Renan Andrade Pereira

DESENVOLVIMENTO DE UM SISTEMA DE ANÁLISE DE SENTIMENTO PARA TEXTOS DE AVALIAÇÃO DE APLICATIVOS

São Paulo, Brasil

Novembro 2015

Renan Andrade Pereira
NUSP 6488548

DESENVOLVIMENTO DE UM SISTEMA DE ANÁLISE DE SENTIMENTO PARA TEXTOS DE AVALIAÇÃO DE APLICATIVOS

Monografia apresentada no Departamento de Engenharia Mecatrônica e Sistemas Mecânicos da Escola Politécnica da Universidade de São Paulo para obtenção do título de Engenheiro. Área de Concentração: Engenharia Mecatrônica

Universidade de São Paulo
Escola Politécnica
Trabalho de Conclusão de Curso

Orientador Prof. Dr. Fábio G. Cozman

São Paulo, Brasil
Novembro 2015

Declaração de Originalidade

Este relatório é apresentado como requisito parcial para obtenção do título de Engenheiro na Escola Politécnica da Universidade de São Paulo. É o produto do meu próprio trabalho, exceto onde indicado no texto. O relatório pode ser livremente copiado e distribuído desde que a fonte seja citada.

Agradecimentos

Agradeço ao professor Fábio Cozman pela orientação e apoio durante todas as etapas do projeto. Agradeço também aos professores orientadores das disciplinas PMR2500 e PMR2550 – Projeto de Conclusão de Curso I e II – pelo apoio e orientação fornecidos durante as aulas da disciplina.

Por fim agradeço à empresa BestCool & Fun Games pela disponibilização dos textos de avaliação do aplicativo Ant Smasher e acesso ao sistema de avaliação dos aplicativos da loja de aplicativos Google Play Android.

I wish to do something Great and Wonderful, but I must start by doing the little things like they were Great and Wonderful. - Albert Einstein

Resumo

Com o crescimento do mercado de smartphones, as empresas de aplicativo estão competindo cada vez mais por usuários e adquirindo valores de mercado na casa dos bilhões de dólares. A análise das opiniões emitidas pelos usuários de aplicativos é importante para os desenvolvedores terem um retorno sobre os requisitos desejados do aplicativo, defeitos, pedidos de novas funcionalidades e experiência dos usuários em relação a funcionalidades específicas. Além disso, monitorar a opinião dos usuários depois do lançamento de uma nova versão do aplicativo é importante porque as lojas dão maior visibilidade aos aplicativos com melhores notas e avaliações, o que se traduz em mais downloads. A avaliação manual das opiniões é muito custosa, devido ao alto volume de informações. A partir da demanda real de uma empresa brasileira de aplicativos, um sistema de análise de sentimentos para as avaliações do aplicativo android "Ant Smasher" foi desenvolvido. Este aplicativo possui mais de 100 milhões de downloads e recebe cerca de 200 avaliações por dia. O sistema, a partir do texto da avaliação e do idioma, classifica as avaliações automaticamente em “positivo”, “negativo” ou “neutro”. Este trabalho analisa e compara diferentes algoritmos, como Naive-Bayes, regressão logística, algumas técnicas de processamento de linguagem natural, e escolhe o algoritmo que mais atende aos requisitos. O sistema é composto de um plugin do Google Chrome para classificar automaticamente as avaliações na página “Ratings & Reviews” de um aplicativo Android (loja Goolge Play) e de um projeto em node que a partir de um arquivo csv contendo várias avaliações gera um relatório sobre o sentimento destas.

Palavras-chave: análise de sentimento, avaliação de aplicativos, algoritmo de aprendizado de máquina, Naive-Bayes, regressão logística, processamento de linguagem natural.

Abstract

With the smartphone market growth, mobile application companies are competing more and more for user acquisition and reaching billion dollars valuations. The analysis of user opinion on mobile apps is very important for these companies to have a direct feedback on the desired requirements of their application, bugs, new features requests and the user experience. Tracking user opinion is also very important, specially on new releases, since the app stores (Google Play and Apple Store) give more visibility to the highest rated apps. The manual classification of these reviews can be very expensive tough, mainly due to the high volume of information. Taking a real demand from a Brazilian mobile app company (Best Cool & Fun Games) the purpose of this report was to develop a sentiment analysis system for the mobile app “Ant Smasher”, which gets more than 200 reviews per day and has more then 100 million downloads. The system automatically classifies a review in “positive”, “negative” or “neutral”, receiving as input the review text and the language it was written. This report analyses and compares different machine learning algorithms, such as Naïve-Bayes and logistic regression, and some natural language processing techniques choosing the one that best fulfill the requirements. The system is composed by a Google Chrome plugin to analyze the reviews from the Google Play Store (Ratings and reviews section) and a project in nodejs that takes as input a csv file containing several reviews and generates a report based on the predicted sentiment of these reviews.

Keywords: sentiment analysis, mobile app reviews, machine learning, Naive-Bayes, logistic regression, natural language processing

Lista de Tabelas

Lista de Figuras

Lista de Abreviaturas

NB – Algoritmo Naive-Bayes

Mlogit – Algoritmo de regressão logística multinomial

BP – Algoritmo Banco de Palavras

substChar – Técnica de pré-processamento de texto de substituição de caracteres especiais

emoticon – Técnica de pré-processamento de texto Banco de Palavras de emoticons

lema – Técnica de pré-processamento de texto lematização

stemmer - Técnica de pré-processamento de texto de stemização

strDist - Técnica de pré-processamento de texto distância entre strings

stopWords - Técnica de pré-processamento de texto remoção de palavras vazias

BCFG – Best Cool & Fun Games

TP – ocorrências verdadeiro positivo (True Positive)

FP – ocorrências falso positivo (False Positive)

FN – ocorrências falso negativo (False Negative)

Índice

1. Introdução

Com a utilização cada vez maior dos smartphones, cuja penetração no mercado mundial entre usuários de telefone está prevista para quase 50% em 2017 [1], os aplicativos ganham grande importância. Esse mercado gera empresas bilionárias, como foi o caso da empresa desenvolvedora do aplicativo “Candy Crush”, que quando abriu seu capital na bolsa de valores de Nova York em março de 2014 arrecadou um pouco menos que US$ 500 milhões, implicando num valor de mercado de US$ 7 bilhões de dólares [2].

Não somente as empresas que abrem capital faturam alto nesse mercado. A empresa brasileira Best Cool & Fun Games, desenvolvedora do aplicativo Ant Smasher já conseguiu mais de 100 milhões de downloads [3] e faturou em 2013 cerca de US$ 3 milhões [4] .

Nesse mercado as principais lojas de aplicativos (Google Play e App Store) permitem aos clientes comprarem, procurarem e compartilharem sua opinião sobre o aplicativo por meio de um sistema de avaliação, que inclui uma nota de 0 a 5 estrelas e um comentário em texto. Essas avaliações são relevantes para análises de negócio e mercado [5], [6] e também para os desenvolvedores de aplicativo terem retorno sobre aos requisitos do aplicativo, defeitos, pedidos de novas funcionalidades e experiência dos usuários em relação a funcionalidades especificas [7], [8], [9]. Essas opiniões podem ser usadas também para melhorar o aplicativo para as próximas versões [10], [11]. Além disso as notas dadas pelos usuários são usadas pelas lojas para ranquear os aplicativos e possivelmente colocá-los em uma posição de destaque na loja [12].

Essas avaliações também são importantes para o cliente, que pode usá-las para sua decisão de baixar um aplicativo ou não. Em uma pesquisa feita com consumidores de produtos online [13], 98% disseram que avaliações online dos produtos tiveram grande influência na decisão de comprar um produto ou serviço.

Apesar disso, analisar todas as avaliações pode ser uma tarefa trabalhosa para grandes aplicativos. Um estudo mostrou que por aplicativo, em média, cerca de 22 avaliações são inseridas nas lojas Google Play e App Store [9]. Outro problema, além da quantidade, é a qualidade das avaliações, que variam desde críticas construtivas até xingamentos. Os comentários podem conter

informações diversas sobre as funcionalidades de um aplicativo, o que torna difícil o trabalho de filtrar as avaliações de uma certa funcionalidade ou característica.

Uma das soluções para extrair informações dessa grande quantidade de avaliações é a análise de sentimentos. Esta é uma técnica de aprendizado de máquina na qual sentimentos, opiniões e emoções humanas sobre determinado tópico são classificados de maneira automática [14]. A análise de sentimentos utiliza técnicas de processamento de linguagem natural [15], que consiste em traduzir as principais regras sintáticas e semânticas de uma língua para prever qual o sentimento geral de um usuário sobre um produto ou sobre suas funcionalidades.

As duas principais técnicas utilizadas em algoritmos de análise de sentimento são técnicas simbólicas ou de conhecimento de base e técnicas de aprendizado de máquina [16]. Na primeira a análise de sentimento é baseada principalmente em consulta às bases de dados já existentes que contêm sentimentos e pesos relacionados à uma palavra. Já na segunda técnica (aprendizado de máquina) treina-se um algoritmo para classificar um texto. É importante frisar que os algoritmos de aprendizado de máquina necessitam de um treinamento anterior baseado em um conjunto de avaliações previamente classificadas por um humano. O ganho de escala desses algoritmos está no tamanho reduzido desse conjunto em comparação a todas as avaliações disponíveis.

O algoritmo da análise de sentimento se torna mais complexo quando se considera outras línguas diferentes do inglês, onde a literatura e bases de dados sobre o tema não é tão abundante [17].

Tratando-se de soluções de mercado para o problema, existem já alguns serviços para análise de sentimento de aplicativos [18], [19] e também algumas soluções de software para as empresas monitorarem o sentimento dos consumidores em suas redes sociais [20].

Diante do crescente interesse desse mercado e a demanda por uma ferramenta que consiga extrair automaticamente informações de uma grande quantidade de dados, a proposta do presente trabalho é desenvolver um sistema de análise de sentimento para avaliações de aplicativos.

2. Objetivo

O objetivo do presente trabalho de conclusão de curso é o desenvolvimento de um sistema de análise de sentimentos para avaliações de aplicativos.

Esse sistema é composto de um software responsável por extrair as opiniões sobre um aplicativo automaticamente a partir da loja online e classificar um texto de avaliação em positivo, negativo ou neutro. Diferentes algoritmos foram analisados nesse trabalho para a construção de tal software.

Os dados utilizados para o desenvolvimento deste sistema foram os do aplicativo Ant Smasher, aplicativo com mais de 100 milhões de Downloads e que recebe cerca de 200 avaliações por dia, desenvolvido pela empresa Best Cool & Fun Games.

Figura 1 - Logo oficial do jogo Ant Smasher [21]

3. Estado da arte

A análise de sentimentos (também conhecida como “opinion mining”) consiste na aplicação de técnicas de processamento de linguagem natural e análise de texto, e tem como objetivo identificar e extrair sentimentos de um texto. Ela pode ser usada para identificar uma atitude, julgamento, avaliação ou comunicação emocional de um autor em relação a um determinado tópico em um documento [22].

