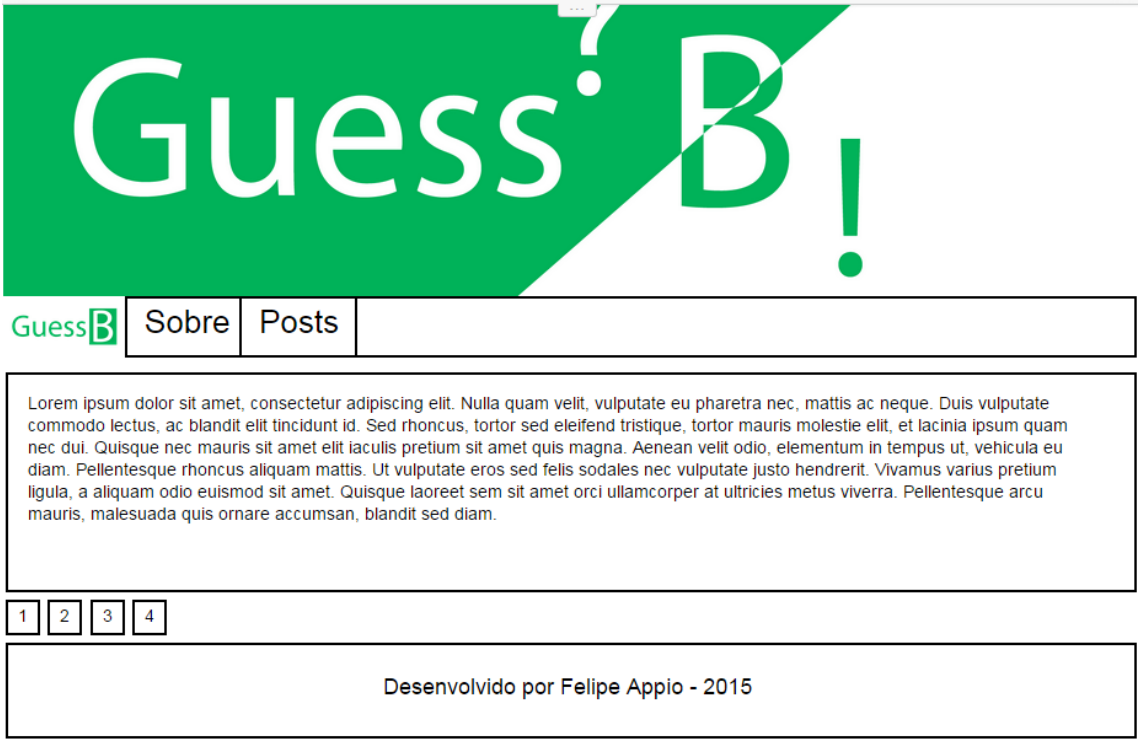
Conforme ilustrado na Figura 10 a primeira atividade (carregar o classificador) é responsável por efetivamente carregar os classificadores da biblioteca NLTK (2015). Em seguida, na próxima atividade os textos (*corpora*) são carregados em memória para posteriormente ser realizado o processo de *vectorization* (vide Seção 2.2.2.1). Feito isso, há um nó de decisão que define se o texto será pré-processado ou não. Em seguida são calculadas as palavras mais informativas (palavras que aparecem com frequência em determinada classe, por exemplo, a palavra "bom" na classe positivo). O próximo passo é realizar a estratificação dos dados (se for especificado) e efetivamente realizar o treinamento. Após o treinamento é efetuado o cálculo das métricas do classificador. Por

último, os parâmetros gerados pelo classificador são gravados em arquivo binário, que posteriormente será utilizado para carregar o classificador e efetivamente rotular os comentários.

4.1.5 Prototipação

Na etapa de especificação da ferramenta proposta neste trabalho foram criados três protótipos por intermédio da ferramenta *mockups*[36]. A Figura 11 apresenta o protótipo criado para a página principal da ferramenta.

Figura 11 - Protótipo da página principal da ferramenta.



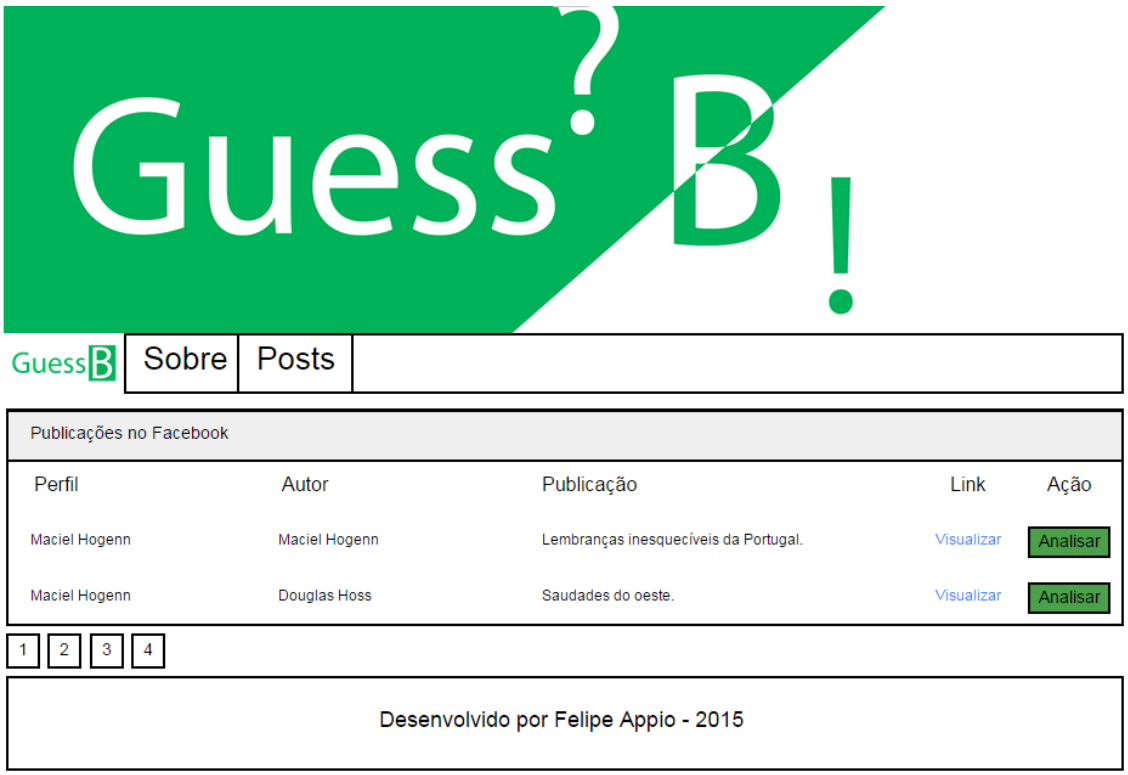
Fonte: Produção própria do autor.

Observa-se que na parte superior do protótipo encontra-se o logotipo da ferramenta. Logo abaixo (no centro) há um texto onde será apresentado uma breve descrição da ferramenta (o que ela faz, como ela faz). Logo abaixo do texto está a paginação, e por fim, o rodapé.

O próximo protótipo (Figura 12) apresenta as postagens textuais presentes na linha do tempo do usuário.

[36] Disponível em: <https://moqups.com/>.

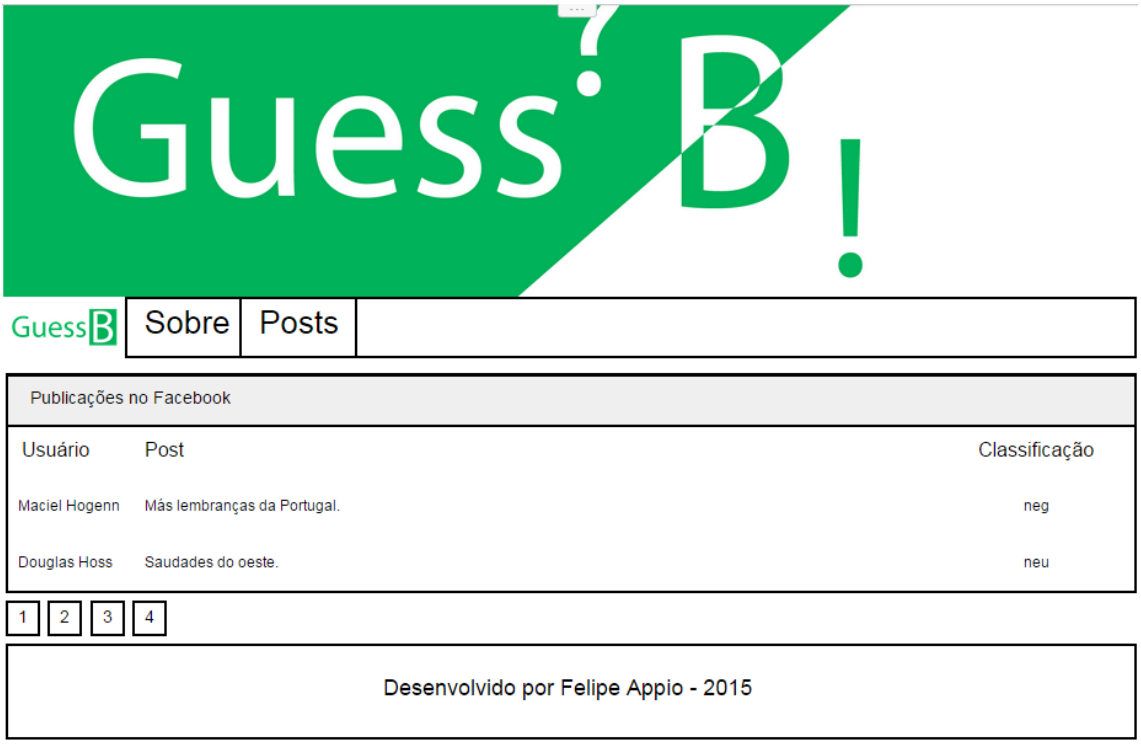
Figura 12 - Protótipo da página que apresenta as postagens textuais presentes na linha do tempo do usuário.



Fonte: Produção própria do autor.

Pode-se observar na figura anterior (Figura 12) que é possível visualizar a página do perfil (página do perfil do usuário da ferramenta, ou da (s) pagina (s) administrada (s) pelo usuário). Além disso, pode-se observar a publicação (conteúdo textual), o autor da publica, bem como o *link* com o endereço da postagem (quando houver). Por último, um botão com a funcionalidade de analisar a postagem. Quando o usuário clica no botão com a funcionalidade de analisar, o mesmo é direcionado para a página que apresenta os resultados da classificação (Figura 13).

Figura 13 - Protótipo da página que apresenta os resultados da classificação gerada pela ferramenta.



Fonte: Produção própria do autor.

A Figura 13 apresenta autor, comentário textual e o rótulo (classe) gerado pelo classificador de todos os comentários presentes em uma determinada postagem textual na linha do tempo do usuário.

4.2 IMPLEMENTAÇÃO

Esta Seção descreve o desenvolvimento da ferramenta. Inicialmente, abordam-se as tecnologias utilizadas, a estrutura utilizada para o projeto, bem como as técnicas que são utilizadas para fazer análise de sentimentos no *Facebook*.

4.2.1 Tecnologias Utilizadas

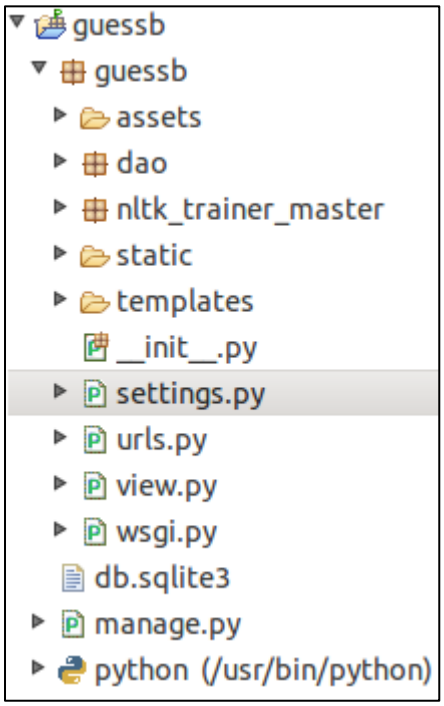
A ferramenta foi desenvolvida no sistema operacional *Linux Ubuntu* 14.04, utilizando a IDE *Eclipse luna* versão 4.4 juntamente com o *plugin PyDev*. A linguagem de programação utilizada foi *Python* versão 2.7. Além disso, como a ferramenta é

executada em ambiente *Web*, foi optado por utilizar o *framework Django* (vide Seção 2.4) versão 1.7. Com o intuito de fazer a comunicação com a rede social foi utilizado a biblioteca *Django-Facebook* (vide Seção 2.4).

4.2.2 Estrutura do Projeto

Esta Seção apresenta a estrutura do projeto desenvolvido na IDE *Eclipse luna*, utilizando o *framework Django* (Figura 14).

Figura 14 - Estrutura do projeto.



Fonte: Produção própria do autor.

Conforme pode-se observar na Figura 14, no diretório raiz do projeto há um arquivo nomeado de *manage.py* e um pacote nomeado com o mesmo nome do diretório raiz (*guessb*). O projeto está estruturado dessa forma, pois ao criar o projeto com o *Django framework* gera por padrão essa estrutura.