Essa técnica pode ser feita em três níveis: nível do documento, nível da frase ou nível de um determinado aspecto ou funcionalidade. Uma análise no nível do documento implica em classificar os sentimentos gerais expressados pelo autor no documento todo em positivo, negativo ou neutro [23], [24], [25], [26]. Já a análise no nível da frase consiste em determinar se a frase é objetiva ou subjetiva, e em seguida classificar as subjetivas em positivas, negativas ou neutras [27], [28], [29], [30]. O último nível consiste num estudo mais detalhado do texto, decidindo não apenas a subjetividade geral, ou se uma sentença é positiva ou negativa, mas sim se o autor expressa ou não uma opinião/sentimento de uma funcionalidade específica [31], [32], [33]. Normalmente esse último nível consiste em duas etapas: extrair do documento as sentenças que possuem os aspectos a serem analisados e então decidir se tal aspecto é negativo, positivo ou neutro.

Apesar da análise no nível do documento ser a mais simples e genérica, é comum encontrar avaliações que contenham certa inconsistência: “A câmera é boa mas as lentes são péssimas”, ou ainda “As lentes da câmera são boas mas o suporte é horrível”. Nesse tipo de texto um algoritmo no nível da frase demonstra maior acurácia [22].

Podemos ainda destacar três técnicas de classificar o sentimento de um texto [34]:

(a) Técnicas de aprendizado de máquina de classificação de texto, como Naive-Bayes, máquina de suporte vetorial, entropia máxima [35], [16] redes de Markov [36], regressão logística [37].

(b) Extrair as palavras mais relevantes de uma amostra de textos previamente classificados e fazer uma simples contagem das palavras positivas e negativas em um documento novo.

(c) Usar bases de dados contendo dados sobre o peso de cada palavra em relação aos aspectos positivo, negativo ou neutro [38], [39].

Alguns trabalhos apresentam ainda algumas técnicas inovadoras, como [40] que apresenta técnicas de clusterização, [41] que propõe a construção de uma base de dados própria para o peso e polaridade de uma palavra e [42] que introduz lógica Fuzzy no algoritmo para torná-lo mais preciso.

Outro aspecto significativo na análise de sentimento é o pré-processamento das palavras. Um dos métodos usados é o “Bag-of-Words”, agrupamento de todas as palavras relevantes de várias avaliações em um conjunto, sem preocupaçao com a relação gramatical entre elas [36]. Alguns autores propõem modificações nesse método e conseguem atingir melhorias na acurácia final da análise de sentimento [36], [40].

Ainda no pré-processamento, diversos autores defendem o uso de técnicas de processamento de linguagem natural para melhor classificar o texto. Essas técnicas permitem a melhor compreensão de várias regras de uma língua, como exemplificado em [43], aonde a simples classificação da palavra em substantivo ou adjetivo foi usada para melhorar o algoritmo. De fato vários autores fazem uso da função da palavra na oração para a análise de sentimentos, como em [44] que propõe a identificação de advérbios e sua posição na frase para melhorar a acurácia da classificação.

Principais Desafios do Algoritmo

Alguns dos desafios do algoritmo estão relacionados à dificuldade de se obter o sentimento a partir de um texto e estão relacionados a algumas características intrínsecas da linguagem escrita [38]:

- sinonimidade: um conceito pode ser expresso por diferentes palavras e expressões

- polissemia: uma palavra pode ter múltiplos significados

- contexto: uma expressão pode ter significado bem diferente das palavras nela contidas

Outros desafios do algoritmo já estão mais relacionados ao tipo de avaliação a ser analisada. Alguns autores [35], [16] analisaram o sentimento de mensagens vindas de redes sociais como o Twitter, por exemplo e levantam as

seguintes dificuldades:

- Tamanho da mensagem: geralmente curta, dificultando a aplicação de algoritmos complexos de processamento de linguagem natural.

- Presença de gírias, palavrões, erros de grafia, etc

- Presença de emoticons[1]

Essas características dificultam a utilização de algoritmos mais complexos do ponto de vista da estrutura sintática e morfológica de uma palavra. Para as gírias, é possível utilizar dicionários que contém um banco de gírias, como o Urban Dictionary [45].

Uma dessas dificuldades pode, no entanto, ser explorada para auxiliar o algoritmo na classificação dos sentimentos: os emoticons. Ao criar-se uma base de dados para os emoticons, estes podem ser um forte indicativo da polaridade de uma avaliação [34], [35].

Para a avaliação de aplicativos, alguns trabalhos defendem que a classificação de sentimentos é bastante complexa [46], [37], inclusive sugerindo que é mais difícil que a classificação de mensagens do Twitter, pela ausência de hashtags[2].

Um dos fatores que também contribuem para a complexidade da classificação em aplicativos é a incoerência encontrada em muitas avaliações, quando por exemplo os usuários avaliam um aplicativo em 5 estrelas mas escrevem textos demonstrando insatisfação [47]. Devido a esse fato, neste trabalho somente os textos serão levados em consideração e não o número de estrelas (nota) dadas pelo usuário.

[1] Emoticons podem ser definidos como uma sequência de caracteres tipográficos, tais como: :), :(, ^-^, :3 e :-); ou, também, uma imagem (usualmente, pequena), que traduz ou quer transmitir o estado

[2] Tags são palavras-chave (relevantes) ou termos associados a uma informação, tópico ou discussão que se deseja indexar de forma explícita no aplicativo Twitter, e também adicionado ao Facebook, Google+ e/ou Instagram.

4. Requisitos do Projeto

Para melhor descrição dos requisitos e resultados o sistema de análise de sentimentos desenvolvido pode ser dividido em duas partes:

- Software de análise de sentimento: parte responsável por receber o texto a ser avaliado, rodar o algoritmo de análise de sentimento e tratar a resposta do algoritmo. Essa é a parte que o usuário interage.

- Algoritmo de análise de sentimento: responsável por analisar uma opinião e retornar o sentimento dela. Dependendo do algoritmo escolhido, este pode requerer algum tipo de treinamento (processamento) prévio antes de ser utilizado.

4.1. Requisitos do Software de análise de sentimento

Conforme descrito na seção 1, a análise de sentimentos é uma importante ferramenta para o desenvolvedor de aplicativos. A partir dos objetivos descritos na seção 2 e consulta à empresa de aplicativos Best Cool & Fun Games sobre as principais necessidades de um sistema dese tipo, os seguintes requisitos foram definidos para o projeto:

- O software deve analisar as opiniões de um aplicativo a partir do site[3] da loja de aplicativos do Google, da seção ‘Ratings & Reviews, onde o desenvolvedor do aplicativo pode ver as opiniões e respondê-las ao usuário.

- A partir da resposta do algoritmo de análise de sentimento o software deve comparar o sentimento do texto (positivo, negativo ou neutro) com a nota (rating/estrelas) dada pelo usuário. Caso o texto seja positivo e a nota menor que três estrelas, o software deve responder ao usuário uma mensagem pré-formatada perguntando por que o usuário não deu a nota máxima (5 estrelas).

- O software deve gerar um relatório a partir de arquivo csv padrão disponível para desenvolvedores no site da Google Play, contendo o número de estrelas (rating), texto e idioma da avaliação. O relatório deve conter a quantidade de opiniões classificadas como positivas, negativas ou neutras.

[3] https://play.google.com/apps/publish/

Tal relatório é importante para acompanhar o sentimento geral das avaliações perto do lançamento de uma nova versão do aplicativo, por exemplo.

4.2. Requisitos do algoritmo de análise de sentimento

No caso de um algoritmo que necessite de pré-processamento/treinamento prévio antes de ser acoplado ao software, o tempo máximo desse pré-processamento deve ser de 24 horas.

O tempo de resposta do algoritmo para uma opinião (depois de treinado) deve ser de no máximo 18 segundos.

A geração do relatório csv com as estatísticas das opiniões de um dia deve ser feita em no máximo uma hora.

O algoritmo deve retornar um sentimento 'positivo', negativo' ou 'neutro' a partir de cada opinião. A classe 'neutro' é necessária porque várias avaliações possuem conteúdo incoerente ou não dão uma opinião final/sentimento final sobre o produto.

A precisão[4] do algoritmo nas classes positivas e negativas deve ser maior do que 80% e a abrangência[5] dessas classes também deve ser maior que 80%. Essa última medida é importante porque a classe de opiniões negativa não é tão frequente, e a abrangência consegue traduzir o quanto de opiniões negativas foram identificadas pelo algoritmo. As medidas de avaliação dos algoritmos e seu significado serão aprofundadas na seção 6.

[4] Precisão nesse projeto será definida como a razão entre os documentos (opiniões) corretamente atribuídos a uma classe sobre o total de documentos atribúidos a uma classe.
[5] Também chamado de recall (inglês), a abrangência é definido como o número de documentos (opiniões) corretamente atribuídas a uma classe sobre o total de documentos que realmente pertencem à classe.

5. Síntese de Soluções

5.1. Software de análise de sentimento

A plataforma a ser utilizada deve atender aos requisitos de analisar as opiniões a partir do site da Google Play (espaço do desenvolvedor) e indicar quais necessitam de alguma resposta. Uma plataforma que atende a esses requisitos é uma extensão de navegador (plug-in[6]). A plataforma do sistema então será uma extensão do Google Chrome. Este navegador foi escolhido por ser o mais utilizado [48].

Uma vantagem dessa plataforma é a possibilidade de se chamar diferentes funções pelo console do navegador (com parâmetros) sem a complexidade do desenvolvimento de uma interface gráfica.

Outra vantagem é a escalabilidade: ela pode ser facilmente utilizada por outros aplicativos da loja Android.

A linguagem de programação a ser utilizada é Javascript, a mais indicada para a plataforma escolhida.

Para a geração do relatório a partir de CSV padrão da Google Play contendo as avaliações, um projeto em node será utilizado (javascript).

O plugin Google Chrome conterá as seguintes funções para o usuário:

- analyzeReview: que a partir de uma opinião (string) e idioma retorna o sentimento (‘positivo’, ’negativo’ ou ‘neutro’).

- replyReviewsSemiAutomatic: automaticamente extrai as opiniões da seção “Opiniões & Reviews” de um aplicativo na loja Google Play e caso o sentimento da avaliação seja positivo e o rating (estrelas) baixo, o programa automaticamente insere um texto perguntando ao usuário por que ele deu um rating baixo se gostou do aplicativo.

O projeto em node terá a seguinte função:

- generateCsvReport: gera as estatísticas a partir de um csv com as opiniões dos usuários.

5.2. Algoritmos de Análise de Sentimento

[6] Plug-ins são recursos semelhantes a programas que são instalados e funcionam de maneira oculta, facilitando o recuso para funcionalidade de algo que se deseja que seja executado

Conforme explicado na seção 3 diferentes algoritmos podem ser usados para classificar o sentimento de uma opinião. Alguns serão usados neste trabalho e terão seus resultados apresentados discutidos na seção 7. O algoritmo que melhor atender aos requisitos será escolhido para ser usado no sistema.