O arquivo *manage.py* é responsável por gerenciar o servidor (implantar projeto, parar, reiniciar (HOLOVATY, 2007). Já o pacote (*guessb*) armazena todos os arquivos que estão no projeto *Web* (HOLOVATY, 2007). Dentro desse pacote encontram-se alguns subdiretórios, arquivos e outro pacotes:

a) O pacote nomeado de *assets* armazena as imagens da ferramenta (*Web banner; cover image;* ícone, etc.);
b) O pacote *dao* armazena as classes relacionadas a camada de acesso a dados;
c) O pacote *nltk_trainer_master* armazena os módulos e classes relacionadas a biblioteca de análise de sentimento utilizada;
d) No subdiretório *templates* é armazenado os modelos (padrões) utilizados para gerar páginas *Web*;
e) O arquivo *settings.py* armazena as configurações do projeto, como por exemplo, o diretório *template* deve estar no *settings.py*;
f) O arquivo *url.py* mapeia os *links* e métodos que estão dentro do arquivo *view.py*. Desse modo, quando um determinado *link* é acessado, um determinado método dentro do arquivo *view.py* é invocado.

4.2.3 Integração com o Facebook

Nesta seção será apresentado os passos necessários para obter o *token* de acesso juntamente com o *Facebook*.

Conforme foi citado na Seção (2.3) é necessário possuir um *token* de acesso para adquirir um determinado conjunto de dados (postagens) do *Facebook*. Para tal, é necessário ter um aplicativo criado no *Facebook*, pois é necessário ter o código do aplicativo (vide Seção 2.3). Além disso, quando necessita-se de permissões adicionais (informações públicas, email, lista de amigos) é necessário enviar o aplicativo para que o *Facebook* o revisione (vide Seção 2.3). Os mecanismos de criação e revisão de aplicativos do *Facebook* não serão abordados, em função de não ser objetivo deste trabalho. Mais informações podem ser encontradas em Facebook (2015a).

Conforme foi citado na Seção (2.3), o mecanismo de autenticação utilizado na ferramenta proposta é por intermédio da *Javascript* SDK. De acordo com Facebook (2015d), para utilizar a *Javascript* SDK é necessário adicionar um botão social do *Facebook* (ilustrado na Figura 15).

Figura 15 - Integração com o *Facebook* utilizando a *Javascript* SDK.

```
<fb:login-button
  scope="public_profile", "user_posts"
        "user_status", "user_photos",
        "user_videos" onlogin="checkLoginState();">
        Posts </fb:login-button>
```

Fonte: Produção própria do autor.

Conforme pode-se observar na Figura 15, há duas *meta-tags*: *scope* e *onlogin*. A primeira determina as permissões que serão requisitadas pela ferramenta para o *Facebook*. A segunda determina que após a tentativa de autenticação ser realizada, será disparada a função *checkLoginState()*. Esta função está ilustrada na Figura 16.

Figura 16 - Função *Javascript checkLoginState()*.

```
function statusChangeCallback(response)
    if (response.status === 'connected')
        showPosts(response.authResponse);

function checkLoginState() {
    FB.getLoginStatus(function(response) {
        statusChangeCallback(response);
    });
}
```

Fonte: Produção própria do autor.

Conforme é ilustrado na Figura 16, a função *checkLoginState()* chama outra função nomeada de *statusChangeCallBack(reponse)* que, por sua vez, invoca outra função (*showPosts(authResponse)*) que simplesmente faz uma requisição para o servidor da ferramenta proposta e armazena o *token* de acesso na sessão.

4.2.4 Coleta dos dados

Nesta seção será demonstrado como foi realizado a implementação da coleta dos dados (postagens textuais) do *Facebook* utilizando a biblioteca *Django-Facebook*.

Conforme foi ilustrado na Seção (4.1.3), o projeto foi desenvolvido utilizando o padrão de projeto *abstract factory*. Em virtude disso, a ferramenta poderá funcionar com outras redes sociais, contanto que seja criado duas classes concretas: uma que especialize a entidade abstrata *DAOFactory* - esta conhecida como fábrica; e outra que implemente os métodos abstratos presentes na entidade abstrata *DAO* (Figura 17).

Figura 17 - Classe abstrata *DAO*.

```
class DAO:
    __metaclass__ = ABCMeta

    @abstractmethod
    def getFeed(self, firstIndex, lastIndex):pass

    @abstractmethod
    def getCommentsFeed(self, id, firstIndex, lastIndex):pass
```

Fonte: Produção própria do autor.

Conforme pode-se observar na Figura 17, esta entidade é uma classe abstrata e possui dois métodos abstratos: *getFeed(self)* que retorna todos as postagens textuais que estão na linha do tempo de um determinado usuário em uma dada rede social - no contexto desse trabalho, o *Facebook* -. Já o método *getCommentsFeed(self, id)* retorna todos os comentários de uma determinada publicação na linha do tempo do usuário de uma dada rede social.

A Figura 18 apresenta a classe abstrata (fábrica de objetos) *DAOFactory*.

Figura 18 - Classe abstrata *DAOFactory*.

```
class DAOFactory(object):
    __metaclass__ = ABCMeta

    @staticmethod
    def getDAOFactory():
        return FacebookDAOFactory()

    @abstractmethod
    def getGenericDAO(self, ACCESS_TOKEN):pass
```

Fonte: Produção própria do autor.

Conforme é possível observar, essa entidade possui um método estático *getDAOFactory()* que é responsável por criar a classe concreta (*FacebookDAOFactory*) e um método abstrato *getGenericDAO(self)* que deve ser implementado pelas classes que a derivam. A classe especialista da entidade *DAOFactory* é apresentada na Figura 19.

Figura 19 - Classe concreta *FacebookDAOFactory*.

```
15 class FacebookDAOFactory(DAOFactory):
16
17     def getGenericDAO(self):
18         return GenericDAOFacebook()
```

Fonte: Produção própria do autor.

Essa classe possui somente um método, que por sua vez retorna a entidade *GenericDAOFacebook*. A entidade *GenericDAOFacebook* implementa os dois métodos abstratos *getFeed(self, firstIndex, lastIndex), getCommentsFeed(self, id, firstInde, lastIndex)* da entidade *DAO*, que são ilustrados na Figura 20 e Figura 22, respectivamente. Além disso, ela um método nomeado de *getFeedPage(self, content)* que recebe por parâmetro um vetor de dicionários (*content*) e o adiciona as publicações referentes as páginas que o usuário administra.

Figura 20 - Implementação do método *getFeed(self, firstIndex, lastIndex)* da entidade *GenericDAOFacebook*.

```
def getFeed(self, firstIndex, lastIndex):
    facebookObject = OpenFacebook(self._accessToken_)
    feedData = facebookObject.get('me/feed')
    content = []
    profileName = facebookObject.get('me').get('name')
    profileId = facebookObject.get('me').get('id')

    for data in feedData['data']:
        if 'comments' in data:
            content.append(
                        dict(profileName=profileName,
                             profileId=profileId,
                             authorId=data.get('from').get('id'),
                             authorName=data.get('from').get('name', ''),
                             messageContent=data.get('message',
                                                     '(Postagem sem texto)'),
                             link=data.get('link', '#'),
                             postId=data.get('id')))

    self.getFeedPage(content)
    return content
```

Fonte: Produção própria do autor.

Conforme é possível observar na Figura 20, é instanciado um objeto do tipo *OpenFacebook* da biblioteca *Django-Facebook* que necessita do *token* de acesso do *Facebook* (SCHELLENBACH, 2011). Em seguida, por intermédio do método *get()* passando como parâmetro *'me'* - parâmetro que determina que será obtido os dados relacionados ao perfil do usuário da ferramenta, com base no seu *token* de acesso (FACEBOOK, 2015e) - é obtido um vetor de dicionários - estrutura de dado parecida

com um mapa (PYTHON, 2015) -. Em seguida, por intermédio dos parâmetros *name* e *id* Facebook (2015e), é armazenado nas variáveis *profileName* e *profileId* o nome e o código identificador do perfil do usuário no *Facebook*, pois esses dados serão utilizados futuramente para mostrar ao usuário da ferramenta em qual perfil (página do perfil do usuário ou uma determinada página que o usuário administra).

Com o intuito de percorrer todas as posições desse vetor, foi criado uma estrutura de repetição *for*, onde em cada iteração é adicionado em outro vetor de dicionário o código identificador do autor; nome do autor; mensagem (conteúdo); *link* da postagem; e o código da postagem. Por último, é o método *getFeedPage(content)*, que recebe por parâmetro um vetor de dicionários é invocado (Figura 21).

Figura 21 - Método *getFeedPage(self, content)* da entidade *GenericDAOFacebook*, que adiciona as publicações das páginas gerenciadas pelo usuário.

```
def getFeedPage(self, content):
    facebookObject = OpenFacebook(self._accessToken_)
    accountData = facebookObject.get('me/accounts')
    pages = accountData.get('data')

    for page in pages:
        pageAccessToken = page.get('access_token')
        facebookObject = OpenFacebook(pageAccessToken)
        feedData = facebookObject.get('me/feed')
        profile = facebookObject.get('me')

        for data in feedData['data']:
            if 'comments' in data:
                content.append(dict(profileName=profile.get('name'),
                                    profileId=profile.get('id'),
                                    authorId=data.get('from').get('id'),
                                    authorName=data.get('from').get('name', ''),
                                    messageContent=data.get('message',
                                                            '(Postagem sem texto)'),
                                    link=data.get('link', '#'),
                                    postId=data.get('id'),
                                    accessToken=pageAccessToken))
```

Fonte: Produção própria do autor.

Conforme é possível observar na Figura 20 e Figura 21, os métodos ilustrados nestas Figuras são muito semelhantes, porém possuem algumas diferenças. O método ilustrado na Figura 21 - *getFeedPage(self, content)* - por intermédio do parâmetro *'me/accounts'* o *OpenFacebook* retorna um vetor de dicionários contendo as informações referentes as páginas que o usuário da ferramenta administra Facebook (2015c). Por conta disso, é necessário percorrer todas as posições do vetor - que foi realizada por intermédio da estrutura de repetição *for*. Por último, além dos dados

mencionados no parágrafo acima, é armazenado o *token* de acesso de cada publicação nas páginas que o usuário administra, pois para visualizar os comentários de uma determinada postagem é necessário ter o *token* de acesso, e o *token* de acesso do usuário é diferente do *token* de acesso das páginas que o usuário administra (FACEBOOK, 2015c).

A Figura 22 apresenta a implementação do método que retorna todos os comentários presentes em uma determinada postagem na linha do tempo de um determinado usuário.

Figura 22 - Implementação do método *getCommentsFeed(self, firstIndex, lastIndex, id)* da entidade *GenericDAOFacebook*.

```
def getCommentsFeed(self, id, firstIndex, lastIndex):
    facebookObject = OpenFacebook(self._accessToken_)
    feedData = facebookObject.get('me/feed')
    content = []
    commentsLength = 0
    classifier = NBClassifierLoader()

    for i in  xrange(0, len(feedData['data'])):
        if (feedData['data'][i].get('id') == id):
            comments = feedData['data'][i].get('comments').get('data')
            commentsLength = len(comments)

            for j in xrange(0, commentsLength):
                if (j >= firstIndex) and (j < lastIndex):
                    messageContent = comments[j].get('message', '')
                    authorName = comments[j].get('from').get('name')
                    authorId = comments[j].get('from').get('id')
                    polarity = classifier.classify(messageContent)
                    content.append(dict(
                                        authorId=authorId,
                                        authorName=authorName,
                                        messageContent=messageContent,
                                        polarity=polarity))
            break
    return content, commentsLength
```

Fonte: Produção própria do autor.

Pode-se observar na Figura 22 o método *getCommentsFeed(sel,id, firstIndex, lastIndex)* é semelhante ao método *getFeed(self, firstIndex, lastIndex)*. A diferença entre os dois é que o primeiro método recebe um parâmetro que representa o código identificador da postagem na linha do tempo. Além disso, para cada publicação presente na linha do tempo do usuário é verificado se o código identificador do

comentário é igual ao código passado por parâmetro. Caso essa condição seja verdadeira, então os dados: autor do comentário; código identificador do autor do comentário; a mensagem; e o rótulo gerado pelo classificador (que é obtido por intermédio do método *classify(self, text)* da entidade *NBClassifierLoader*).

O método ilustrado na Figura 22 retorna um vetor de dicionários contendo todos os comentários de uma determinada publicação. Além disso, ele também retorna a quantidade de total de comentários presentes em uma determinada publicação, pois é necessário saber esse número para fazer a paginação.

4.2.5 Classificação

Nesta seção as questões relacionadas ao classificador serão abordadas. Inicialmente é apresentando a base de dados (*corpus*) utilizada para o treinamento. Em seguida, apresenta-se como foi realizado o seu treinamento e avaliação. Por fim, é apresentado a justificativa para o uso do *Multinomial Naive* Bayes.

4.2.5.1 Base de dados utilizada durante o treinamento

Santos e Ladeira (2014) desenvolveram uma aplicação para mineração de opiniões baseada em um conjunto de comentários - aproximadamente 759 mil - em português brasileiro, obtidos do *Google Play*[37]. Esses comentários possuem certas peculiaridades, como por exemplo, gírias, abreviações, erros de digitações. O objetivo do trabalho é verificar se o pré-processamento é efetivo ou não. Para tal, um dos autores, juntamente com outro pesquisador, rotularam manualmente 10 mil comentários em positivo, negativo ou neutro.

Entrou-se em contato com os autores via *email* solicitando essa base com 10 mil comentários rotulados. Os autores colaboraram com o pedido, e cederam três arquivos em formato *CSV* contendo a polaridade (positivo, negativo ou neutro) identificação do usuário e o respectivo comentário.

[37] Plataforma para baixar aplicativos para *Android* (GOOGLE, 2015).

Desenvolveu-se então uma aplicação usando linguagem de programação Java (versão 1.8.45) utilizando a IDE para transformar os arquivos, pois a biblioteca utilizada neste trabalho (*NLTK-Trainer*) utiliza um formato diferente para treinar o classificador. Assim, cada comentário foi migrado para um arquivo no formato *txt*. Desse modo, cada comentário fica gravado em um arquivo *txt* diferente. Para discriminar as diferentes classes, foram criadas três pastas com o nome de *neu*, *pos*, *neg*. Evidentemente, dentro de cada uma delas, ficam os comentários rotulados.