5.2.1. Algoritmo banco de palavras

Este é um algoritmo que a partir de um banco de palavras pré-definido decide o sentimento do texto da avaliação do usuário.

O banco de palavras contém três categorias com pesos: “palavras positivas” (peso 1), “palavas negativas” (peso -1) e “palavras extremamente negativas” (peso -100). Para cada palavra de uma opinião seu respectivo peso é associado e somado ao peso total. Ao final de todas as palavras se o peso total for positivo a opinião é classificada como positiva, caso seja negativo como negativo e se nenhuma palavra for encontrada no banco de palavras o texto é classificado como neutro.

A lista de palavras pode ser encontrada na tabela 1:

Palavras positivas	Palavras negativas	Palavras extremamente negativas
loves, love, luv, loved, love it, amazing, fun, funny, great, greatest, liked, like, best, thanks, super, cool, excelent, nice, awesome, interesting, amusing, good, addicting, addictive, wow, outstanding, recommend, thumbs up, sick, beautiful, entertaining, entertains, exceptional, perfect, excellent, coolest, nicest, enjoyed, lovely, cooler, adorable, wonderful	disappointed, stopped, working, fine, ok, issue, fixed, bug, load, not, screen, uninstalled, annoying, bland, sucks, stupid, bad, shitty, worst, woorst, wooorst, woooorst, did not, didn't, fuck, do not recommend, no good, “ ugly, shit, virus, boring, poor, poorly, bored, boring, not recommend, useless, slow, install, violence, improve, crap, pay, paying, crash, crashes, disgusting, disgust, horrible	ads, ad, advertisements, advertisement, advertising, advertise, commercials, publicity, adds, announcement, announcements, promotions

Tabela 1 - Banco de palavras para o algoritmo

A grande vantagem desse algoritmo é a simplicidade: com poucas linhas de

código, sem treinamento prévio das palavras ou qualquer pré-processamento consegue-se classificar um texto.

$$Classificação\ de\ uma\ review$$

1. $Inicialização\ das\ variáveis$:

   $weigth = 0$

   $palavrasPositivas, palavrasNegativas,$
   $palavrasExtremamenteNegativas\ (conforme\ tabela\ 1)$

2. $Para\ cada\ palavra\ w_i\ da\ review\ a\ ser\ analisada$:

   $SE\ weight \in palavrasPositivas, \quad então\ weight = weight + 1$

   $Senão, SE\ weight \in palavrasNegativas, \quad então\ weight = weight - 1$

   $Senão, SE\ weight \in palavrasExtremamenteNegativas, então\ weight = weight - 100$

3. $Classificar\ a\ review\ de\ acordo\ com\ o\ sinal\ de\ weight$:

   $SE\ weight > 0, então\ review\ é\ POSITIVA$

   $SE\ weight < 0, então\ review\ é\ NEGATIVA$

   $SE\ weight == 0, então\ review\ é\ NEUTRA$

Figura 2 - Algoritmo de classificação Banco de Palavras

Um problema, no entanto, é a obtenção desse banco de palavras. Nesse caso elas foram obtidas por um “expert” (pessoa que trabalhava na empresa há certo tempo analisando e respondendo as avaliações) que observou certo padrão que se repetia nos textos de avaliação. Outro ponto negativo é a falta de escalabilidade: todas as opiniões precisam ser traduzidas para o inglês, além disso algumas palavras são específicas de um aplicativo em particular, necessitando de um expert para cada novo aplicativo que o sistema for implementado. Neste aplicativo em específico, por exemplo, vemos que a presença de propagandas normalmente indica a polaridade negativa de um comentário. Para aplicativos que não possuem propagandas, tal banco de palavras não teria boa performance.

Outra característica que podemos observar nesse tipo de algoritmo é que ele tenta prever os erros de grafia mais comuns de uma palavra, não sendo exaustivo, dependendo da memória do “expert” para conter os erros mais comuns. Exemplo: “excelent, excellent”.

5.2.2. Algoritmo Naive-Bayes

O algoritmo de Naive-Bayes é amplamente utilizado nos problemas de classificação de texto e bem estudado e citado, estando presente em vários trabalhos de classificação de texto [49], [50], [51], [52].

Naive-Bayes assume que a distribuição de palavras num documento é multinomial. O documento é tratado como uma sequência de palavras e assume-se que a a posição de cada palavra é independente da outra.

Para classificação assume-se que há um número fixo de classes $c \in \{1, 2, \dots, m\}$, cada um com um vetor de parâmetros $\vec{\theta}_c = \{\theta_{c1}, \theta_{c2}, \dots, \theta_{cn}\}$, onde n é tamanho do vocabulário[7], $\sum_i \theta_{ci} = 1$ e θ_{ci} é a probabilidade que a palavra i ocorra na classe c. A probabilidade de um documento ocorrer dada a sua classe (segundo o teorema de Naive-Bayes) é:

$$p(d|\vec{\theta}_c) = \frac{(\sum_i f_i)!}{\prod_i f_i!} \prod_i (\theta_{ci})^{f_i} \quad (1)$$

aonde f_i é a frequência da palavra i no documento d.

Assumindo uma distribuição a priori sobre as classes, $p(\theta_c)$ nós chegamos à regra de classificação de um documento, que seleciona a classe que contém a maior probabilidade a posteriori [53]:

$$l(d) = argmax_c[\log p(\theta_c) + \sum_i f_i \log \theta_{ci}] \quad (2)$$

Os parâmetros θ_{ci} são estimados a partir de um training set contendo documentos já classificados, de acordo com a seguinte fórmula [54]:

$$\widehat{\theta_{ci}} = \frac{N_{ci}+1}{N_c+\alpha} \quad (3)$$

Sendo N_{ci} o número de vezes que a palavra i aparece nos documentos de classe c, N_c o número total de palavras encontradas na classe c e α o tamanho do vocabulário (total) de palavras.

Após o cálculo de todos os parâmetros, para um documento novo a equação (2) deve ser utilizada para decidir qual a classe do documento em questão.

No caso deste trabalho temos 3 classes (“positivo”, “negativo” e “neutro”) e a partir das opiniões manualmente classificadas (training set) foi possível calcular os parâmetros θ_{ci} de todas as classes. Uma vez que todos os parâmetros foram calculados a equação (2) é aplicada para cada uma das

[7] O vocabulário é o conjunto de palavras encontrados nos documentos de uma classe c.

classes e a classe que obteve o maior valor é escolhida para classificar o texto.

Treinamento do Algoritmo

1. *Extrair todas as palavras (Vocabulário) dos textos do training set*
2. *Para cada classe c (positivo, negativo, neutro) estimar a probabilidade a priori* $p(\theta_c)$:

$$p(\theta_c) = \frac{\text{Número de reviews pertencentes à classe } c}{\text{Número total de reviews}}$$

3. *Para cada palavra do vocabulário calcula-se o parâmetro* $\hat{\theta}_{ci}$ *para cada classe :*

$$\hat{\theta}_{ci} = \frac{(\text{Número de vezes que a palavra } i \text{ aparece nas reviews de classe } c) + 1}{(\text{Número total de palavras na classe } c) + (\text{Número total de palavras do vocabulário})}$$

Figura 3 - Algoritmo de treinamento de Naive-Bayes

Classificação de uma review

1. *Extrai-se todas as palavras* w_i *do texto d a ser classificado.*
2. *Para cada classe calcula-se a probabilidade a posteriori, utilizando-se os parâmetros correspondentes à* w_i:

$$l(positivo|d) = \log p(\theta_{positivo}) + \sum_i f_i \log \theta_{(positivo)i}$$

$$l(negativo|d) = \log p(\theta_{negativo}) + \sum_i f_i \log \theta_{(negativo)i}$$

$$l(neutro|d) = \log p(\theta_{neutro}) + \sum_i f_i \log \theta_{(neutro)i}$$

3. *A review é classificada segundo a classe que obteve o maior valor de l*

Figura 4 - Algoritmo de classificação Naive-Bayes

A hipótese da independência das palavras presente em Naive-Bayes é bem forte. Ela implica que em um texto a posição das palavras não tem relação com o sentimento, o que pode não ser verdade. Se uma opinião começa com um adjetivo positivo e depois critica alguma coisa específica, o algoritmo não conseguirá “aprender” tal comportamento, como nos exemplos: “Muito bom, mas gráficos ruins”, “Nice game but bad graphics”.

Outra implicação dessa hipótese é que o peso de uma palavra na classificação em positivo, negativo ou neutro é independente das outras

palavras do texto. Isso exclui o efeito de advérbios de um adjetivo: “muito bom” para o algoritmo é o mesmo que “bom”, “pouco bom”, “quase bom”. Apesar dessas hipóteses imprecisas o algoritmo tem boa eficácia nos problemas de classificação de texto, conforme mostrado em [55].

Outro aspecto do algoritmo Naive-Bayes é a baixa qualidade dos estimadores. Se ao invés de apenas classificar o texto em positivo, negativo ou neutro estivéssemos interessado no valor numérico da probabilidade do texto pertencer à determinada classe, Naive-Bayes não seria uma boa escolha. Apesar disso as classificações obtidas a partir dos números são confiáveis [56].

5.2.3. Regressão Logística Multinomial

Outro algoritmo a ser testado é a regressão logística multinomial, também conhecida como Entropia Máxima nos problemas de processamento de linguagem. Este algoritmo pertence à família dos classificadores chamados de exponenciais.

Para esta classificação assume-se que há um vetor de características $x = \left[x_1, x_2, \dots, x_j, \dots, x_d\right]^T$ na opinião a ser classificada. No caso deste trabalho essas características são as palavras de um texto e d é o tamanho do vocabulário. O valor x_j será 1 se o texto conter a a palavra j e 0 caso contrário.

A classe de um texto é encodada como um vetor $y = [y_1, \dots, y_k]^T$ de tamanho igual ao número de classes. Neste vetor apenas uma coordenada é igual a 1 (classe correspondente ao documento) enquanto todas as outras são 0.

A probabilidade do texto pertencer à classe k, dado seu vetor de características x é dada por:

$$p(y_k = 1|x, \boldsymbol{B}) = \frac{\mathrm{e}^{\beta_k^T x}}{\sum_{k'} \mathrm{e}^{\beta_{k'}^T x}}, \tag{4}$$

Aonde a matriz $B = [\beta_1, \dots, \beta_k]$ representa os parâmetros de cada classe $\beta_k = [\beta_{k1}, \beta_{k2}, \dots, \beta_{kd}]$.

Considerando agora exemplos já categorizados $D = \{(x_1, y_1), \dots, (x_i, y_i), \dots, (x_n, y_n)\}$. A estimativa da máxima probabilidade dos parâmetros B é equivalente a minimizar:

$$l(B|D) = -\sum_i [\sum_k y_{ik}\ \beta_k^T x_i - \ln(\sum_k \exp(\beta_k^T x_i)] \tag{5}$$

Um problema que surge ao se utilizar este algoritmo é o overfitting: o algoritmo só consegue classificar com boa acurácia as opiniões presentes nos exemplos D já categorizados. Se em uma opinião negativa uma palavra aparece "por acaso", ou seja, não tem relação direta com o sentimento do algoritmo, esta palavra terá um coeficiente que não traduz seu real significado. Para atenuar o problema pode-se selecionar os parâmetros do vetor x que serão realmente incluídos no treinamento do algoritmo. Um métodos simples de mitigar este erro é admitir somente as características que apareceram mais de f vezes naquela classe.

Dada a complexidade do algoritmo e da implementação de métodos numéricos para a resolução da equação (5), optou-se por utilizar o software estatístico R [57] e o pacote "nnet" [58] para a aplicação da regressão logística multinomial.

Treinamento do algoritmo

1. *Inicialização das variáveis*:

   $Vocabulário = [\,]$

   D: *matriz de reviews já classificadas. A primeira coluna é o texto, a segunda é a classe.*

2. *Construção do vetor de parâmetros x (Vocabulário)*:

   Para cada review já classificada (linha da matriz D):

      Para cada palavra do texto:

         SE palavra não está em vocabulário,
         então palavra é inserida em vocabulário

3. *Selecionar parâmetros (palavras) que serão usadas pelo algoritmo*

   Para cada palavra em vocabulário, contar ocorrência nos textos de D:

      $SE\ \big(ocorrência(palavra)\big) < f$, *então palavra é removida do vocabulário*

4. *Construção do vetor x para cada review já classificada*:

   Para cada review já classificada (linha i da matriz D):

      Para cada palavra de posição j do texto:

         SE *palavra não está em vocabulário, então* $x[i][j] = 0, SENÃO\ x[i][j] = 1$

5. *Aplicação da função mlogit do pacote nnet do software – R*:

   $\boldsymbol{B} = logit(D, X)$

Figura 5 - Treinamento Regressão Logística Multinomial

Para a classificação do texto, uma vez a matriz **B** calculada basta aplicar a fórmula (4) para obtermos a probabilidade de cada uma das classes. O software R possui uma função (predict) que classifica um texto a partir dos coeficientes gerados na função.

5.2.4. Técnicas de pré-processamento de texto

Conforme visto na seção 3 várias técnicas de pré-processamento de texto podem ser aplicadas para melhorar a performance de um sistema de classificação de textos. Essas técnicas podem ser aplicadas tanto no algoritmo Naive-Bayes quanto no de regressão logística multinomial. Nesta seção as técnicas utilizadas neste trabalho são apresentadas.

Stemização (stem)

A stemização consiste em reduzir palavras flexionadas ou derivadas ao seu tronco ou raiz. O algoritmo escolhido para aplicar a stemização é o "Porter Stemmer", desenvolvido pelo pesquisador Martin Porter para a língua inglesa. Este algoritmo é um dos mais populares e está baseado na ideia de que os sufixos em inglês (1200) podem ser em sua maioria formados por um conjunto de sufixos menores e mais simples [59]. O algoritmo consiste em 5 passos, e dentro de cada passo várias regras são aplicadas contra a palavra e se uma delas passa o sufixo é removido e o algoritmo passa então para a próxima regra.

Para este algoritmo o módulo porter-stemmer para nodejs foi utilizado [60].

Base de dados de emoticons (emoticons)

Conforme mencionado em [35], [34] construir uma base de dados de emoticons pode melhorar a performance do algoritmo. A partir de emoticons presentes em [61] construiu-se a seguinte base de dados:

Emoticons positivos	Emoticons negativos
^.^ ^-^ ^_^ B-) *-* *_* :-) :) :D :o) :] :3 :c) :> =] 8) =) :} :^) : ⊃) ;) (: [: ;D ;P :p :P :-D 8-D 8D x-D xD X-D XD =-D =D	>:[:-(:(:-c :c :-< : ⊃ C :< :-[:[:{ ;(:'-(:'(>:\\ >:/ :-/ :-. :/ :\\ =/ =\\ :L =L :S):

=-3 =3 B^D :-)) ;-) ;) *-) *) ;-] ;] ;D ;^) :-, >:P :-P :P X-P x-p xp XP :-p :p =p :-Þ :Þ :þ :-þ :-b :b d:	

Os emoticons encontrados nos textos são então substituídos pelas palavras “positiveEmoticons” e “negativeEmoticons” para emoticons positivos e negativos respectivamente. Dessa forma o sentimento expressado por diferentes emoticons fica reunido em uma palavra.

Substituição de caracteres especiais (substChar)

Para melhorar a performance dos algoritmos de classificação é importante remover todos os caracteres que não trazem informações novas ao algoritmo. Para tal todos os caracteres diferentes de números e letras são removidos do texto. Essa substituição é bem útil para eliminar os caracteres de pontuação e deve ser feita depois da substituição da base de dados de emoticons.

Remoção de palavras vazias (stopWord)

Um método conhecido para diminuir o ruído de textos e aumentar a performance dos classificadores é a remoção de palavras-vazias [62]. Palavras vazias são as palavras mais comuns de uma língua que muitas vezes não trazem informação nenhuma, como artigos, preposições, pronomes, conectores, etc.

Palavras vazias em português	Palavras vazias em inglês
"de","a","o","que","e","do","da","em","um","para","é","com","não","uma","os","no","se","na","por","mais","as","dos","como","mas","foi","ao","ele","das","tem","à","seu","sua","ou","ser","quando","muito","há","nos","já","está","eu","também","só","pelo","pela","até","isso","ela","entre","era","depois","sem","mesmo","aos","ter","seus","quem","nas","me","esse","eles","estão","você","tinha","foram","essa","num","nem","suas","meu","às","minha","têm","numa","pelos","elas","havia","seja","qual","será","nós","tenho","lhe","deles","essas","esses","pelas","este","fosse","dele","tu","te",	"a","able","about","across","after","all","almost","also","am","among","an","and","any","are","as","at","be","because","been","but","by","can","cannot","could","dear","did","do","does","either","else","ever","every","for","from","get","got","had","has","have","he","her","hers","him","his","how","however","i","if","in","into","is","it","its","just","least","let","like","likely","may","me","might","most","must","my","neither","no","nor","not","of","off","often","on","only","or","other","our","own","rather","said","say","says","she","should","since","so","some","than","that","the","their","them","then","t

"vocês","vos","lhes","meus","minhas","teu","tua","teus","tuas","nosso","nossa","nossos","nossas","dela","delas","esta","estes","estas","aquele","aquela","aqueles","aquelas","isto","aquilo","estou","está","estamos","estão","estive","esteve","estivemos","estiveram","estava","estávamos","estavam","estivera","estivéramos","esteja","estejamos","estejam","estivesse","estivéssemos","estivessem","estiver","estivermos","estiverem","hei","há","havemos","hão","houve","houvemos","houveram","houvera","houvéramos","haja","hajamos","hajam","houvesse","houvéssemos","houvessem","houver","houvermos","houverem","houverei","houverá","houveremos","houverão","houveria","houveríamos","houveriam","sou","somos","são","era","éramos","eram","fui","foi","fomos","foram","fora","fôramos","seja","sejamos","sejam","fosse","fôssemos","fossem","for","formos","forem","serei","será","seremos","serão","seria","seríamos","seriam","tenho","tem","temos","tém","tinha","tínhamos","tinham","tive","teve","tivemos","tiveram","tivera","tivéramos","tenha","tenhamos","tenham","tivesse","tivéssemos","tivessem","tiver","tivermos","tiverem","terei","terá","teremos","terão","teria","teríamos","teriam"	here","these","they","this","tis","to","too","twas","us","wants","was","we","were","what","when","where","which","while","who","whom","why","will","with","would","yet","you","your"

Tabela 2 - Lista de palavras vazias em inglês e português. Retirado de [63] e [64]

Lematização (lema)

Esta é uma técnica utilizada para agrupar palavras que possuem o mesmo lema. Lema é a forma canônica de uma palavra. A lematização e a stemização são técnicas similares, mas a stemização se mantém restrita mais à própria palavra, enquanto a lematização leva em consideração o contexto (classe gramatical) e o significado da palavra. O algoritmo de stemização é constituído de uma série de regras (lógica), enquanto que na lematização a consulta a uma base de palavras é necessária.

Para a lematização o módulo ‘natural nodejs’ foi utilizado (biblioteca em javascript contendo várias funções de processamento de linguagem [65]). Este módulo utiliza a base de dados WordNet para encontrar o lema de uma palavra. WordNet [66] é uma base de dados léxica em inglês desenvolvida pela universidade de Princeton.

Distância entre strings (distStr)

Uma técnica que pode ser utilizada também no intuito de agrupar palavras com o mesmo significado é a distância entre strings. Observou-se que no tipo de texto dos usuário grande quantidade de palavras com pequenos erros de digitação (“interesting”, “interestng”,“legal”, “legl”) outras palavras com vogais duplicadas (“woorst”, “worst”, “bom”,”booom”), . Para tentar agrupar essas palavras em um mesmo grupo de maneira mais genérica pode-se aplicar uma técnica de distância entre palavras/strings.

Neste trabalho a métrica de distância Jaro-Winkler foi utilizada. Essa métrica é baseada no número e na ordem dos caracteres comuns entre duas strings e dá uma pontuação maior para string que são parecidas desde o início. A pontuação varia de 0 (completamente diferente) a 1 (completamente iguais). Ela é utilizada para detectar típicos erros gramaticais [67] e foi baseada na observação de Winkler que erros de digitação ocorrem com mais frequência no meio ou no fim de uma palavra, mas raramente no começo.

A distância de Jaro entre duas strings s_1 e s_2 é dada por [68]:

$$d_j = \begin{cases} 0 \; se \; m = 0 \\ \frac{1}{3}\left(\frac{m}{|s_1|} + \frac{m}{|s_2|} + \frac{m-t}{m}\right) se \; m \; ! = 0 \end{cases} \quad (6)$$

Onde m é o número de caracteres coincidentes e t o número de transposições.

Dois caracteres de s_1 e s_2 são considerados coincidentes se eles são o mesmo e não estão mais distantes que $\left[\frac{\max(|s_1|,|s_2|)}{2}\right] - 1$.

A transposição é definida como o número de caracteres coincidentes encontrados em diferentes posições de s_1 e s_2 divido por 2.

A distância de Jaro-Winkler é uma modificação da distância de Jaro e é definida por [68]:

$$d_w = d_j + (lp(1 - d_j)) \quad (7)$$

aonde d_j é a distância de Jaro para as strings s_1 e s_2 , l é o comprimento do prefixo comum às duas strings (até no máximo 4 caracteres) e p é uma constante que representa o quanto a distância deve ser ajustada para cima por causa do prefixo comum. Winkler utiliza $p = 0.1$ em seu trabalho.

Ao classificar um texto, quando uma palavra não consta no vocabulário treinado, a métrica de Jaro-Winkler é utilizada e substitui-se a palavra por uma que tenha no mínimo uma pontuação de 0.95.

6. Metodologia de avaliação do desempenho dos algoritmos

Conforme descrito na seção 4 os requisitos do projeto foram definidos em torno da precisão e abrangência. Outra medida importante bastante utilizada em trabalhos de classificação de textos e aprendizado de máquina é o F-Score.