4.2.5.2 Treinamento/Escolha

Conforme Perkins (2014), *NLTK-Trainer* é um conjunto de *scripts*, cuja finalidade é facilitar o desenvolvimento de aplicações que fazem uso de técnicas de aprendizagem de máquina para classificação textual. Pelo fato de *python* - linguagem em que essa biblioteca foi desenvolvida (vide Seção 2.5) - ser uma linguagem interpretada, é possível treinar diferentes classificadores - disponíveis em Scikit-learn (2014a) e NLTK (2015).

Segundo Perkins (2014), para utilizar os classificadores do *NLTK-Trainer* recomenda-se a instalação de algumas dependências: *numpy*[38]*; scipy*[39]*; scikit-learn*[40]*; argparse*[41]. O treinamento dos classificadores foi feito na verão 1.9.2 do *numpy*, versão 0.15 do *scipy*, 0.16 da biblioteca *scikit-learn* e versão 2.7.10rc do *argparse*. A versão utilizada do *python* foi a 2.7.9 que vem com o sistema operacional *Ubuntu* versão 14.04.

Os classificadores avaliados neste trabalho são: árvores de decisão; classificador de florestas randômicas; regressão logística; três implementação do teorema de *bayes*; *extra trees*; *k-neighbors*. Conforme Scikit-learn (2014b), todos são classificadores multiclasses - exceto a técnica de regressão logística, que faz necessário a especificação de um parâmetro - por natureza. Portanto, a escolha de avaliar esses classificadores no contexto desse trabalho se justifica pelo caráter multiclasses do problema (positivo, negativo ou neutro).

---

[38] Uma biblioteca para computação científica em *python* (NUMPY, 2013).
[39] Outra biblioteca para computação científica em *python* (SCIPY, 2015).
[40] Uma biblioteca para aprendizagem de máquina (SCIKIT-LEARN, 2014a).
[41] Um módulo em *python* para escrever linhas de comando de forma amigável (PYTHON, 2015b).

Para efetuar o treinamento utilizando o *script train_classifier.py* se faz necessário invocar o interpretador com a palavra chave *python e* posteriormente passar como argumento o endereço do *script* (PEARKINS, 2014). Logo em seguida, é necessário passar os parâmetros do *script* com a finalidade de treinar o classificador. A lista completa dos parâmetros pode ser obtida em Perkins (2011). Entretanto, para o treinamento dos classificadores os seguintes parâmetros foram utilizados para fazer o treinamento sem pré-processamento são: *ngrams; classifier*; *min_score*; *instances; multi cross-fold; corpus*.

O parâmetro *ngrams* recebe um número inteiro que define a quantidade de colocações a serem utilizadas durante o treinamento. Foi escolhido utilizar 1 2 3, pois as colações aumentam o desempenho do classificador (vide Seção 2.3). O parâmetro *classifier*, conforme o próprio nome já diz, define a técnica de aprendizagem de máquina a ser utilizada. Já o parâmetro *instances* define um grupo de palavras que representam uma única instância do treinamento. As opções disponíveis para esse parâmetro, são: *files, paras, sents*. Foi utilizado o parâmetro *paras*, pois em relação aos outros dois, este obteve maior acurácia, de um modo geral. O parâmetro multi define que classificador classificará em mais de duas classes - multiclasses.

O parâmetro *min_score* define a pontuação mínima das palavras de alta informação que serão utilizadas durante o treinamento - palavras que tendem a aparecer somente em um determinado rótulo (PERKINS, 2014). Avaliando empiricamente no intervalo de 1 a 10, observou-se que a pontuação mínima de 5 foi a que obteve melhor desempenho. *Corpus* define o caminho dos arquivos utilizados para o treinamento, relativo ao caminho do diretório *nltk_data* (que pode ser configurado manualmente durante a instalação do NLTK). *Cross-fold* define o tamanho da amostragem da validação cruzada. Foi optado por utilizar 10, pois conforme Perkins (2014), esse valor é comumente utilizado na literatura científica, e pode ser considerado um bom tamanho de amostragem.

Após efetuar o treinamento com base nos parâmetros acima explanados foi construído uma tabela com os valores das métricas de desempenho obtidas, conforme Quadro 12. Para que o quadro coubesse em uma única página, o nomes dos classificadores foram abreviados: *Decision Tree* (DT); *Random Forest* (RF); *Logistic Regression* (LR); *Bernoulli Naive Bayes* (BNB); *Gaussian Naive Bayes* (GNB);

Multinomial Naive Bayes (MNB); *K-Nearest Neighbor* (KNN); *Gradient Boosting* (GB); *Extra Tress* (ET).

Quadro 12 - Métricas dos classificadores avaliados.

Classificador	*DT*	*RF*	*LR*	*BNB*	*GNB*	*MNB*	*KNN*	*GB*	*ET*
Acurácia média (desvio padrão)	**~0.76 (~0.01)**	**~0.78 (~0.01)**	**~0.84 (~0.01)**	**~0.83 (~0.01)**	**~0.84 (~0.01)**	**~0.86 (~0.01)**	**~0.73 (~0.02)**	**~0.78 (~0.05)**	**~0.80 (~0.01)**
Precisão Média (desvio padrão)	~0.68 (~0.14)	~0.73 (~0.11)	~0.80 (~0.09)	~0.79 (~0.08)	~0.83 (~0.10)	~0.81 (~0.10)	~0.67 (~0.15)	~0.71 (~0.01)	~0.74 (~0.11)
***Recall* Média**	~0.68 (0.14)	~0.67 (~0.19)	~0.77 (~0.13)	~0.75 (~0.14)	~0.76 (~0.13)	~0.80 (~0.10)	~0.58 (~0.26)	~0.67 (~0.11)	~0.70 (~0.18)
***F-Measure* Média**	~0.68 (~0.14)	~0.68 (~0.10)	~0.79 (~0.11)	~0.77 (~0.10)	~0.77 (~0.10)	~0.81 (~0.09)	~0.59 (~0.20)	~0,69 (~0.02)	~0.71 (~0.14)

Fonte: Produção própria do autor.

O Quadro 12 apresenta as médias das quatro métricas de desempenho de classificador (vide Seção 2.5.1), bem como o desvio padrão de cada média. O desvio padrão é uma medida de dispersão utilizada para calcular a dispersão de um dado valor em relação a média. Quanto mais baixo o desvio padrão, mais próximo a média esse

valor está. Por outro lado, quanto mais alto, maior é a dispersão entre do valor em relação a média (MUNDO EDUCAÇÃO, 2015). Desse modo, procura-se o desvio padrão mais baixo possível.

Analisando o Quadro 12, é possível observar que a técnica *Multinomial Naive Bayes* do método *Naive Bayes* foi a que obteve a melhor acurácia . Além disso, seu desvio padrão manteve-se uniforme (baixo). Portanto, optou-se em utilizar esse classificador neste trabalho.

Os parâmetros *filter-stop-words, punctuation, stemmer, spelling-replacer* são parâmetros utilizados para fazer pré-processamento antes do treinamento do classificador. O primeiro remove as *stopwords*, com base em uma lista fornecida pelo NLTK. No caso dessa ferramenta, foi utilizado as *stopwords* em português. Em seguida, o argumento *punctuation* remove as pontuações das sentenças.

Os parâmetros *stemmer* e *spelling-replacer* foram adicionados a biblioteca *NLTK-Trainer*. O primeiro define o *stemmer* da biblioteca NLTK a ser utilizado para realizar o *stemming* das palavras (redução do léxico para a forma infinitiva da palavra). Para esse parâmetro optou-se em utilizar o *RSLPStemmer*, pois o mesmo realiza normalização para palavras da língua portuguesa. Por último, *spelling_replacer* determina que será utilizado uma classe para corrigir letras repetidas (correção ortográfica) em palavras com base na biblioteca *PyEnchant* (KELLY, 2011).

Foi optado por avaliar se o pré-processamento afeta o desempenho do classificador. Portanto, foi treinado o modelo (classificador) com as melhores métricas (*Multinomial Naive Bayes*), porém com funcionalidades de pré-processamento. Ou seja, antes de realizar o treinamento e avaliação do modelo, foi feito pré-processamento do texto. O Quadro 13 apresenta os resultados coletados.

Quadro 13 - Métricas do *MultinomialNB* com e sem pré-processamento.

	Com pré-processamento	**Sem pré-processamento**
Acurácia média (desvio padrão)	**0.844032 (0.014215)**	**0.859178 (0.011793)**
Precisão média (desvio padrão)	0.802493 (0.094317)	0.815527 (0.096739)
***Recall* média (desvio padrão)**	0.779889 (0.120447)	0.805076 (0.098330)
***F-measure* média (desvio padrão)**	0.790000 (0.104804)	0.809728 (0.095404)

Fonte: Produção própria do autor.

Conforme pode-se observar no Quadro 13, comparando as métricas da técnica *Multinomial Naive Bayes* com e sem pré-processamento, constata-se que o pré-processamento afeta de forma negativa o desempenho do classificador. Portanto, a ferramenta proposta utiliza um classificador treinado sem pré-processamento.

4.3 OPERACIONALIDADE

Nesta seção será apresentada a operacionalidade da ferramenta, mostrando o seu funcionamento, passo a passo. Quando o usuário acessa a ferramenta é direcionado para página principal da aplicação, que fornece uma descrição da ferramenta (do que se trata), conforme Figura 23.

Figura 23 - Página principal da ferramenta.

Fonte: Produção própria do autor.

Na parte superior da ferramenta pode-se observar a barra de navegação (padrão em todas as páginas da ferramenta), e o logotipo que também é padrão. Para visualizar as postagens textuais presentes na linha do tempo do usuário, ele deve clicar no menu *Posts* (botão referente as postagens textuais na linha do tempo do usuário) na barra de navegação. Caso o usuário não esteja conectado ao *Facebook* será mostrado uma caixa de diálogo do *Facebook*, onde será apresentado os campos de *email* e senha (Figura 24).

Figura 24 - Caixa de *login* do *Facebook*.

Fonte: Produção própria do autor.

Do contrário - caso o usuário esteja conectado ao *Facebook* - será exibido uma caixa de diálogo informando que a ferramenta deseja obter permissões para acessar os dados do perfil do usuário no *Facebook*, conforme Figura 25.

Figura 25 - Caixa de diálogo que apresenta ao usuário as permissões necessárias para acessar os dados no *Facebook*.

Fonte: Produção própria do autor.

Em seguida o usuário será direcionado para a página que apresenta as postagens textuais que estão na sua linha do tempo (Figura 26).

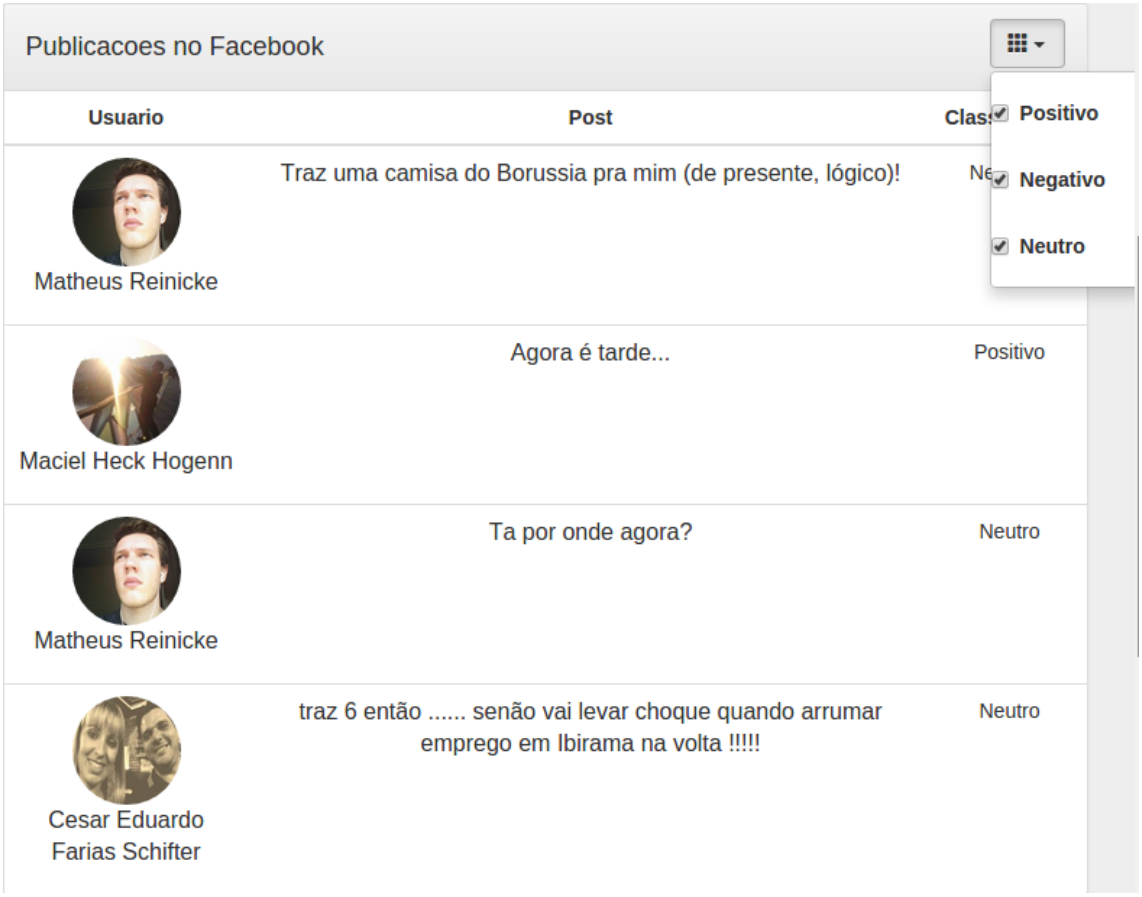
Figura 26 - Página que apresenta as postagens textuais que estão na linha do tempo do usuário.