A tabela 2 resume essas medidas com seus respectivos significados:

Métrica	Fórmula	Interpretação
Precisão	$TP/(TP + FP)$	Porcentagem de documentos corretamente identificados sobre o total de documentos preditos para determinada classe.
Abrangência	$TP/(TP + FN)$	Porcentagem de documentos corretamente identificados sobre o total de documentos pertencentes a determinada classe.
F-Score	$2 . \frac{Precisão\ .\ Abrangência}{Precisão + Abrangência}$	Média harmônica entre precisão e abrangência (medida do compromisso entre precisão e acurácia).

Tabela 3 - Resumo das métricas. TP são as ocorrências verdadeiro positivo (true positive), FP as ocorrências falso positivo (false positive) e FN as ocorrências falso negativo (false negative).

No caso de classificação de textos a abrangência é uma medida importante para classes que tem baixa ocorrência, para que o algoritmo consiga detectar esta classe. A medida do F-Score por sua vez consegue medir o quanto se está comprometendo a acurácia em relação a abrangência, podendo ser interpretado como uma espécie de função “E” entre as duas métricas (se uma delas for baixa essa medida também será baixa).

No caso das avaliações negativas por exemplo, podemos ter uma precisão alta (quase todas as vezes que o algoritmo previu que o texto era negativo ele era negativo de fato). No entanto se a abrangência foi baixa (algoritmo não está conseguindo identificar os textos negativos) isso pode prejudicar o desenvolvedor pois ele não saberia o número de avaliações negativas recebidas, daí a importância da utilização da métrica F-Score para esse problema.

Outra técnica utilizada em algoritmos de aprendizado de máquina para comparar algoritmos é a validação cruzada (cross-validation). Essa técnica é utilizada para medir a acurácia de um modelo na prática, pois ao treinar um algoritmo de aprendizado de máquina existe o risco de overfitting, isto é, do

algoritmo ter uma performance muito boa somente nos dados em que foi treinado. Sendo assim para calcular as medidas de acurácia, abrangência e F-Score não recomenda-se utilizar os mesmos dados utilizados para treinamento do algoritmo.

Para mitigar esse risco a validação cruzada separa os dados já classificados em duas partes: uma parte seria para o treinamento do algoritmo (grupo de treinamento/training set) e outra parte (geralmente 10%) para testar o algoritmo e validar os resultados (grupo de validação/testing set).

Um dos problemas do método anterior é que o resultado pode variar muito dependendo de quais documentos foram para o grupo de treinamento e quais foram para o grupo de validação. Uma maneira de contornar isso é fazendo k-fold cross validation, que consiste em dividir o dataset em k grupos e performar a validação cruzada para cada um dos k datasets. Para a k-ésima validação cruzada o grupo de teste seria o grupo k e o grupo de treinamento seriam todos os outros grupos exceto o k.

Após fazer a validação k-fold obtém-se k valores de F-Score. Após analisar o erro decorrente das diferentes maneiras de calcular o F-Score a partir desses dados, [68] sugere a seguinte fórmula para o cálculo do F-Score com o menor erro:

$$F - Score = {}^{2\,.\,TP}/_{(2\,.\,TP + FP + FN)},$$

aonde TP é o total de verdadeiros positivos (true positives), FP o total de falsos positivos (false positives) e FN o total de falsos negativos (false negatives) encontrados em todas as k iterações da validação cruzada.

Neste projeto a avaliação entre os diferentes algoritmos se dará com o cálculo do F-Score realizando k-fold validation com k = 10.

7. Resultados dos algoritmos

Os algoritmos e técnicas de processamento de texto descritos na seção anterior foram aplicados no dataset fornecido pela empresa.

O dataset contém 8296 textos de revisão do aplicativo ant smasher feitos pelos usuários no período de janeiro de 2014 até maio de 2015. Além dos textos, o dataset fornecido contém o idioma do texto (inglês e português) e o sentimento identificado (feito manualmente por um funcionário da empresa). O arquivo fornecido está no formato csv e pode ser acessado junto com o código do programa deste projeto no anexo A.

Texto	Idioma	Sentimento
Pelo visto e bom To baixando	pt	1
Good excelent	en	1
Legal Muito legal de mais	pt	1
horrible	en	-1
hello	en	0
um saco	pt	-1

Tabela 4 - Três primeiras linhas do dataset fornecido.

O sentimento neste arquivo está encodado: 1 corresponde a uma avaliação positiva, -1 a uma avaliação negativa e 0 a uma avaliação neutra.

Em todos os resultados abaixo descritos a técnica de k-fold cross validation foi utilizada com k = 10.

7.1. Algoritmo banco de palavras (BP)

Conforme discutido na seção 5.2.1 este algoritmo consiste em comparar o texto de uma opinião com um banco de polaridade de palavras. Neste algoritmo a técnica de k-fold cross validation é desnecessária pois tal algoritmo não precisa de treinamento e não há, portanto, risco de overfitting. A técnica foi mantida para a medida de F-Score ser comparada com a dos outros algoritmos.

O algoritmo banco de palavras sem nenhum outro algoritmo de pré-processamento de texto fornece os seguintes resultados:

Classe	F-Score	precisão	abrangência
positivo	0.881	0.962	0.813
negativo	0.724	0.778	0.678
neutro	0.610	0.470	0.867

Tabela 5 - F-Score, precisão e abrangência do algoritmo BP (BP)

A seguir os resultados ao aplicarmos os algoritmos de pré-processamento de texto:

Classe	F-Score	precision	recall
positivo	0.888	0.963	0.825
negativo	0.756	0.783	0.731
neutro	0.635	0.501	0.864

Tabela 6 - Algoritmo BP com substituição de caracteres especiais (BP + substChar)

Classe	F-Score	precision	recall
positivo	0.892	0.963	0.831
negativo	0.758	0.784	0.733
neutro	0.639	0.509	0.860

Tabela 7 - Algoritmo BP com substituição de caracteres especiais e banco de emoticons (BP + substChar + emoticon)

Classe	F-Score	precision	recall
positivo	0.897	0.962	0.839
negativo	0.770	0.781	0.759
neutro	0.655	0.531	0.855

Tabela 8 - Algoritmo BP com substituição de caracteres especiais, banco de emoticons e distância entre palavras (BP + substChar + emoticon + distStr)

O gráfico abaixo ilustra a evolução do F-score para as diferentes classes:

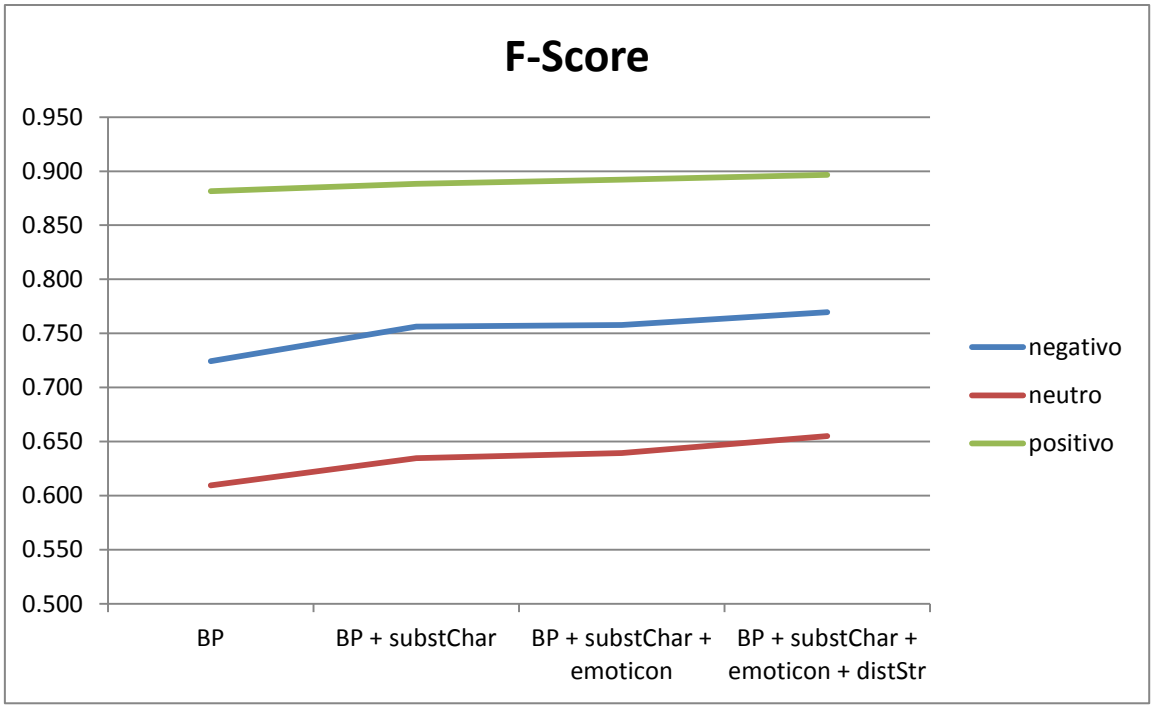

Figura 6 - Variação do F-Score para os diferentes algoritmos com processamento de texto (dataset dos textos em inglês)

Como este algoritmo só foi feito para o idioma inglês, os textos de revisão em português não foram testados.

7.2. Algoritmo Naive-Bayes (NB)

O algoritmo Naive-Bayes apresentou os seguintes resultados **para as opiniões em inglês**:

Classe	F-Score	precisão	abrangência
positivo	0.879	0.803	0.974
negativo	0.829	0.858	0.807
neutro	0.132	0.868	0.073

Tabela 9 - Algoritmo NB (inglês)

Para a língua inglesa todos os algoritmos de pré-processamento de texto apresentados neste trabalho podem ser utilizados.

Classe	F-Score	precisão	abrangência
positivo	0.883	0.812	0.970
negativo	0.844	0.858	0.838
neutro	0.192	0.856	0.109

Tabela 10 - Algoritmo NB com substituição de caractere especial , banco de dados de emoticon e distância de palavras (NB + substChar + emoticon + distStr) (inglês)

Ao aplicarmos o pré-processamento de remoção de palavras vazias o F-Score da classe neutra diminui enquanto o F-Score das outras classes aumenta ligeiramente:

Classe	F-Score	precisão	abrangência
positivo	0.884	0.811	0.973
negativo	0.857	0.872	0.850
neutro	0.154	0.765	0.085

Tabela 11 - Algoritmo NB com substituição de caractere especial, banco de dados de emoticon, distância de palavras e remoção de palavras vazias (NB + substChar + emoticon + strDist + stopWord) (inglês)

Ao aplicarmos o algoritmo de stemização o F-Score da classe neutra aumenta em 33%:

Classe	F-Score	precisão	abrangência
positivo	0.884	0.815	0.968
negativo	0.848	0.855	0.850
neutro	0.206	0.848	0.120

Tabela 12 - Algoritmo NB com substituição de caractere especial, banco de dados de emoticon, distância de palavras e stemização (NB + substChar + emoticon + strDist + stemmer) (inglês)

O algoritmo de lematização quando aplicado dá um resultado parecido com o da distância entre palavras:

Classe	F-Score	precisão	abrangência
positivo	0.879	0.807	0.968
negativo	0.840	0.857	0.829
neutro	0.157	0.778	0.089

Tabela 13 - Algoritmo NB com substituição de caractere especial, bando de emoticons e lematização (NB + substChar + emoticon + lema) (inglês)

O gráfico abaixo resume o desempenho das diferentes técnicas de processamento de texto para o algoritmo Naive-Bayes:

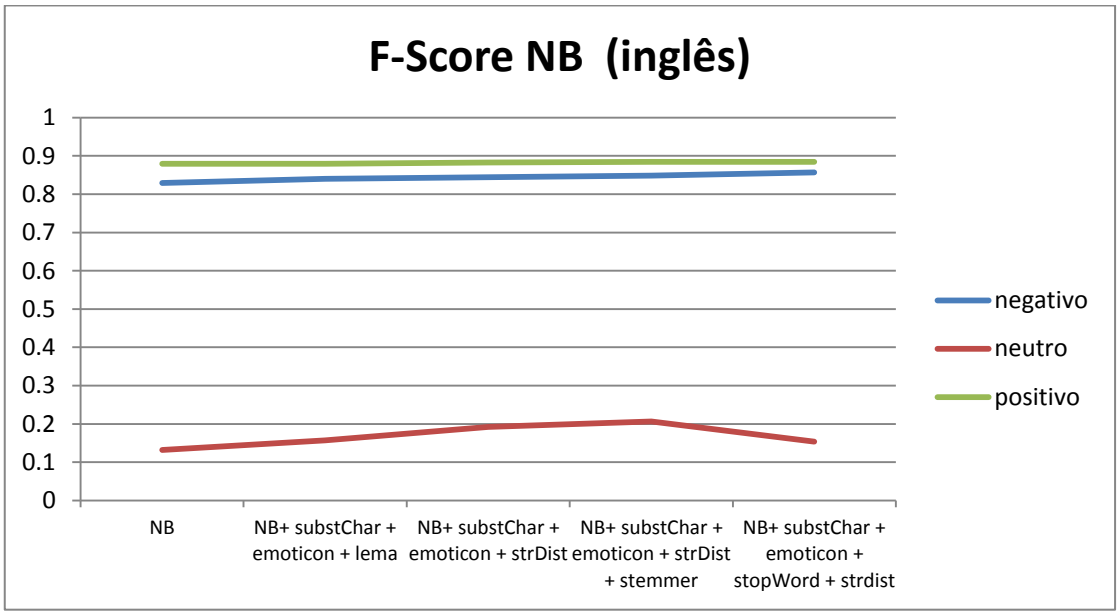

Figura 7- F-Score dos diferentes métodos de processamento de texto para Naive-Bayes (inglês)

Para os textos em português somente os métodos de processamento de texto lematização e stemização não podem ser utilizados pois suas bibliotecas funcionam apenas para textos em inglês.

Classe	F-Score	precisão	abrangência
positivo	0.900	0.840	0.969
negativo	0.800	0.850	0.757
neutro	0.188	0.687	0.107

Tabela 14 - Desempenho do algoritmo Naive-Bayes sem nenhum processamento de texto (português)

A aplicação das técnicas de substituição de caractere, base de dados de emoticons e distância de palavras (NB + substChar + emoticon + strDist) melhorou em 15% o F-Score da classe neutra do algoritmo quando comparada com o algoritmo Naive-Bayes sem nenhum processamento de texto. Já a aplicação dessas técnicas com a remoção de palavras vazias piorou o desempenho do algoritmo, como podemos ver no gráfico a seguir.

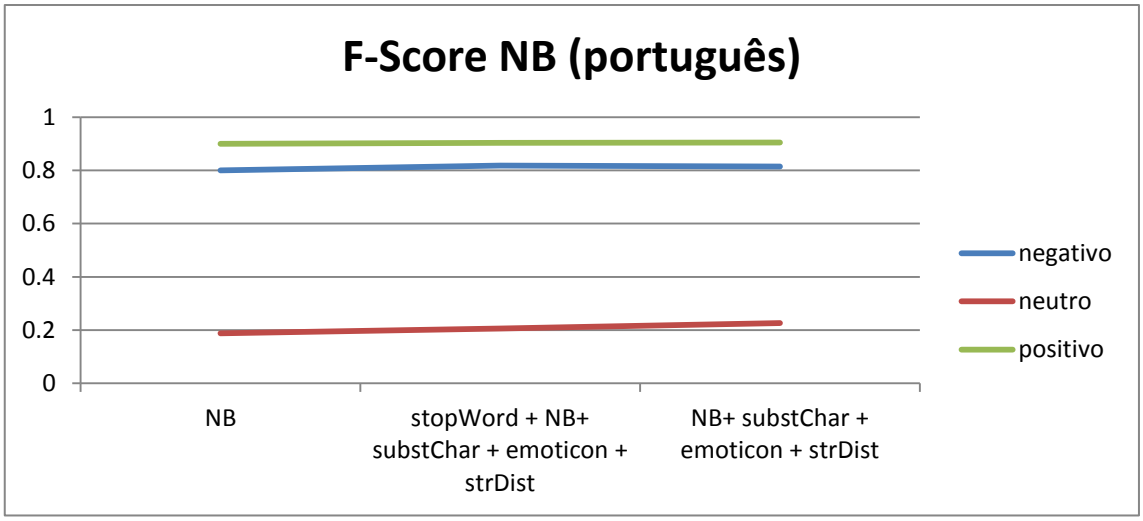

Figura 8 - Desempenho do algoritmo Naive-Bayes para textos em português

7.3. Algoritmo Regressão Logística Multinomial (mlogit)

O algoritmo de regressão logística multinomial apresentou os seguintes resultados para as opiniões em inglês:

Classe	F-Score	precisão	abrangência
positivo	0.894	0.907	0.882
negativo	0.732	0.758	0.708
neutro	0.685	0.629	0.751

Tabela 15 - Resultados algoritmo regressão logística multinomial sem processamento de texto (inglês)

Nenhuma técnica de pré-processamento de texto descrita neste trabalho melhorou a performance do algoritmo (F - Score) como podemos ver na tabela abaixo.

F-Score por classe			Algoritmo
negativo	neutro	positivo	
0.732	0.685	0.894	Mlogit
0.705	0.672	0.888	MLogit + substChar
0.706	0.671	0.886	MLogit + substChar + emoticon
0.706	0.671	0.886	Mlogit + substChar + emoticon + strDist
0.691	0.672	0.883	Mlogit + substChar + emoticon + stopWord
0.549	0.592	0.832	Mlogit + substChar + emoticon + strDist + stemmer
0.59	0.553	0.776	Mlogit + substChar + emoticon + lema

Tabela 16 - Desempenho (F-Score) do algoritmo de regressão logística multinomial ara diferentes técnicas de pré-processamento de texto (inglês)

A tabela 17 mostra também que a abrangência da classe neutra aumenta quando-se adiciona as técnicas de pré-processamento de texto mas a precisão diminui para essa mesma classe. Para as outras classes as técnicas de pré-processamento de texto pioram tanto a abrangência quanto a precisão.

Precisão/abrangência por classe			Algoritmo
negativo	neutro	positivo	
0.758 / 0.708	0.629 / 0.751	0.907 / 0.882	Mlogit
0.757 / 0.659	0.602 / 0.76	0.9 / 0.876	MLogit + substChar
0.752 / 0.665	0.6 / 0.763	0.901 / 0.872	MLogit + substChar + emoticon
0.752 / 0.665	0.6 / 0.763	0.901 / 0.872	Mlogit + substChar + emoticon + strDist
0.74 / 0.648	0.601 / 0.763	0.897 / 0.87	Mlogit + substChar + emoticon + stopWord
0.634 / 0.484	0.478 / 0.779	0.869 / 0.798	Mlogit + substChar + emoticon + strDist + stemmer
0.589 / 0.591	0.414 / 0.829	0.886 / 0.691	Mlogit + substChar + emoticon + lema

Tabela 17 - Precisão e abrangência para as diferentes técnicas de pré-processamento de texto (inglês)

Para as opiniões em português o desempenho do algoritmo é apresentado na tabela 18.

Classe	F-Score	precisão	abrangência
positivo	0.902	0.911	0.893
negativo	0.773	0.811	0.738
neutro	0.545	0.482	0.627

Tabela 18 - Desempenho do algoritmo mlogit para opiniões em português

Os desempenhos das técnicas de processamento de linguagem foram semelhantes ao dos textos em inglês: nenhuma técnica de processamento de texto apresentada conseguiu melhorar o F-Score do algoritmo de regressão logística multinomial.

A piora na performance do algoritmo está ligada a diminuição do número de features (palavras do vocabulário) ao aplicar-se alguma técnica de processamento de texto. Ao remover-se os caracteres especiais, por exemplo, algumas palavras desaparecem do vocabulário e o algoritmo tem sua performance levemente prejudicada pois o número de “features” que predizem o algoritmo diminui.

7.4. Análise dos resultados

Das seções anteriores podemos comparar as melhores versões de cada algoritmo para os textos em inglês:

F-Score por classe			Algoritmo
negativo	neutro	positivo	
0.770	0.655	0.897	BP + substChar + emoticon + distStr
0.848	0.206	0.884	NB+ substChar + emoticon + strDist + stemmer
0.732	0.685	0.894	Mlogit

Tabela 19 - Comparação entre as melhores versões dos diferentes algoritmos para os textos em inglês

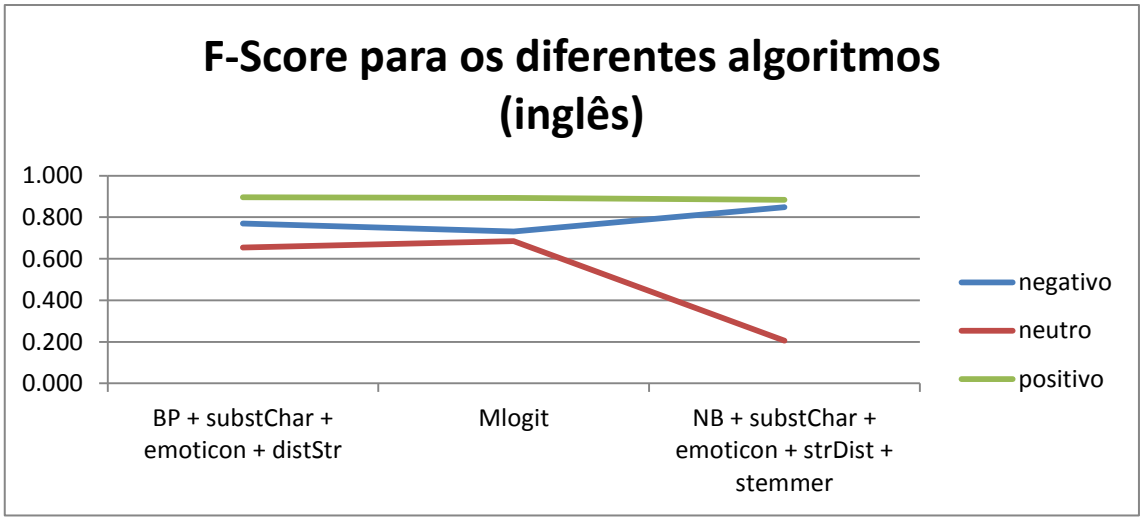

Figura 9 - F-Score para os diferentes algoritmos para os textos em inglês

Observa-se que os algoritmos banco de palavras (BP) e regressão logística multinomial (mlogit) possuem desempenho parecido. Observamos no algoritmo Naive-Bayes uma forte queda no F-Score da classe neutra e uma melhora no desempenho da classe negativa. Observa-se que essa queda do F-Score se deve a uma grande queda na abrangência (a precisão da classe neutra para o algoritmo Naive-Bayes chega a ser maior do que os outros algoritmos). Isso deve-se ao fato de que o algoritmo Naive-Bayes classifica mais documentos como negativos do que os outros algoritmos, levando a um alto número de falsos positivos para a classe neutra. Ao classificar mais documentos como negativos acaba aumentando a precisão e abrangência da classe negativa.