Fonte: Produção própria do autor.

Conforme pode-se observar na Figura 26, pode-se visualizar o perfil (página que o usuário administra ou a página do seu perfil), o nome completo do autor da postagem, bem como sua respectiva imagem no perfil do *Facebook*. Além disso, ao lado do autor, há a referida postagem textual (conteúdo) e o *link* com o endereço da postagem. Por último, há um botão com um símbolo semelhante à um rótulo onde é possível realizar a classificação textual automática.

Quando o usuário clica no botão do rótulo, a ferramenta rotula todos os comentários textuais daquela determinada postagens e os apresenta ao usuário (Figura 27).

Figura 27 - Página que apresenta os comentários de uma determinada postagem.



Fonte: Produção própria do autor.

É possível observar na Figura anterior (Figura 27) que a página é muito semelhante a anterior (vide Figura 26). Ela foi desenvolvida nesse formato, pois as duas páginas apresentam conteúdo semelhante. As principais diferenças entre elas são: a página anterior (Figura 26) possui as páginas que o usuário da ferramenta administra, bem como os *links* das postagens presentes nessas páginas. Já a página ilustrada na Figura 27 apresenta o autor do comentário, o conteúdo do comentário e o rótulo que o classificador atribuiu. Além disso, essa última além de apresentar o rótulo (classe) gerada pelo classificador, fornece um filtro por classes. Desse modo, nas páginas que possuem muitos comentários, é possível selecionar somente os negativos, por exemplo.

4.4 RESULTADOS EXPERIMENTAIS

Este trabalho teve como resultado o desenvolvimento de uma ferramenta *Web* capaz de realizar análise de sentimento em postagens textuais na rede social *Facebook*.

Com a finalidade de avaliar a ferramenta foram selecionados 150 comentários textuais do perfil disponíveis no perfil de Juvenal Antena[42]. Em seguida, destes 150 comentários, foram rotulados empiricamente 90 comentários em positivo, negativo ou outro.

Optou-se em utilizar esse perfil por questões éticas - o perfil foi criado justamente para realizar os testes com a ferramenta proposta -. Além disso, em virtude da ferramenta ter sido treinada utilizando uma base de comentários textuais advindos do *Google Play* (vide Seção 4.2.5.1), os comentários publicados no perfil de Juvenal Antena foram extraídos do *Google Play*. Cabe salientar que estes comentários não são os mesmos utilizados durante o treinamento, pertencem apenas ao mesmo contexto (*Google Play*).

Após a execução da ferramenta foram obtidos os dados que demonstram os resultados gerados pela ferramenta. Desse modo, para cada uma das três classes obteve-se um percentual de acerto e erro. O Quadro 14 ilustra esses dados, onde há uma coluna nomeada de classe real representando as classes que foram rotuladas empiricamente, e outra coluna nomeada de classe prevista que representa as classes que foram geradas pela ferramenta.

Quadro 14 - Resultados gerados pela ferramenta para as classes positivo, negativo e neutro.

		Classe Prevista		
		Positivo	Negativo	Neutro
Classe Real	Positivo	~83,3%	~3,3%	~13.3%
	Negativo	~13,3%	~63,3%	~23,3%
	Neutro	20%	10%	70%
	Erro	~16,6%	~36,6%	~30%
Taxa de acerto		**~72,2%**		
Taxa de erro		**~27,8%**		

Fonte: Produção própria do autor.

[42] Disponível em: < https://www.facebook.com/profile.php?id=100009652808830>.

Conforme pode-se observar no Quadro 14, o percentual de acertos para a classe positiva foi de ~83,3%, ou seja, 25 comentários de um total de 30 foram rotulados adequadamente como sendo positivos. O restante, ~16,6%, foram rotulados inadequadamente como sendo ~3,33% negativo e ~13,33% positivo.

Para a classe negativa, é possível observar uma taxa de acerto não muito satisfatória (~63,3%), se comparada com a taxa de acerto para a classe positiva. Além disso, aproximadamente 37% (11 comentários) foram rotulados de forma inadequada, sendo que 7 comentários negativos (~23,3%) foram rotulados como neutros e os 4 restantes (~13,3%) como sendo positivos.

Por último, pode-se observar no Quadro 14, que a taxa de acerto para a classe neutro foi de 70%, ou seja, 21 comentários foram rotulados de forma adequada de um total de 30. Entretanto, 10% (3 comentários) que na verdade são neutros, foram rotulados inadequadamente como negativo. Além disso, 6 comentários (20%) também foram rotulados inadequadamente, setes como sendo positivos.

Com base nessas análises, pode-se inferir que de um modo geral a ferramenta apresenta média de acerto aceitável, a qual se estabelece em torno de 72,2%. Considerando que pode ser utilizada por usuários que possuam milhares de comentários, ela pode servir de apoio para realizar mineração de opiniões. Contudo, rotulou de forma inadequada ~27,8% ou ~25 comentários.

Isso pode ter acontecido em virtude da alguns comentários não fazerem uso de adjetivos e/ou verbos que possuam conteúdo significativo - seja ofensivo ou não - e por conta disso, tais comentários são mais difíceis de rotular. Além disso, o classificador pode não ter identificado um determinado verbo ou adjetivo em alguma outra classe.

Na sentença "Esse aplicativo é ignorante.", por exemplo, caso o classificador não conheça o adjetivo "ignorante", ou seja, a base de dados utilizada durante o treinamento não tenha sido abrangente o suficiente, ele provavelmente não rotulará de forma adequada. Visto que o classificador não conhece este adjetivo, outro fator que contribui para esse problema é a quantidade de vezes que as palavras "aplicativo", "esse" e "e" aparecem nas classes. Este desconhecimento de alguns verbos/adjetivos - não constam na base de treinamento - é fator determinante para o resultado final gerado pelo classificador. Desse modo, se essas palavras aparecerem com maior frequência

sendo classificadas como comentários positivos, então a probabilidade de o *Multinomial Naive Bayes* atribuir a esta sentença a classe *positivo* é maior.

5 CONCLUSÃO

Esta Seção apresenta as considerações finais deste trabalho. Primeiramente é abordado as contribuições deste trabalho, e por último as perspectivas futuras são abordadas.

5.1 CONTRIBUIÇÕES

Este trabalho propôs o desenvolvimento de uma ferramenta *Web* que auxiliará organizações, figuras públicas, ou quaisquer pessoas que recebam milhares comentários em suas páginas no *Facebook*, a monitorar as opiniões expressas em tais comentários.

Com a finalidade de obter os dados advindos do *Facebook* a ferramenta proposta utiliza o *Django-Facebook*, que por sua vez é uma implementação da *Facebook Graph* API. Além disso, a ferramenta rotula os comentários em três classes com base no *Multinomial Naive Bayes* disponível no conjunto *NLTK-Trainer*, que por sua vez foi devidamente treinado.

A ferramenta proposta rotula os comentários disponíveis na (s) página (s) do *Facebook* que o usuário da ferramenta gerencia, bem como os comentários presentes na linha do tempo do usuário. Por fim, os resultados obtidos com essa ferramenta são satisfatórios, com uma taxa média de acerto de ~72%.

5.2 PERSPECTIVAS FUTURAS

Como sugestão para trabalhos futuros, pode-se ampliar a base de dados utilizada durante o treinamento, o que provavelmente irá aumentar a taxa de acerto desta ferramenta. Além disso, a ferramenta poderia classificar comentários em outras línguas, como por exemplo, inglês norte-americano, e consequentemente avaliar se o uso de *ngrams* de outras ordens (eg. *bigram*), afeta positivamente no desempenho dessa ferramenta.

Outra sugestão seria aumentar a quantidade de classes em diferentes níveis de polaridade (bom, muito bom, extremamente bom, excelente, etc.), o que possibilitaria um melhor discernimento sobre qual comentário deveria ser tratado com prioridade.

Além do mais, a ferramenta poderia ser integrada com outras redes sociais, como o *Google+* ou o *Twitter*. Isso poderia ser realizado sem muito esforço, em virtude do padrão de projeto *abstract factory* ter sido utilizado para realizar o acesso a camada de dados. Outra abordagem seria combinar diferentes classificadores com a finalidade de avaliar se essa estratégia impactaria positivamente na taxa de acerto.

REFERÊNCIAS

ARANHA, C. N. **Uma abordagem de pré-processamento automático para mineração de textos em português**: sob o enfoque da inteligência computacional. 2007. 144 f. Tese (Doutorado) - Curso de Engenharia Elétrica, Departamento de Engenharia Elétrica, Pontíficia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2007. Cap. 3. Disponível em: <http://www.maxwell.lambda.ele.puc-rio.br/Busca_etds.php?strSecao=resultado&nrSeq=10081@1>. Acesso em: 10 maio 2014.

BALAHUR, A. et al. **Sentiment Analysis in the News**. In: Proceedings of the 7th International Conference on Language Resources and Evaluation (LREC'2010). p. 2116. p. 20120. 2010. Disponível em: < http://arxiv.org/ftp/arxiv/papers/1309/1309.6202.pdf>. Acesso em 15 abr. 2015.

BECKER, K.; TUMITAN, D. **Introdução à Mineração de Opiniões**: Conceitos, Aplicações e Desafios. Simpósio Brasileiro de Banco de Dados, 2013. Disponível em: <http://sbbd2013.cin.ufpe.br/Proceedings/artigos/pdfs/sbbd_min_02.pdf>. Acesso em 05 maio de 2015.

BENEVENUTO, F; ALMEIDA, J. M; SILVA, A. S. **Atualizações em Informática 2011:** Coleta e Análise de Grandes Bases de Dados de Redes Sociais. Porto Alegre: Puc-rio, 2011. 407 p.

BROWN, R. **Django vs Flask vs Pyramid**: Choosing a Python Web Framework. 2015. Disponível em: <https://www.airpair.com/python/posts/django-flask-pyramid>. Acesso em: 10 abr. 2015.

BROWN, P. F. et al. **Class-based n-gram models of natural language**. Computational linguistics, v. 18, n. 4, p. 467-479, 1992.Disponível em: <http://delivery.acm.org/10.1145/180000/176316/p467-brown.pdf?ip=179.97.97.1&id=176316&acc=OPEN&key=4D4702B0C3E38B35%2E4D4702B0C3E38B35%2E4D4702B0C3E38B35%2E6D218144511F3437&CFID=672417161&CFTOKEN=13352685&__acm__=1431727467_d17a3f1d58c7aed822ffb883cbe40cb5>. Acesso em 04 mar. 2015.

CARRILHA JÚNIOR, J. R. **Desenvolvimento de uma Metodologia para Mineração de Textos.** 2007. 96 f. Dissertação (Mestrado) - Curso de Engenharia Elétrica, Departamento de Centro Técnico Científico, Pontíficia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2007. Cap. 3. Disponível em: <http://www.maxwell.lambda.ele.puc-rio.br/Busca_etds.php?strSecao=resultado&nrSeq=11675@1>. Acesso em: 15 mar. 2014.

CAVALIN, P. R.et al. **Real-time Sentiment Analysis in Social Media Streams**: The 2013 Confederation Cup Case. In: Encontro Nacional de Inteligência Artificial e

Computacional (ENIAC). IBM Reasearch. p. 538. p. 543. 2014. Disponível em: < http://www.lbd.dcc.ufmg.br/colecoes/eniac/2014/0094.pdf>. Acesso em: 15 abr. 2015.

COLLETA, L. F. S. et al. **Combining Classification and Agrupamento for Tweet Sentiment Analysis**. In: Brazilian Conference on Intelligent Systems (BRACIS). IEEE. p. 210. p. 215. 2014. Disponível em: < http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6984832 >. Acesso em: 15 abr. 2015.

CONNOLLY, D. **Hypertext Transfer Protocol -- HTTP/1.1**. 2004. Disponível em: < http://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html#sec9.3>. Acesso em 14 març. 2015.

COOLEY, R; MOBASHER, B; SRIVASTAVA, J. **Web mining**: Information and pattern discovery on the world wide Web. In: Tools with Artificial Intelligence, 1997. Proceedings, Ninth IEEE International Conference on. IEEE, 1997. p. 558-567. Disponível em: < http://ieeexplore.ieee.org/xpl/abstractAuthors.jsp?tp=&arnumber=632303&url=http%3A%2F%2Fieeexplore.ieee.org%2Fiel3%2F4999%2F13719%2F00632303>. Acesso em: 12 abr. 2014.

DJANGO, S.F. **The Web framework for perfectionists with deadlines**. 2015. Disponível em: <https://www.djangoproject.com/>. Acesso em: 10 abr. 2015.

DAVE, K.; LAWRENCE, S.; PENNOCK, D. M. **Mining the peanut gallery**: Opinion extraction and semantic classification of product reviews. In: Proceedings of the 12th international conference on World Wide Web. ACM, 2003. p. 519-528. Disponível em: <http://dl.acm.org/citation.cfm?id=775226>. Acesso em: 20 abr. 2015.

FACEBOOK. **APIs and SDKs**. 2015a. Disponível em: < https://developers.facebook.com/docs/apis-and-sdks>. Acesso em: 12 de mar. 2015.

FACEBOOK. **Graph API Overview**. 2015b. Disponível em: < https://developers.facebook.com/docs/graph-api/overview/>. Acesso em 12 mar. 2015.

FACEBOOK. **Access Tokens**. 2015c. Disponível em: <https://developers.facebook.com/docs/facebook-login/access-tokens>. Acesso em: 15 mar. 2015.

FACEBOOK. **Facebook Login for the Web with Javascrit SDK**. 2015d. Disponível em: <https://developers.facebook.com/docs/facebook-login/login-flow-for-Web/v2.3>. Acesso em 30 mar. 2015.

FACEBOOK. **Graph API Reference**. 2015e. Disponível em: <https://developers.facebook.com/docs/graph-api/reference>. Acesso em 03 maio. 2015.

GAMMA, E. et al. **Padrões de Projeto**: Soluções Reutilizáveis de Software Orientado a Objetos. Porto Alegre: Bookman, 2000. 346 p.

GIL, A. C. **Como elaborar projetos de pesquisa**. 5. ed. São Paulo: Atlas, 2010.

GO, A.; BHAYANI, R.; HUANG, L. **Twitter sentiment classification using distant supervision**. CS224N Project Report, Stanford, p. 1-12, 2009. Disponível em: <http://s3.eddieoz.com/docs/sentiment_analysis/Twitter_Sentiment_Classification_using_Distant_Supervision.pdf>. Acesso em: 16 abr. 2015.

GOOGLE. **Google Play**. 2015. Disponível em: < https://play.google.com/store?hl=pt-BR>. Acesso em: 02 abr. 2015.

GUEDES, G. T. A. **UML 2**: Uma abordagem prática. São Paulo: Novatec, 2009. 480 p.

HAMILTON, H. **Confusion Matrix**. 2012. Disponível em: <http://www2.cs.uregina.ca/~dbd/cs831/notes/confusion_matrix/confusion_matrix.html>. Aceeso em 12 abr. 2015.

HOLOVATY, A. **The Django Book**. 2007. Disponível em: <http://www.djangobook.com/en/2.0/index.html>. Acesso em: 21 abr. 2015.

HE, B. et al. **An effective statistical approach to blog post opinion retrieval**. In: Proceedings of the 17th ACM conference on Information and knowledge management. ACM, 2008. p. 1063-1072. Disponível em: < http://dl.acm.org/citation.cfm?id=1458223 >. Acesso em 15 abr. 2015.

HU, M.; LIU, B. **Mining and summarizing customer reviews**. In: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2004. p. 168-177. Disponível em: <http://www.aaai.org/Papers/AAAI/2004/AAAI04-119.pdf>. Acesso em: 15 abr. 2015.

IRFAN, R. et al. **A survey on text mining in social networks**. The Knowledge Engineering Review, v. 30, n. 02, p. 157-170, 2015.

KAO, A; POTEET, S. R. (Ed.). **Natural language processing and text mining**. Springer, 2007.

KELLY, R. **PyEnchant**: a spellchecking library for Python.2011. Disponível em: <http://pythonhosted.org/pyenchant/>. Acesso em: 16 abr. 2015.

KISS, J; **Facebook's 10th birthday**: from college dorm to 1.23 billion users. 2014. Disponível em: <http://www.theguardian.com/technology/2014/feb/04/facebook-10-years-mark-zuckerberg>. Acesso em: 16 abr. 2014.

LORENA, A. C.; CARVALHO, A. C. P. L. F. de C. **Uma introdução às Support Vector Machines**. Revista de Informática Teórica e Aplicada, Porto Alegre, v. 14, n. 2,

p.44-67, 2007. Quadrimestre. Disponível em: < http://www.seer.ufrgs.br/rita/article/view/rita_v14_n2_p43-67> Acessado em: 21 mar. 2014.

MCCALLUM, A.; NIGAM, K. **A comparison of event models for naive bayes text classification**. In: AAAI-98 workshop on learning for text categorization. 1998. p. 41-48. Disponível em: <http://www.cs.cmu.edu/~knigam/papers/multinomial-aaaiws98.pdf>. Acesso em: 07 maio. 2015.

MELVILLE, P. GRYC, W. LAWRENCE, R. D. **Sentiment analysis of blogs by combining lexical knowledge with text classification**. In: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2009. p. 1275-1284. Disponível em: < http://dl.acm.org/citation.cfm?id=1557156 >. Acesso em: 15 abr. 2015.

MUNDO EDUCAÇÃO.**Variância e desvio padrão**. 2015. Disponível: <http://www.mundoeducacao.com/matematica/variancia-desvio-padrao.htm>. Acesso em 14 abr. 2015.

NLTK, P. **Natural Language Toolkit.** 2015. Disponível em: <http://www.nltk.org/>. Acesso em: 10 abr. 2015.

PANG, B. LEE, L. VAITHYANATHAN, S. **Thumbs up?**: sentiment classification using machine learning techniques. In: Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10. Association for Computational Linguistics, 2002. p. 79-86. Disponível em: < http://dl.acm.org/citation.cfm?id=1118704>. Acesso em: 15 abr. 2015.

PERKINS, J. **Python 3 Text Processing with NLTK 3 CookBook.** 2. ed. Birmingham B3 2pb, Uk: Packet Publishing Ltda, 2014. 287 p.

PERKINS, J. **Welcome to NLTK-Trainer's documentation!**. 2011. Disponível em: <https://nltk-trainer.readthedocs.org/en/latest/>. Acessado em 12 abr. 2015.

PYTHON. **Dictionaries**. 2015a. Disponível em: <https://docs.python.org/2/tutorial/datastructures.html#dictionaries>. Acesso em 24 abr. 2015.

RASCHKA, S. **Naive Bayes and Text Classification**: I-Introduction and Theory. arXiv preprint arXiv:1410.5329, 2014. Disponível em: <http://arxiv.org/pdf/1410.5329.pdf>. Acesso em: 06 abr. 2015.

RUSSELL, S. J. NORVIG, P. **Artificial intelligence**: a modern approach. Englewood Cliffs: Prentice hall, 2009. 1152 p.

SAIF, H. et al. **On Stopwords, Filtering and Data Sparsity for Sentiment Analysis of Twitter**. Nineth International Conference on Language Resources and Evaluation, 2014, pp. 810–817. Disponível em: <http://www.lrec-conf.org/proceedings/lrec2014/pdf/292_Paper.pdf>. Acesso em 13 abr. 2015.

SANTOS, F. L. LADEIRA, M. **The role of text pre-processing in opinion mining**. In: Intelligent Systems (BRACIS), Brazillian Conference on. IEEE, 2014. p. 50-54. Disponível em: < http://ieeexplore.ieee.org/xpl/articleDetails.jsp?reload=true&arnumber=6984806>. Acesso em: 02 abr. 2015.

SANTOS, L. M. **Protótipo para Mineração de Opinião em Redes Sociais**: Estudos de Casos Selecionados Usando o Twitter. 2010. 103 f. TCC (Graduação) - Curso de Ciência da Computação, Departamento de Ciência da Computação, Universidade Federal de Lavras, Lavras, 2010. Disponível em: <http://www.bcc.ufla.br/wp-content/uploads/2013/2010/LeandroMatioli.pdf>. Acesso em: 14 maio 2014.

SCHELLENBACH, T. **Open Facebook API**. 2011. Disponível em: <http://django-facebook.readthedocs.org/en/latest/open_facebook/api.html>. Acesso em: 03 maio. 2015.

SCHELLENBACH, T. **Django Facebook By Thierry Schellenbach**. 2015. Disponível em: <https://github.com/tschellenbach/Django-facebook> Acesso em: 29 mar. 2015.

SCIKIT-LEARN. **User guide**: contents - scikit-learn 0.16.1. 2014a. Disponível em: <http://scikit-learn.org/stable/user_guide.html>. Acesso em: 04 abr. 2015.

SCIKIT-LEARN. **Naive Bayes**. 2014b. Disponível em: <http://scikit-learn.org/stable/modules/naive_bayes.html>. Acesso em: 05 maio. 2015.

SOARES, F. de A. **Mineração de Textos na Coleta Inteligente de Dados na Web**. 2008. 120 f. Tese (Doutorado) - Curso de Engenharia Elétrica, Departamento de Centro Técnico Científico, Pontíficia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2008. Cap. 2. Disponível em: <http://www.maxwell.lambda.ele.puc-rio.br/Busca_etds.php?strSecao=resultado&nrSeq=13212@1>. Acesso em: 07 abr. 2014.

STATISTA. **Leading social networks worldwide as of March 2015, ranked by number of active users (in millions)**. 2015. Disponível em: <http://www.statista.com/statistics/272014/global-social-networks-ranked-by-number-of-users/>. Acesso em 06 abr. 2015.

TABOADA, M. et al. **Lexicon-Based Methods for Sentiment Analysis**. In: Association for Computational Linguistics. MIT Press. v. 37, n. 2, p. 267-307, 2011.Disponível em: <http://www.mitpressjournals.org/doi/abs/10.1162/COLI_a_00049>. Acesso em: 15 abr. 2015.

WITTEN, I. et al. **Text mining**: A new frontier for lossless compression. In: Data Compression Conference, 1999. Proceedings. DCC'99. IEEE, 1999. p. 198-207. Disponível em: < http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=755669>. Acesso em: 20 marc. 2014.

W3C. **HTTP - Hypertext Transfer Protocol**. 2003. Disponível em: <http://www.w3.org/Protocols/>. Acesso em: 12 abr. 2015.

APÊNDICE

FERRAMENTA PARA DETECTAR COMENTÁRIOS OFENSIVOS NO FACEBOOK

Felipe Appio[1]
[1]Universidade do Estado de Santa Catarina
felipexw@gmail.com

Resumo

Este trabalho apresenta a proposta de uma ferramenta *Web* que realizará mineração de opiniões - positivo, negativo ou neutro - no *Facebook*. Pode-se conceituar rede social como sendo um grupo de pessoas que interagem por intermédio de qualquer mídia de comunicação. Análise de sentimento, mineração de opiniões ou análise de subjetividade é um campo de estudo dentro de inteligência artificial que busca extrair informações subjetivas como opiniões, sentimentos, escritos em linguagem natural. Uma das técnicas bem difundidas em análise de sentimento é a técnica *Naive Bayes*, que é uma técnica de aprendizado de máquina supervisionado presente em uma biblioteca de mineração de texto, nomeada de *NLTK-Trainer*. A importância monitorar tais opiniões reside no fato de que organizações que fazem uso de redes socais necessitam monitorar seus produtos/serviços visando traçar estratégias com base nesses sentimentos. Para extrair dados do *Facebook* foi utilizado uma biblioteca nomeada de *Django-Facebook*, que é uma implementação da *Facebook Graph* API. Por fim, com a finalidade de avaliar a ferramenta proposta, foram coletados 90 comentários igualmente distribuídos em positivos, negativos ou neutros. Ao final observou-se que a ferramenta obteve uma taxa média de acerto que ficou em torno de 72%.

Palavras-chave: Análise de Sentimento. Multinomial Naive Bayes. Facebook.

Abstract

This paper presents the proposal of a Web tool that hold mining opinions - positive, negative or neutral - the Facebook. You can conceptualize social network as a group of people who interact through any medium of communication. Sentiment analysis, mining opinions or subjectivity of analysis is a field of study in artificial intelligence that seeks to extract subjective information such as opinions, feelings, written in natural language. One well known techniques in the sense of analysis is Naive Bayes technique, which is a machine learning technique supervised present in a text mining library named NLTK Trainer. The importance of monitoring these views lies in the fact that organizations that make use of social networks need to monitor your products / services in order to develop strategies based on these feelings. To extract data from Facebook was used a named library Django-Facebook, which is an implementation of the Facebook Graph API. Finally, in order to evaluate the proposed tool, they collected 90 comments also distributed in positive, negative or neutral. At the end it was observed that the tool scores a hit rate that was around 72%.

Keywords: *Traffic. Highways. Security. Google Maps.*

1. Introdução

Rede social *online* pode ser conceituada como um grupo de pessoas que interagem por meio de qualquer mídia de comunicação. Pode-se definir, também, como sendo um sistema *Web* que fornece a possibilidade de construir perfis públicos; organizar uma lista de usuários com o qual é possível compartilhar conexões; visualizar e percorrer suas listas de conexões (BENEVENUTO; ALMEIDA e SILVA, 2011). Com base nessas definições há várias redes sociais *online* disponíveis como, por exemplo, o *Facebook*[43], *Twitter* [44], *Google* +[45].

Ao final de 2013 a rede social *online Facebook* atingiu 1.23 bilhões de usuários (KISS, 2014), o que implica na geração massiva de conteúdos nesta rede. No *Facebook* pode-se encontrar páginas de artistas como Luciano Huck[46], de empresas como Coca-Cola[47], e de figuras públicas como Barack Obama[48]. Tais páginas podem ter muitos seguidores, como por exemplo a página da Coca-Cola, que no dia 05 de abril de 2015 possuía mais de 89 milhões de seguidores. De modo geral, quanto maior for o número de seguidores de uma página, maior é a repercussão dos comentários textuais publicados na mesma. Isso pode ser um problema se o conteúdo publicado contiver informações negativas, pois conforme Becker e Tumitan (2013), as organizações baseiam suas estratégias de negócio com base na opinião dos clientes sobre um determinado produto ou serviço.

Desse modo, em páginas com milhares de seguidores como a página da Coca-Cola, torna-se difícil monitorar os comentários textuais compartilhados nas postagens dessas páginas, principalmente comentários negativos. Visto que as organizações ou indivíduos podem monitorar a opinião dos seguidores (BECKER; TUMITANM, 2013), torna-se evidente que comentários com semântica negativa (ruim) devem ser tratados de forma prioritária, sendo necessário haver algum modo de filtrá-los.

O processo automatizado de extrair informações textuais com um propósito específico de textos não estruturados é conhecido como mineração de textos – que vem do inglês *text mining* (WITTEN et al. 1999). Ainda dentro de mineração de texto, há uma área chamada de análise de sentimento que objetiva classificar opiniões expressas em textos (COLLETA et al, 2014) - geralmente em positivo (bom), negativo (ruim) ou neutro (objetivo) (BECKER; TUMITAN, 2013). Essa classificação pode ocorrer por intermédio de duas abordagens: utilizando algoritmos baseados em aprendizagem de máquina, ou através de ontologias (IRFAN, 2015). Este trabalho realiza a classificação de textos por intermédio de um classificador baseado em aprendizado de máquina.