Observamos que para os textos em português o mesmo acontece.

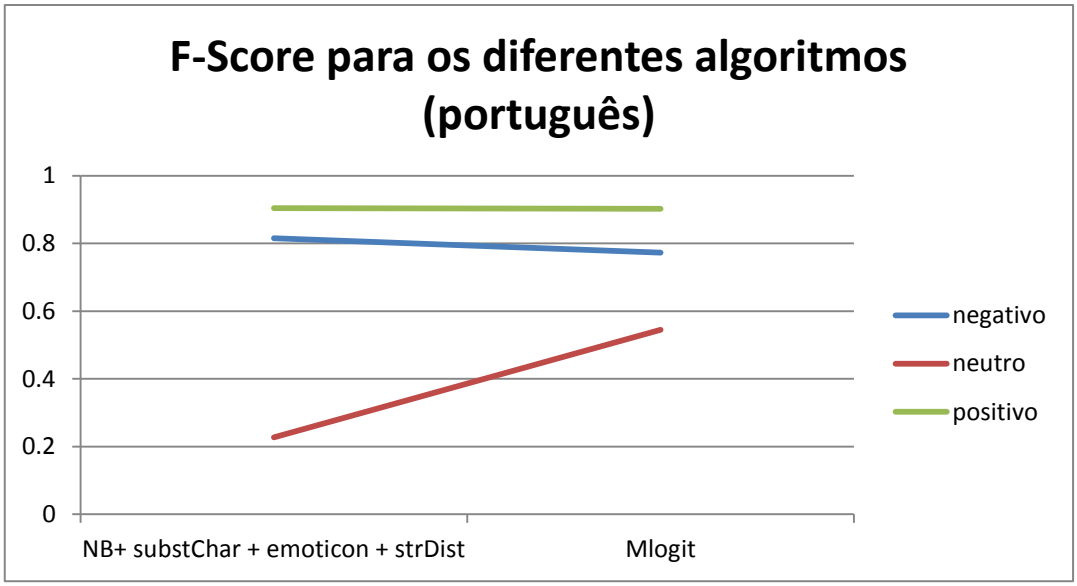

Figura 10 - F-Score para os diferentes algoritmos para os textos em português

Outro resultado que depreendemos das seções anteriores é a falta de eficácia das técnicas de processamento de texto normalmente utilizada em outros problemas. O algoritmo de distância entre palavras (distStr) funcionar melhor do que o lema de uma palavra pode ser explicado por exemplo pelo número de palavras com erros de grafia encontrado nos textos. O algoritmo de remoção de palavras vazias (bastante usado em problemas de classificação de texto no Twitter [62]) acabou piorando a performance dos algoritmos. Isso deve-se à natureza particular dos textos encontrados nas opiniões de aplicativo: a presença de palavras vazias pode significar, por exemplo, um texto mais bem elaborado, que pode ser uma característica típica de textos negativos, enquanto que textos positivos tendem a ser mais sucintos.

A boa performance do algoritmo banco de palavras reflete a particularidade do problema de classificação de textos de avaliação de um aplicativo. Apesar de extremamente simples e de fácil implementação em qualquer linguagem o algoritmo possui performance comparável aos outros apresentados. A desvantagem deste algoritmo, como já discutido na seção 5.2.1 é a dependência de um expert para a definição do banco de palavras. Além disso para mudar de idioma ou de app outros bancos de palavras devem ser feito, tornando o algoritmo pouco escalável.

Pode-se argumentar que os outros algoritmos (Naive-Bayes e regressão logística) também precisariam de um esforço (construção e classificação

manual de um training set) para mudar de idioma, mas a tarefa de classificar os textos de opiniões é menos complexa do que definir quais palavras tem mais peso na polaridade de uma opinião manualmente.

O algoritmo Naive-Bayes mesmo não tendo boa performance na classe neutra é o único algoritmo que atende aos requisitos, pois possui maior F-Score na classe negativa. Além disso o algoritmo é facilmente escalável para outros aplicativos.

O algoritmo escolhido para compor o sistema desenvolvido neste trabalho é o Naive-Bayes com as técnicas de: substituição de caractere especial, banco de dados de emoticon, distância entre palavras e stemização para os textos em inglês. Para as opiniões em português o mesmo algoritmo foi escolhido mas a técnica de stemização não foi utilizada pois esta só está disponível para o inglês.

8. Sistema de análise de sentimento

Após a discussão de resultados dos diferentes algoritmos na seção anterior, o algoritmo a ser implementado no sistema de análise de sentimento para as opiniões em inglês é Naive-Bayes com as técnicas de substituição de caractere especial, banco de dados de emoticon, distância entre palavras e stemização. Para os textos em português o algoritmo é o mesmo, mas a técnica de stemização não será utilizada pois a biblioteca utilizada para esta técnica não suporta o idioma português.

No anexo B instruções mais detalhadas de como instalar e utilizar o sistema são fornecidas.

8.1. Plug-in Google Chrome

O sistema de análise de sentimento funciona a partir da seção ‘Ratings & Reviews’ do site da loja google Play.

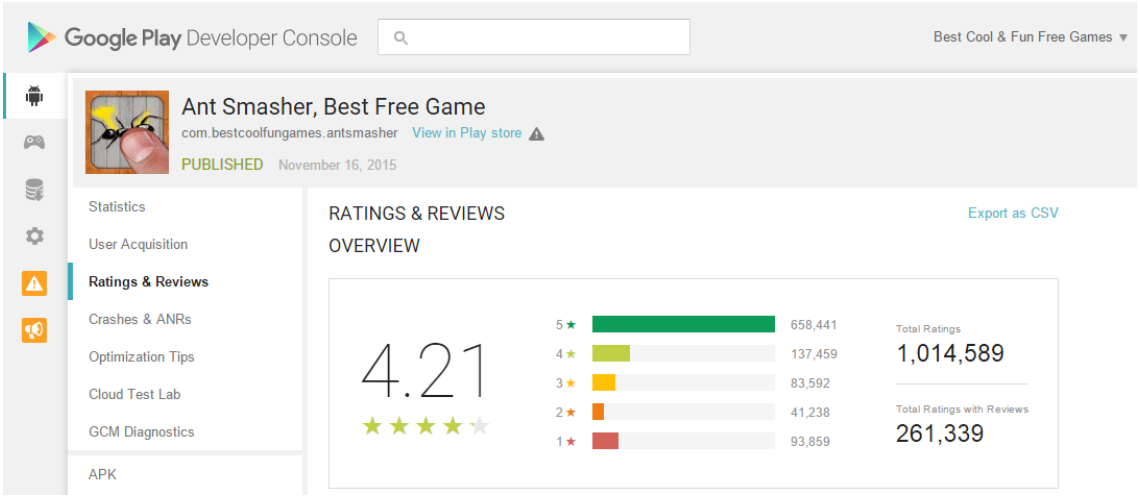

Figura 11- Seção Ratings & Reviews da loja Google Play para um aplicativo

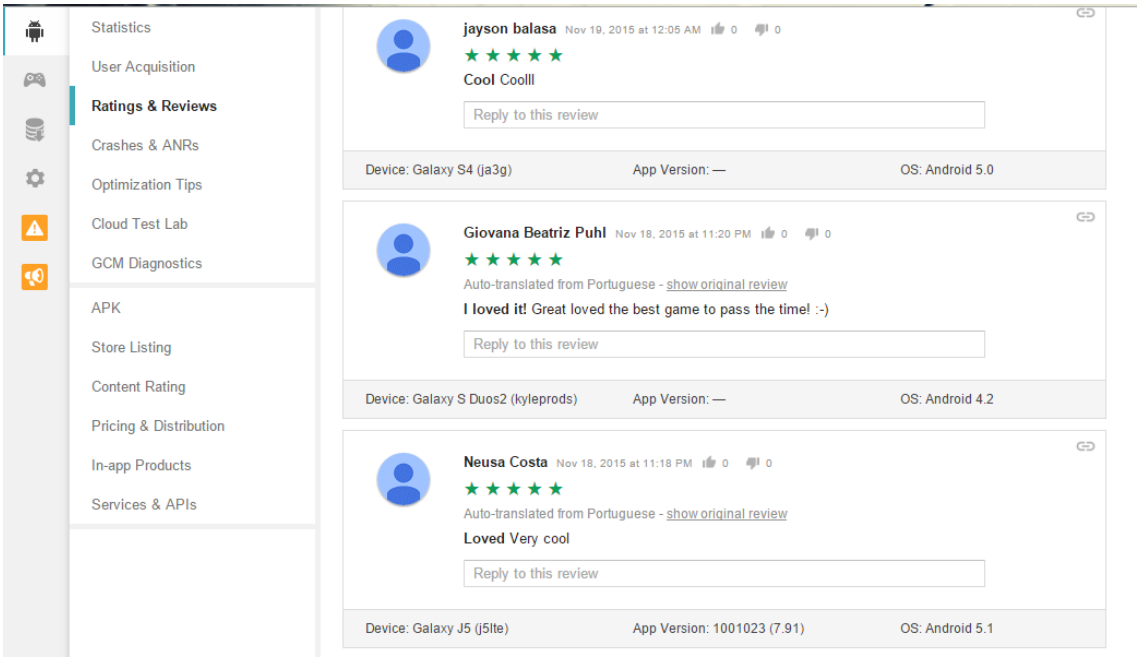

Figura 12 - Opiniões de usuários na seção 'Ratings & Reviews' a serem classificadas pelo sistema

Para acessar tal página o desenvolvedor do aplicativo deve acessar sua conta na Google Play e selecionar o aplicativo que deseja ver as opiniões.

Na página de opiniões o usuário deve abrir o console do Google Chrome e chamar a função replyReviewsSemiAutomatic() e o algoritmo vai analisar cada uma das opiniões. Se a opinião for classificada como positiva e o usuário tiver dado menos do que três estrelas para o aplicativo o sistema automaticamente gera uma resposta perguntando porque o usuário deu nota baixa se gostou do aplicativo.