Aprendizagem de máquina - termo que vem do inglês, *Machine learning* (ML) - é uma área de inteligência artificial que estuda e desenvolve algoritmos que aprendem a partir de observações. Dentro de ML há aprendizagem supervisionada – que será objeto desse trabalho, pois será utilizado a técnica de aprendizagem estatística *Multinomial Naive Bayes* para classificação textual (RUSSEL; NORVIG, 2009).

---

[43] Disponível em: <https://www.facebook.com/?_rdr>.
[44] Disponível em: <https://twitter.com/?lang=pt>.
[45] Disponível em: <https://plus.google.com/?hl=pt_BR>.
[46] Disponível em: < https://www.facebook.com/LucianoHuck?fref=ts>.
[47] Disponível em: < https://www.facebook.com/cocacolabr?fref=ts>.
[48] Disponível em: < https://www.facebook.com/barackobama?fref=ts>.

Este trabalho propõe uma ferramenta cuja finalidade é apontar postagens textuais (comentários nas publicações) ofensivas, publicadas na linha do tempo de um perfil no *Facebook* (usuário da ferramenta). Futuramente essa ferramenta poderá ser integrada com outras redes sociais. A ferramenta classificará sentenças escritas em português brasileiro em três classes: positivo; negativo ou neutro. Posteriormente será efetuado um experimento com a finalidade de validar a ferramenta.

Este trabalho se justifica pelo fato de colaborar com usuários que possuem milhares de publicações (advindas de outros usuários) no seu perfil da rede social *Facebook*, que não conseguem monitorar todas as postagens textuais em virtude do grande volume.

2. Ferramenta para comentários ofensivos no Facebook

Nesse capítulo será abordada a fundamentação teórica necessária para o desenvolvimento deste trabalho.

2. Web Mining

De acordo com Cooley; Mobasher e Srivastava (1997), *Web mining* pode ser definido como sendo a análise e descoberta de informações advindas da *World Wide Web*. Conforme Carrilha Júnior (2007) há três abordagens para minerar dados no ambiente *Web*:

a) *Web Content Mining* – ou mineração do conteúdo *Web* - preocupa-se em analisar o conteúdo dentro das páginas HTML, isto é, textos; imagens; etc.
b) *Web Usage Mining* – preocupa-se em identificar padrões de acesso dos usuários, por exemplo, do ponto de vista do *link*, quais partes dos sites precisam ser alteradas com base nos acessos dos usuários. Por último,
c) *Web Structure Mining* – responsável por estudar os relacionamentos entre os *hiperlinks*. Nesse sentido, avalia-se a quantidade de ligações que uma página possui com outra, podendo-se inferir qual página possui mais referências, isto é, as mais divulgadas.

Esse trabalho encontra-se dentro da subárea *Web Content Mining*, visto que o conteúdo a ser processado está presente no ambiente *Web*, mais precisamente na rede social *Facebook.*

2.1 MINERAÇÃO DE TEXTO E ANÁLISE DE SENTIMENTO

Conforme Witten et al. (1999), mineração de textos - *text mining* - é o processo de analisar textos não estruturados de forma automática, procurando por padrões, de modo a extrair informações com um propósito específico. Essas informações podem ser opiniões ou emoções expressas em textos. Para tal, há uma subárea nomeada de análise de sentimentos, também conhecida como mineração de opinião ou análise de subjetividade que, busca justamente classificá-las (BECKER; TUMITAN, 2013).

Há três maneiras de fazer isso, utilizando algoritmos de aprendizagem de máquina (supervisionada ou não), ou técnicas de *Natural Language Processing*[49] (NLP) (TABOADA et al, 2011). Além disso, é possível combinar técnicas de pré-processamento (técnicas de NLP) com técnicas de aprendizagem de máquina.

[49] Termo que vem do inglês e pode ser definido como processamento de linguagem natural.

A área de NLP trata da extração de informações como “por que”, “quando”, “como”, "o que" expressa em textos. Suas técnicas fazem uso de conceitos de linguística, gramática e pragmática (KAO; POTEET, 2007).

Conforme Aranha (2007), as etapas de análise de sentimento acontecem da seguinte maneira:

a) A coleta de dados textuais é a primeira etapa da mineração de textos, cujos dados são obtidos de diversas fontes, como por exemplo redes sociais;
b) Em seguida vem a etapa de pré-processamento – técnicas de processamento de linguagem natural - onde os dados sofrem tratamento em sua estrutura e organização;
c) O desenvolvimento de cálculos, inferências e extração de conhecimento sobre esses dados é chamado de mineração, que vem logo após a indexação;
d) A última etapa é a análise dos dados, isto é, a interpretação dos dados. Pode ser feita com o uso de medidas quantitativas ou por um especialista no assunto.

Essas fases de mineração de dados são ilustradas na Figura 1Figura 1 e explanadas nas seções a seguir.

Figura 28 - Etapas de mineração de dados, produção própria do autor.

2.1.1 Coleta de Dados e Pré-Processamento

Conforme Carrilha Júnior (2007), por coleta entende-se o ato de buscar e recuperar dados com o intuito de formar uma base textual da qual se pretende extrair informações. Há vários meios de se obter esses dados textuais. Arquivos encontrados em discos rígidos; dados advindos da *Web* ou dados advindos dos mais diversos bancos de dados.

Conforme Carrilha Júnior (2007), o pré-processamento é a etapa que se segue logo após a coleta dos dados. O pré-processamento é necessário para realizar a mineração de dados, pois ele aumenta a qualidade dos dados em questão. Embora para outros autores, como para Santos e Ladeira (2014), o pré-processamento nem sempre tem uma melhora significativa dentro de mineração de textos.

Para Irfan et al. (2015) há dois métodos básicos de pré-processamento: a seleção de características e a extração de características. Esses métodos serão detalhados na seção a seguir.

2.1.1.1 Extração das Características

Conforme Scikit-learn (2014a), o processo de extração de características - *feature extraction* (FE) - é o processo de transformar dados arbitrários, como imagens ou textos, em dados numéricos que possam ser utilizadas pelos algoritmos de aprendizagem de máquina.

O modelo *bag-of-words*[50] é um dos modelos mais simples utilizados em extração de características textuais, pois não é necessário saber a ordem de uma determinada palavra

[50] Termo que vem do inglês e pode ser traduzido para saco de palavras.

no texto, basta saber que ela está presente e a quantidade de vezes que ela aparece (PERKINS, 2014). Para Raschka (2014), o processo de extrair um vocabulário V (todas as palavras que aparecem, desconsiderando as repetições) de 1 ou mais documentos de textos D_i e contar suas respectivas frequências é chamado de vetorização. Desse modo, o modelo *bag-of-words* de um documento $D_1 = \text{Cada estado tem suas próprias leis}$ e outro documento $D_2 = \text{Cada país tem suas próprias culturas}$ será mapeado conforme Quadro 1:

	Cada	estado	tem	Suas	próprias	leis	país	culturas
x_{D1}	1	1	1	1	1	1	0	0
x_{D2}	1	0	1	1	1	0	1	1
$\sum$	2	1	2	2	2	1	1	1

Quadro 15 - Representação *bag-of-words* dos documentos D_1 e D_2.

De acordo com Raschka (2014), o modelo *bag-of-words* pode conter valores binários - representando se a palavra está presente no documento ou não - ou as respectivas frequências das palavras. O que determina esses valores é o classificador utilizado. Em virtude desse trabalho utilizar o *Multinomial Naive Bayes* (vide Seção 2.5.2), será utilizado o modelo *bag-of-words* contendo as frequências das palavras.

Segundo Irfan et al. (2015), é possível categorizar a extração de características em análise morfológica, análise sintática e análise semântica. O método de análise morfológica consiste em *tokenization*[51]e redução do léxico (CARRILHA JÚNIOR, 2007). Em virtude desse trabalho utilizar somente a técnica de análise morfológica (vide Seção 2.5) os outros métodos não serão abordados.

Tokenization é a primeira etapa a ser realizada durante o pré-processamento, a sua finalidade é extrair as unidades mínimas de um texto. Um *token*[52] geralmente é uma palavra que pode estar relacionado com outras palavras, caracteres ou símbolos (CARRILHA JÚNIOR, 2007). Desse modo, após se aplicar *tokenization* na sentença: "Chorão foi um dos maiores cantores brasileiros.", obter-se-ia como resultado: [Chorão] [foi] [um] [dos] [maiores] [cantores] [brasileiros] [.], neste caso cada *token* seria representado por um elemento entre colchetes.

Apesar de esse processo parecer relativamente simples, é necessário ter cuidado ao realizá-lo, pois alguns termos, quando separados mudam completamente o sentido da frase. O ponto final, por exemplo, além de finalizar a sentença, pode ser utilizado para abreviar números. Para resolver esse problema podem-se identificar símbolos da internet, abreviações, etc. e combiná-las em um único *token* de modo a conservar o sentido das palavras (CARRILHA JÚNIOR, 2007). A palavra "guarda-roupa", por exemplo, poderia ser agrupada em um único *token*: [guarda-roupa].

Feito a *tokenization* é necessário fazer a redução do léxico, pois conforme foi citado em Carrilha Júnior (2007), cada palavra vira um *token* que é mapeado para uma dimensão, gerando assim um vasto número de dimensões, dependendo do texto. Para amenizar esse problema, então, deve-se diminuir a quantidade de palavras em um texto, atentando para não prejudicar a semântica das sentenças (CARRILHA JÚNIOR, 2007).

---

51 Não há tradução para português.

52 Termo sem tradução para português.

Há várias abordagens para amenizar esse problema; contudo, esse trabalho abordará somente três delas: remoção de *stopwords*[53]; normalização; seleção de características (CARRILHA JÚNIOR, 2007).

Conforme Carrilha Júnior (2007), a etapa de remoção de *stopwords* consiste em remover artigos, preposições, conjunções, pontuação e pronomes de uma língua, pois geralmente não possuem um significado relevante para o contexto de mineração de textos. Essas palavras são classificadas como *stopwords* – termo que vem do inglês e não possui tradução para português.

Um conjunto de *stopwords* compõe uma *stoplist*[54], isto é, uma lista com os *tokens* pouco relevantes. As *stopslits* podem ser definidas por um especialista no assunto, ou por intermédio de alguma técnica de NLP, como por exemplo a técnica de medida de entropia do termo, ou a medida de divergência (SAIF, 2014). Conforme Saif (2014), não há diferença significativa entre o a remoção de *stoplists* estáticas ou dinâmicas. Portanto esse trabalho utilizará a abordagem estática, pois conforme Saif (2014), é computacionalmente menos custoso utilizar *stoplists* estáticas. Em virtude disso, só será explanado o mecanismo de remoção de *stoplist* estática (Seção 4.2.5.2).

De acordo com Carrilha Júnior (2007), normalização é a etapa de identificação e agrupamento de palavras que possuem um relacionamento semântico, como por exemplo, os sinônimos. A normalização pode acontecer pelos processos de:

a) *Stemming* – é o processo de converter palavras para sua forma original. Desse modo, termos no plural, verbos em diferentes tempos verbais tornam-se palavras no infinitivo. Segundo Raschka (2014), esse processo pode gerar palavras que não existem.
b) *Lemmatization* – é o processo de converter palavras que foram derivadas de outras palavras em sua respectiva palavra de origem, respeitando a gramática. Pode-se citar como exemplo "as palavras música, músicas e musical compartilham a mesma palavra de origem, música." (SANTOS, 2010, p. 30).

Durante a etapa de normalização, pode-se utilizar tanto o método *lemmatization*, quanto *stemming*, ou ambos. *Lemmatization* é computacionalmente mais custoso do que *stemming*, embora não gere um impacto significativo (em relação a *stemming*) no desempenho de classificadores textuais. Além disso, a combinação dos dois métodos não tem grande impacto no desempenho de classificação textual (RASCHKA, 2014). Portanto, durante a etapa de pré-processamento foi escolhido utilizar somente *stemming* neste trabalho.

Segundo Perkins (2014) uma estratégia utilizada para melhorar o processo de extração de características é utilizar *collocations*[55]. *Collocations* são duas ou mais palavras que tendem a aparecer juntas, como por exemplo, 'Estados Unidos' (PERKINS, 2014). O termo *ngrams*[56] descreve uma colocação de ordem n.

Desse modo, utilizando *bigrams* (*ngram* de ordem 2), a sentença 'Estados Unidos' após o processo de *tokenization* torna-se ['Estados Unidos'] e não ['Estados'] ['Unidos'] (RASCHKA, 2014). Para Pang et al. (2002) e Dave; Lawrence e Pennock (2003), em análise de sentimento *unigrams*, *bigrams* e *trigrams* aumentam o desempenho do classificador. Portanto, esse trabalho utiliza somente *ngram* de rodem 1 até 3.

---

53 Sem tradução para português.
54 Sem tradução para português.
55 Pode ser traduzido para português como colocações (tradução do autor).
56 Sem tradução para português.

Para Brown (1992), *ngrams* são gerado com base em algoritmos de modelagem de linguagem probabilísticos. Em virtude desse trabalho considerar somente *ngrams* de ordem 1 até 3 - conforme citado no parágrafo acima - , não será entrado em detalhes em relação a geração de *ngrams*. Mais informações sobre *ngrams* podem ser obtidas em Brown (1992).

2.1.2 Mineração e Interpretação

Esta etapa é responsável por extrair novos conhecimentos partir de técnicas de aprendizagem de máquina (ML). Dentro de mineração de dados há a mineração de textos, que tem por objetivo extrair conhecimentos oriundos de bases textuais gravados em linguagem natural. Inicialmente deve-se decidir que tipo de informação deseja-se extrair da massa textual. Algoritmos de classificação e agrupamento podem ser aplicados nessa etapa (SOARES, 2008). Como o escopo desse trabalho restringe-se a classificação, não será abordada a questão de agrupamento.

A interpretação, análise ou também nomeada de pós-processamento diz respeito ao tratamento do conhecimento obtido na mineração. Pode ser realizada por um especialista no assunto, bem como por métodos quantitativos e qualitativos (SOARES, 2008).

Essa etapa é responsável por efetivamente visualizar e interpretar os dados obtidos até a mineração. Para realizar a interpretação dos dados, há várias métricas na literatura (SOARES, 2008).

2.2 Facebook Graph API e Django-Facebook

O *Facebook* fornece uma API baseada em *Hyper Text Transfer Protocol*[57] (HTTP) chamada de *Facebook Graph* API, cujo objetivo é fornecer mecanismos para ler e escrever dados na linha do tempo de usuários da rede social *Facebook* (FACEBOOK, 2015a). Em função disso, qualquer linguagem de programação que tenha uma biblioteca HTTP pode manipular os dados através da *Graph* API. Segundo Facebook (2015a), há várias bibliotecas para a *Graph* API. Entretanto, esse trabalho utiliza a biblioteca *Django-Facebook*.

Conforme Schellenbach (2015), *Django-Facebook*[58] é uma biblioteca escrita em *Python*[59] que implementa a *Facebook Graph* API e é compatível com o *framework Django* (SCHELLENBACH, 2015). Segundo Brown (2015), O *Django* começou a ser desenvolvido em meados 2006, e sua versão mais atual é a 1.7 - que será utilizada nesse trabalho.

Foi optado em utilizar o *Django-Facebook* para manipular os dados advindos do *Facebook*, pois a biblioteca escolhida para fazer classificação textual está escrita em *Python* (será explanada na Seção 2.5) e a única implementação da *Facebook Graph* API disponível em *Python* é a *Django-Facebook* (FACEBOOK, 2015a).

2.3 NLTK-Trainer e Aprendizagem de Máquina

---

[57] Um protocolo projetado para permitir comunicação entre cliente e servidor por intermédio de requisições e respostas entre o cliente e o servidor (W3C, 2015).
[58] Disponível em: < https://github.com/tschellenbach/Django-facebook>.
[59] Disponível em: < https://www.python.org/>.

ML é uma área da inteligência artificial dedicada ao desenvolvimento de algoritmos que extraem conceitos (conhecimento) a partir de uma amostra de dados de entrada. Um dos conceitos que regem o ML é o conceito de indução, onde a partir de um conjunto de exemplos gera-se uma função que se aproxima da função que gerou os exemplos. Há dois tipos de aprendizagem de máquina indutivo, o primeiro é a aprendizagem supervisionada – o qual será objeto desse trabalho – e o outro é chamado de aprendizagem não supervisionada. (RUSSEL; NORVIG, 2009).

Conforme Russel e Norvig (2009), aprendizagem de máquina não supervisionada trata da extração de conhecimento a partir de um conjunto de entrada, sem haver um conjunto de saída para o mesmo. Por outro lado, quando há conhecimento do conjunto de saída para seu respectivo conjunto de entrada, entende-se por aprendizagem supervisionada, isto é, a partir de um conjunto X de entrada e outro conjunto Y de saída, gera-se uma função hipótese que se aproxima da função geradora dos conjuntos X e Y. Esse processo de aprendizagem indutivo é chamado de treinamento.

De acordo com Russel e Norvig (2009), geralmente os algoritmos de aprendizagem de máquina são utilizados para classificar dados. Por classificação entende-se como o ato de determinar a qual ordem ou rótulo um dado pertence.

Segundo Hamilton (2012), com a finalidade de avaliar diferentes classificadores, há 4 métricas na literatura: acurácia (*accuracy*), precisão (*precision*), *recall* e *média* harmônica (*F-measure*). Entretanto este trabalho utiliza a acurácia, pois conforme Russel e Norvig (2009), a acurácia determina a capacidade de generalização de um determinado classificador. Desse modo, quanto maior for a acurácia, maior será a probabilidade desse classificador rotular adequadamente.

Pelo fato de haver um grande conjunto de hipóteses possíveis, pode acontecer de a técnica classificar corretamente apenas os dados fornecidos como entrada, e não os outros dados, o que é chamado de superadaptação (RUSSEL; NORVIG, 2009). De acordo com Russel e Norvig (2009), a validação cruzada é uma técnica estatística que diminui a superadaptação e pode ser aplicada em qualquer algoritmo de classificação. O pressuposto básico da validação cruzada "é estimar até que ponto cada hipótese irá prever dados não vistos" (RUSSEL; NORVIG, 2009). Um dos métodos de validação cruzada é conhecida como *k-fold*[60], que consiste em estratificar (dividir) os dados em k subconjuntos. Cada subconjunto possui $1/k$ dos dados para serem utilizados como testes, variando de acordo com cada subconjunto (RUSSEL; NORVIG, 2009).

2.3.1 Naive Bayes

Conforme Russel e Norvig (2009), *Naive Bayes* é um método de aprendizado estatístico (supervisionado) que tem sido extensivamente estudado desde a década de 1950. Possui o nome *Naive Bayes* pois o primeiro termo, *naive* vem do inglês e significa ingênuo, e o segundo pelo fato de ser uma aplicação do teorema de *Bayes* (RUSSEL; NORVIG, 2009). Informações adicionais sobre esse teorema podem ser obtidas em Russel e Norvig (2009).

Esse modelo de aprendizado é considerado ingênuo pois "supõe que os atributos são condicionalmente independentes uns dos outros, dada a classe" (RUSSEL; NORVIG, 2009, p. 696). Além disso, outra suposição que o *Naive Bayes* faz é que os dados são

[60] Sem tradução adequada para o português.

independente e identicamente distribuídos, ou seja, a probabilidade de um evento x acontecer não afeta a probabilidade de um evento y ocorrer (RASCHKA, 2014).

Essas suposições podem ser consideradas ingênuas porque alguns eventos são dependentes um dos outros, como por exemplo, séries temporais. Por outro lado, alguns outros eventos são condicionalmente independentes, como por exemplo, ao jogar uma moeda para cima 3 vezes seguidas, pode-se observar que na primeira jogada a moeda terá 50% de chance de cair com a coroa virada para cima, na segunda jogada a chance será a mesma e assim por diante (RASCHKA, 2014).

Conforme citado acima, é um método de aprendizado estatístico, e portanto, pode ser escrito conforme uma fórmula matemática:

$$\hat{y} = argmax_y \, P(y) \prod_{i=1}^{n} P(x_i \mid y) \quad (5)$$

Onde $P(y)$ é a probabilidade de ser uma classe y em relação as outras classes. $P(x_i \mid y)$ é a probabilidade condicional de o atributo x_i dado a classe y. $argmax$ é a função que maximiza a distribuição das probabilidades (SCIKIT-LEARN, 2014b).

Em outras palavras, o cálculo da hipótese mais provável (valor que maximiza a hipótese, também conhecido como máximo à priori) de um conjunto de atributos x pertencer a uma classe y é dada pela ponderação das probabilidades condicionais de x_i dado a classe y (SCIKIT-LEARN, 2014b). Para Russel e Norvig (2009 p.691) "a aprendizagem *bayesiana* simplesmente calcula a probabilidade de cada hipótese, considerando-se os dados e faz previsões de acordo com ela."

Conforme Raschka (2014), há diferentes implementações para o modelo *Naive Bayes*. Em função desse trabalho utilizar a implementação conhecida como *Multinomial Naive Bayes* - que será abordada na próxima Seção -, as outras técnicas não serão discutidas. Mais informações podem ser obtidas em Russel e Norvig (2009) e em Raschka (2014).

3. Resultados e Experimentais

Este trabalho teve como resultado o desenvolvimento de uma ferramenta *Web* capaz de realizar análise de sentimento em postagens textuais na rede social *Facebook*. Com a finalidade de avaliar a ferramenta foram selecionados 150 comentários textuais do perfil disponíveis no perfil de Juvenal Antena[61]. Em seguida, destes 150 comentários, foram rotulados empiricamente 90 comentários em positivo, negativo ou outro.

Optou-se em utilizar esse perfil por questões éticas - o perfil foi criado justamente para realizar os testes com a ferramenta proposta -. Além disso, em virtude da ferramenta ter sido treinada utilizando uma base de comentários textuais advindos do *Google Play* (vide Seção 4.2.5.1), os comentários publicados no perfil de Juvenal Antena foram extraídos do *Google Play*. Cabe salientar que estes comentários não são os mesmos utilizados durante o treinamento, pertencem apenas ao mesmo contexto (*Google Play*).

Após a execução da ferramenta foram obtidos os dados que demonstram os resultados gerados pela ferramenta. Desse modo, para cada uma das três classes obteve-se um percentual de acerto e erro. O Quadro 14 ilustra esses dados, onde há uma coluna nomeada de classe real representando as classes que foram rotuladas empiricamente, e

[61] Disponível em: < https://www.facebook.com/profile.php?id=100009652808830>.

outra coluna nomeada de classe prevista que representa as classes que foram geradas pela ferramenta.

		Classe Prevista		
		Positivo	Negativo	Neutro
Classe Real	Positivo	~83,3%	~3,3%	~13.3%
	Negativo	~13,3%	~63,3%	~23,3%
	Neutro	20%	10%	70%
	Erro	~16,6%	~36,6%	~30%
Taxa de acerto		**~72,2%**		
Taxa de erro		**~27,8%**		

Quadro 2 - Resultados gerados pela ferramenta para as classes positivo, negativo e neutro.

Conforme pode-se observar no Quadro 14, o percentual de acertos para a classe positiva foi de ~83,3%, ou seja, 25 comentários de um total de 30 foram rotulados adequadamente como sendo positivos. O restante, ~16,6%, foram rotulados inadequadamente como sendo ~3,33% negativo e ~13,33% positivo.

Para a classe negativa, é possível observar uma taxa de acerto não muito satisfatória (~63,3%), se comparada com a taxa de acerto para a classe positiva. Além disso, aproximadamente 37% (11 comentários) foram rotulados de forma inadequada, sendo que 7 comentários negativos (~23,3%) foram rotulados como neutros e os 4 restantes (~13,3%) como sendo positivos.

Por último, pode-se observar no Quadro 14, que a taxa de acerto para a classe neutro foi de 70%, ou seja, 21 comentários foram rotulados de forma adequada de um total de 30. Entretanto, 10% (3 comentários) que na verdade são neutros, foram rotulados inadequadamente como negativo. Além disso, 6 comentários (20%) também foram rotulados inadequadamente, setes como sendo positivos.

Com base nessas análises, pode-se inferir que de um modo geral a ferramenta apresenta média de acerto aceitável, a qual se estabelece em torno de 72,2%. Considerando que pode ser utilizada por usuários que possuam milhares de comentários, ela pode servir de apoio para realizar mineração de opiniões. Contudo, rotulou de forma inadequada ~27,8% ou ~25 comentários.

Isso pode ter acontecido em virtude da alguns comentários não fazerem uso de adjetivos e/ou verbos que possuam conteúdo significativo - seja ofensivo ou não - e por conta disso, tais comentários são mais difíceis de rotular. Além disso, o classificador pode não ter identificado um determinado verbo ou adjetivo em alguma outra classe. Na sentença "Esse aplicativo é ignorante.", por exemplo, caso o classificador não conheça o adjetivo "ignorante", ou seja, a base de dados utilizada durante o treinamento não tenha sido abrangente o suficiente, ele provavelmente não rotulará de forma adequada. Visto que o classificador não conhece este adjetivo, outro fator que contribui para esse problema é a quantidade de vezes que as palavras "aplicativo", "esse" e "e" aparecem nas classes. Este desconhecimento de alguns verbos/adjetivos - não constam na base de treinamento - é fator determinante para o resultado final gerado pelo classificador. Desse modo, se essas palavras aparecerem com maior frequência sendo classificadas como comentários positivos, então a probabilidade de o *Multinomial Naive Bayes* atribuir a esta sentença a classe *positivo* é maior.

3. Contribuições e Perspectivas Futuras

Este trabalho propôs o desenvolvimento de uma ferramenta *Web* que auxiliará organizações, figuras públicas, ou quaisquer pessoas que recebam milhares comentários em suas páginas no *Facebook*, a monitorar as opiniões expressas em tais comentários.

Com a finalidade de obter os dados advindos do *Facebook* a ferramenta proposta utiliza o *Django-Facebook*, que por sua vez é uma implementação da *Facebook Graph* API. Além disso, a ferramenta rotula os comentários em três classes com base no *Multinomial Naive Bayes* disponível no conjunto *NLTK-Trainer*, que por sua vez foi devidamente treinado.

A ferramenta proposta rotula os comentários disponíveis na (s) página (s) do *Facebook* que o usuário da ferramenta gerencia, bem como os comentários presentes na linha do tempo do usuário. Por fim, os resultados obtidos com essa ferramenta são satisfatórios, com uma taxa média de acerto de ~72%.

Como sugestão para trabalhos futuros, pode-se ampliar a base de dados utilizada durante o treinamento, o que provavelmente irá aumentar a taxa de acerto desta ferramenta. Além disso, a ferramenta poderia classificar comentários em outras línguas, como por exemplo, inglês norte-americano, e consequentemente avaliar se o uso de *ngrams* de outras ordens (eg. *bigram*), afeta positivamente no desempenho dessa ferramenta.

Outra sugestão seria aumentar a quantidade de classes em diferentes níveis de polaridade (bom, muito bom, extremamente bom, excelente, etc.), o que possibilitaria um melhor discernimento sobre qual comentário deveria ser tratado com prioridade.

Além do mais, a ferramenta poderia ser integrada com outras redes sociais, como o *Google+* ou o *Twitter*. Isso poderia ser realizado sem muito esforço, em virtude do padrão de projeto *abstract factory* ter sido utilizado para realizar o acesso a camada de dados. Outra abordagem seria combinar diferentes classificadores com a finalidade de avaliar se essa estratégia impactaria positivamente na taxa de acerto.

REFERÊNCIAS

ARANHA, C. N. **Uma abordagem de pré-processamento automático para mineração de textos em português**: sob o enfoque da inteligência computacional. 2007. 144 f. Tese (Doutorado) - Curso de Engenharia Elétrica, Departamento de Engenharia Elétrica, Pontíficia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2007. Cap. 3. Disponível em: <http://www.maxwell.lambda.ele.puc-rio.br/Busca_etds.php?strSecao=resultado&nrSeq=10081@1>. Acesso em: 10 maio 2014.

BECKER, K.; TUMITAN, D. **Introdução à Mineração de Opiniões**: Conceitos, Aplicações e Desafios. Simpósio Brasileiro de Banco de Dados, 2013. Disponível em: <http://sbbd2013.cin.ufpe.br/Proceedings/artigos/pdfs/sbbd_min_02.pdf>. Acesso em 05 maio de 2015.

BENEVENUTO, F; ALMEIDA, J. M; SILVA, A. S. **Atualizações em Informática 2011:** Coleta e Análise de Grandes Bases de Dados de Redes Sociais. Porto Alegre: Puc-rio, 2011. 407 p.

BROWN, R. **Django vs Flask vs Pyramid**: Choosing a Python Web Framework. 2015. Disponível em: <https://www.airpair.com/python/posts/django-flask-pyramid>. Acesso em: 10 abr. 2015.

BROWN, P. F. et al. **Class-based n-gram models of natural language**. Computational linguistics, v. 18, n. 4, p. 467-479, 1992.Disponível em: <http://delivery.acm.org/10.1145/180000/176316/p467-brown.pdf?ip=179.97.97.1&id=176316&acc=OPEN&key=4D4702B0C3E38B35%2E4D4702B0C3E38B35%2E4D4702B0C3E38B35%2E6D218144511F3437&CFID=672417161&CFTOKEN=13352685&__acm__=1431727467_d17a3f1d58c7aed822ffb883cbe40cb5>. Acesso em 04 mar. 2015.

CARRILHA JÚNIOR, J. R. **Desenvolvimento de uma Metodologia para Mineração de Textos.** 2007. 96 f. Dissertação (Mestrado) - Curso de Engenharia Elétrica, Departamento de Centro Técnico Científico, Pontíficia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2007. Cap. 3. Disponível em: <http://www.maxwell.lambda.ele.puc-rio.br/Busca_etds.php?strSecao=resultado&nrSeq=11675@1>. Acesso em: 15 mar. 2014.

CAVALIN, P. R.et al. **Real-time Sentiment Analysis in Social Media Streams**: The 2013 Confederation Cup Case. In: Encontro Nacional de Inteligência Artificial e Computacional (ENIAC). IBM Reasearch. p. 538. p. 543. 2014. Disponível em: < http://www.lbd.dcc.ufmg.br/colecoes/eniac/2014/0094.pdf>. Acesso em: 15 abr. 2015.

COLLETA, L. F. S. et al. **Combining Classification and Agrupamento for Tweet Sentiment Analysis**. In: Brazilian Conference on Intelligent Systems (BRACIS).

IEEE. p. 210. p. 215. 2014. Disponível em: < http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6984832 >. Acesso em: 15 abr. 2015.

COOLEY, R; MOBASHER, B; SRIVASTAVA, J. **Web mining**: Information and pattern discovery on the world wide Web. In: Tools with Artificial Intelligence, 1997. Proceedings, Ninth IEEE International Conference on. IEEE, 1997. p. 558-567. Disponível em: < http://ieeexplore.ieee.org/xpl/abstractAuthors.jsp?tp=&arnumber=632303&url=http%3A%2F%2Fieeexplore.ieee.org%2Fiel3%2F4999%2F13719%2F00632303>. Acesso em: 12 abr. 2014.

DAVE, K.; LAWRENCE, S.; PENNOCK, D. M. **Mining the peanut gallery**: Opinion extraction and semantic classification of product reviews. In: Proceedings of the 12th international conference on World Wide Web. ACM, 2003. p. 519-528. Disponível em: <http://dl.acm.org/citation.cfm?id=775226>. Acesso em: 20 abr. 2015.

FACEBOOK. **APIs and SDKs**. 2015a. Disponível em: < https://developers.facebook.com/docs/apis-and-sdks>. Acesso em: 12 de mar. 2015.

FACEBOOK. **Graph API Overview**. 2015b. Disponível em: < https://developers.facebook.com/docs/graph-api/overview/>. Acesso em 12 mar. 2015.

FACEBOOK. **Access Tokens**. 2015c. Disponível em: <https://developers.facebook.com/docs/facebook-login/access-tokens>. Acesso em: 15 mar. 2015.

FACEBOOK. **Facebook Login for the Web with Javascrit SDK**. 2015d. Disponível em: <https://developers.facebook.com/docs/facebook-login/login-flow-for-Web/v2.3>. Acesso em 30 mar. 2015.

FACEBOOK. **Graph API Reference**. 2015e. Disponível em: <https://developers.facebook.com/docs/graph-api/reference>. Acesso em 03 maio. 2015.

GAMMA, E. et al. **Padrões de Projeto**: Soluções Reutilizáveis de Software Orientado a Objetos. Porto Alegre: Bookman, 2000. 346 p.

GIL, A. C. **Como elaborar projetos de pesquisa**. 5. ed. São Paulo: Atlas, 2010.
GO, A.; BHAYANI, R.; HUANG, L. **Twitter sentiment classification using distant supervision**. CS224N Project Report, Stanford, p. 1-12, 2009. Disponível em: <http://s3.eddieoz.com/docs/sentiment_analysis/Twitter_Sentiment_Classification_using_Distant_Supervision.pdf>. Acesso em: 16 abr. 2015.

GOOGLE. **Google Play**. 2015a. Disponível em: < https://play.google.com/store?hl=pt-BR>. Acesso em: 02 abr. 2015.

GUEDES, G. T. A. **UML 2**: Uma abordagem prática. São Paulo: Novatec, 2009. 480 p.

HAMILTON, H. **Confusion Matrix**. 2012. Disponível em: <http://www2.cs.uregina.ca/~dbd/cs831/notes/confusion_matrix/confusion_matrix.html >. Aceeso em 12 abr. 2015.

HOLOVATY, A. **The Django Book**. 2007. Disponível em: <http://www.djangobook.com/en/2.0/index.html>. Acesso em: 21 abr. 2015.

HE, B. et al. **An effective statistical approach to blog post opinion retrieval**. In: Proceedings of the 17th ACM conference on Information and knowledge management. ACM, 2008. p. 1063-1072. Disponível em: < http://dl.acm.org/citation.cfm?id=1458223 >. Acesso em 15 abr. 2015.

HU, M.; LIU, B. **Mining and summarizing customer reviews**. In: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2004. p. 168-177. Disponível em: <http://www.aaai.org/Papers/AAAI/2004/AAAI04-119.pdf>. Acesso em: 15 abr. 2015.

IRFAN, R. et al. **A survey on text mining in social networks**. The Knowledge Engineering Review, v. 30, n. 02, p. 157-170, 2015.

KAO, A; POTEET, S. R. (Ed.). **Natural language processing and text mining**. Springer, 2007.

KELLY, R. **PyEnchant**: a spellchecking library for Python.2011. Disponível em: <http://pythonhosted.org/pyenchant/>. Acesso em: 16 abr. 2015.

KISS, J; **Facebook's 10th birthday**: from college dorm to 1.23 billion users. 2014. Disponível em: <http://www.theguardian.com/technology/2014/feb/04/facebook-10-years-mark-zuckerberg>. Acesso em: 16 abr. 2014.

LORENA, A. C.; CARVALHO, A. C. P. L. F. de C. **Uma introdução às Support Vector Machines**. Revista de Informática Teórica e Aplicada, Porto Alegre, v. 14, n. 2, p.44-67, 2007. Quadrimestre. Disponível em: < http://www.seer.ufrgs.br/rita/article/view/rita_v14_n2_p43-67> Acessado em: 21 mar. 2014.

MCCALLUM, A.; NIGAM, K. **A comparison of event models for naive bayes text classification**. In: AAAI-98 workshop on learning for text categorization. 1998. p. 41-48. Disponível em: <http://www.cs.cmu.edu/~knigam/papers/multinomial-aaaiws98.pdf>. Acesso em: 07 maio. 2015.

MELVILLE, P. GRYC, W. LAWRENCE, R. D. **Sentiment analysis of blogs by combining lexical knowledge with text classification**. In: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining.

ACM, 2009. p. 1275-1284. Disponível em: < http://dl.acm.org/citation.cfm?id=1557156 >. Acesso em: 15 abr. 2015.

MUNDO EDUCAÇÃO.**Variância e desvio padrão**. 2015. Disponível: <http://www.mundoeducacao.com/matematica/variancia-desvio-padrao.htm>. Acesso em 14 abr. 2015.

NLTK, P. **Natural Language Toolkit.** 2015. Disponível em: <http://www.nltk.org/>. Acesso em: 10 abr. 2015.

NUMPY. **NumPy**. 2013. Disponível em: <http://www.numpy.org/>. Acesso em 14 abr. 2015.

PANG, B. LEE, L. VAITHYANATHAN, S. **Thumbs up?**: sentiment classification using machine learning techniques. In: Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10. Association for Computational Linguistics, 2002. p. 79-86. Disponível em: < http://dl.acm.org/citation.cfm?id=1118704>. Acesso em: 15 abr. 2015.

PERKINS, J. **Python 3 Text Processing with NLTK 3 CookBook.** 2. ed. Birmingham B3 2pb, Uk: Packet Publishing Ltda, 2014. 287 p.

PERKINS, J. **Welcome to NLTK-Trainer's documentation!**. 2011. Disponível em: <https://nltk-trainer.readthedocs.org/en/latest/>. Acessado em 12 abr. 2015.

PYTHON. **Dictionaries**. 2015a. Disponível em: <https://docs.python.org/2/tutorial/datastructures.html#dictionaries>. Acesso em 24 abr. 2015.

RASCHKA, S. **Naive Bayes and Text Classification**: I-Introduction and Theory. arXiv preprint arXiv:1410.5329, 2014. Disponível em: <http://arxiv.org/pdf/1410.5329.pdf>. Acesso em: 06 abr. 2015.

RUSSELL, S. J. NORVIG, P. **Artificial intelligence**: a modern approach. Englewood Cliffs: Prentice hall, 2009. 1152 p.

SAIF, H. et al. **On Stopwords, Filtering and Data Sparsity for Sentiment Analysis of Twitter**. Nineth International Conference on Language Resources and Evaluation, 2014, pp. 810–817. Disponível em: <http://www.lrec-conf.org/proceedings/lrec2014/pdf/292_Paper.pdf>. Acesso em 13 abr. 2015.

SANTOS, F. L. LADEIRA, M. **The role of text pre-processing in opinion mining**. In: Intelligent Systems (BRACIS), Brazillian Conference on. IEEE, 2014. p. 50-54. Disponível em: <

http://ieeexplore.ieee.org/xpl/articleDetails.jsp?reload=true&arnumber=6984806>. Acesso em: 02 abr. 2015.

SANTOS, L. M. **Protótipo para Mineração de Opinião em Redes Sociais**: Estudos de Casos Selecionados Usando o Twitter. 2010. 103 f. TCC (Graduação) - Curso de Ciência da Computação, Departamento de Ciência da Computação, Universidade Federal de Lavras, Lavras, 2010. Disponível em: <http://www.bcc.ufla.br/wp-content/uploads/2013/2010/LeandroMatioli.pdf>. Acesso em: 14 maio 2014.

SCHELLENBACH, T. **Open Facebook API**. 2011. Disponível em: <http://django-facebook.readthedocs.org/en/latest/open_facebook/api.html>. Acesso em: 03 maio. 2015.

SCHELLENBACH, T. **Django Facebook By Thierry Schellenbach**. 2015. Disponível em: <https://github.com/tschellenbach/Django-facebook> Acesso em: 29 mar. 2015.

SCIKIT-LEARN. **User guide**: contents - scikit-learn 0.16.1. 2014a. Disponível em: <http://scikit-learn.org/stable/user_guide.html>. Acesso em: 04 abr. 2015.

SCIKIT-LEARN. **Naive Bayes**. 2014b. Disponível em: <http://scikit-learn.org/stable/modules/naive_bayes.html>. Acesso em: 05 maio. 2015.

SOARES, F. de A. **Mineração de Textos na Coleta Inteligente de Dados na Web**. 2008. 120 f. Tese (Doutorado) - Curso de Engenharia Elétrica, Departamento de Centro Técnico Científico, Pontíficia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2008. Cap. 2. Disponível em: <http://www.maxwell.lambda.ele.puc-rio.br/Busca_etds.php?strSecao=resultado&nrSeq=13212@1>. Acesso em: 07 abr. 2014.

STATISTA. **Leading social networks worldwide as of March 2015, ranked by number of active users (in millions)**. 2015. Disponível em: <http://www.statista.com/statistics/272014/global-social-networks-ranked-by-number-of-users/>. Acesso em 06 abr. 2015.

TABOADA, M. et al. **Lexicon-Based Methods for Sentiment Analysis**. In: Association for Computational Linguistics. MIT Press. v. 37, n. 2, p. 267-307, 2011.Disponível em: <http://www.mitpressjournals.org/doi/abs/10.1162/COLI_a_00049>. Acesso em: 15 abr. 2015.

WITTEN, I. et al. **Text mining**: A new frontier for lossless compression. In: Data Compression Conference, 1999. Proceedings. DCC'99. IEEE, 1999. p. 198-207. Disponível em: < http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=755669>. Acesso em: 20 marc. 2014.

W3C. **HTTP - Hypertext Transfer Protocol**. 2003. Disponível em: <http://www.w3.org/Protocols/>. Acesso em: 12 abr. 2015.