Algoritmo de responder opiniões do usuário

Para cada opinião encontrada na página "Ratings and Reviews":

1. *Classificar a opinião segundo o algoritmo escolhido*
2. *SE opinião é da classe positiva e nota do usuário é menor ou igual a três estrelas:*

   Perguntar ao usuário no mesmo idioma que ele escreveu o texto porque ele deu nota baixa se gostou do aplicativo

3. *SENÃO SE a opinião é da classe positiva:*

   Mandar uma mensagem de agradecimento ao usuário

Figura 13 - Algoritmo utilizado para responder o usuário baseado na classificação de sentimento da opinião

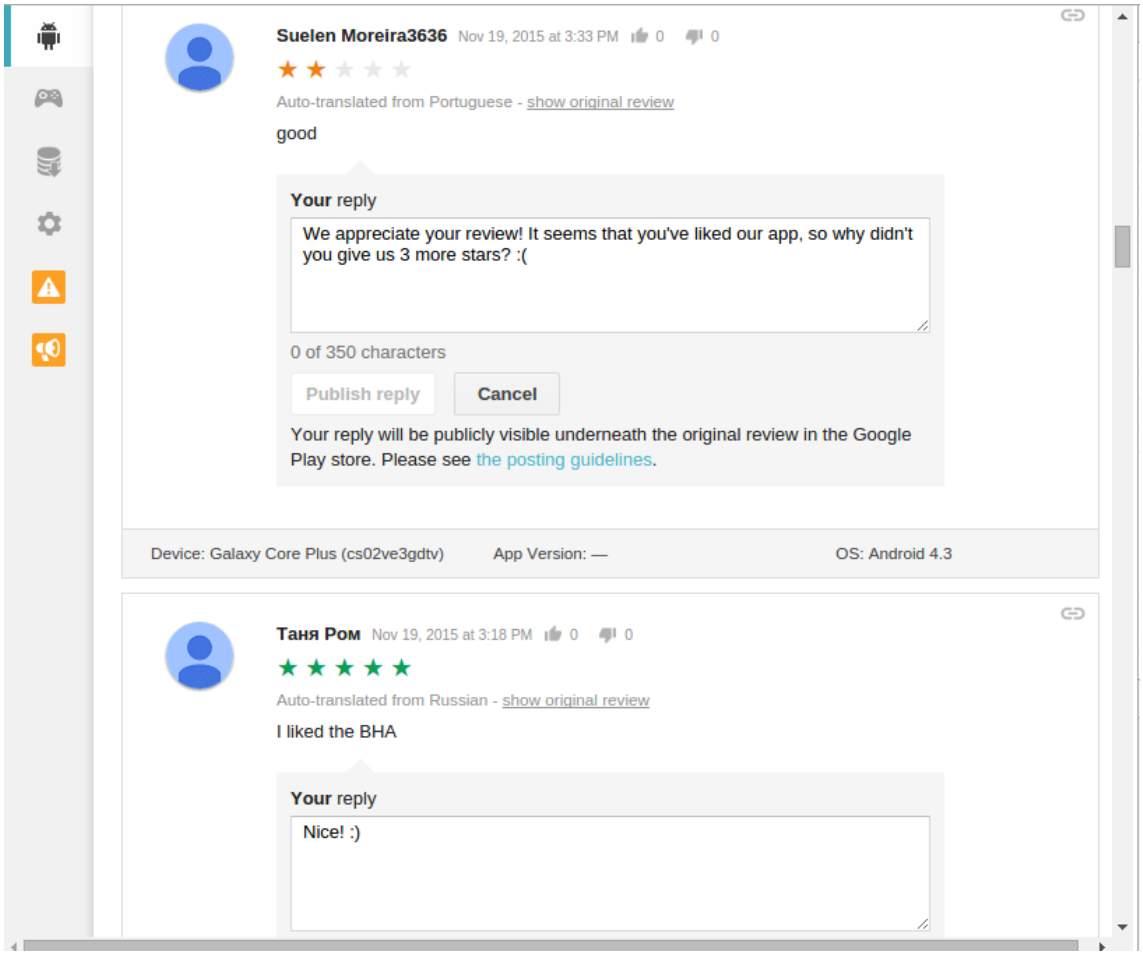

Figura 14 - Respostas geradas pelo sistema de análise de sentimentos

Observamos na Figura 14 os dois tipos de resposta fornecidas pelo sistema: na primeira opinião o texto foi classificado como positivo, mas o usuário deu apenas 2 estrelas para o aplicativo. Neste caso o sistema responde com uma mensagem pré-programada perguntando porque o usuário não dá 5 estrelas para o aplicativo. Já na segunda opinião classificada como positiva o sistema manda uma mensagem de agradecimento. A lista de todas as mensagens pré-definidas pode ser encontrada no código do plugin no Anexo A.

8.2. Geração de relatórios

Na loja de aplicativos android um arquivo csv contendo as opiniões dos usuários de um mês todo pode ser baixado. Ao fornecer o arquivo o sistema gera um relatório contendo o número de reviews classificadas como negativa,

positiva e neutra.

O arquivo contém várias informações como data da opinião, nota, dispositivo no qual a opinião foi feita, entre outros. O sistema irá utilizar somente as colunas “Review Text” que contém o texto a ser classificado e “Reviewer” que contém a lingua na qual o texto foi escrito.

O sistema responde com um arquvo (csv) contendo as seguintes informações:

Idioma	Positivo	Negativo	Neutro
en	1202	122	80
pt	2201	225	122

Tabela 20 - Resposta do sistema para um arquivo contendo 1404 opiniões em inglês e 2548 opiniões em português

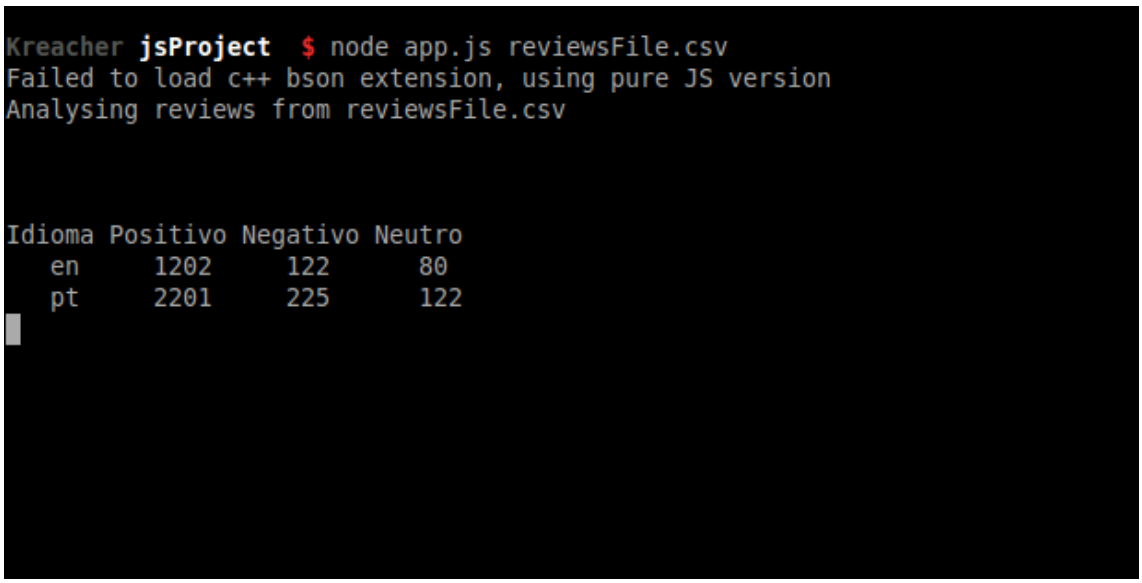

Figura 15 - Utilização do módulo de geração de relatório. Neste exemplo o arquivo ‘reviewsFile.csv’ é o arquivo que contém as opiniões (baixado a partir do site da google play)

9. Conclusão

9.1. Discussão

Este trabalho de conclusão de curso cumpriu seus objetivos ao desenvolver um sistema de classificação de sentimentos de textos de avaliação de aplicativos, cumprindo os requisitos e elaborando um análise comparativa de diferentes soluções de algoritmos de classificação.

O trabalho confirmou a boa performance do algoritmo Naive-Bayes apesar de suas hipóteses de independência iniciais parecerem equívocas.

Outro ponto interessante do trabalho foi mostrar que algoritmos simples como o banco de palavras, mesmo com boa performance podem não ser a melhor escolha por não serem escaláveis e facilmente adaptáveis a outros aplicativos e idiomas.

Ao escolher um algoritmo que tem melhor performance em determinada classe (Naive-Bayes), ao invés de um algoritmo com uma performance média melhor, considerando todas as classes, ressaltou-se também a importância específica da performance do algoritmo em determinada classe para um sistema de classificação de textos de avaliação de aplicativos. Para o desenvolvedor, a classe neutra não é tão importante, sendo mais importante um bom desempenho nas classes positiva e negativa, o que realmente traduzem o sentimento do usuário do aplicativo.

Outra conclusão importante do trabalho é a diferença de performance das técnicas de processamento de linguagem natural quando aplicadas em diferentes tipos de texto. A técnica de remoção de palavras vazias (palavras sem sentido semântico, como preposições, pronomes, conectores), mostra-se eficiente em textos curtos (análise de sentimento do Twitter), apresenta-se pouco eficaz em textos de avaliação de aplicativos. Outro exemplo é o uso de técnicas simples de processamento como o cálculo da distância entre palavras ter um desempenho melhor do que técnicas mais complexas como as de lematização.

9.2. Trabalhos Futuros

A extensão deste trabalho de conclusão de curso pode ter um foco mais

empresarial ou mais acadêmico.

Do ponto de vista empresarial, dada a necessidade de um sistema de análise de sentimentos em uma empresa de aplicativos, o mesmo sistema pode ser desenvolvido e vendido. Ao invés de uma extensão do Google Chrome e um projeto em node, pode-se disponibilizar uma API via web de análise de sentimento, onde o desenvolvedor manda o set de treinamento num formato estabelecido e depois pode enviar um texto de avaliação e o servidor responde com a classificação de tal texto.

Já na área acadêmica pode-se desenvolver ainda mais as técnicas utilizadas de pré-processamento de texto, incluindo algumas não utilizadas neste trabalho, mas que podem aumentar a performance, como consulta a dicionários externos (para correção de grafia), identificação da função gramatical da palavra no texto (essa particularmente difícil dada a natureza curta dos textos).

Do ponto de vista do algoritmo, para melhorar o desempenho da regressão logística, métodos de regularização[8] mais complexos podem ser utilizados. Ainda no que se refere à regressão logística outras características (além do vocabulário) podem ser incluídas, por exemplo, o tamanho do texto, a quantidade de adjetivos/verbos presentes no texto, etc.

[8] Os métodos de regularização diminuem o risco de overfiting do algoritmo, isto é, do algoritmo ter um desempenho somente nos exemplos em que foi treinado.

10. Referência Bibliográfica

1. EMARKETER. Smartphone Users Worldwide Will Total 1.75 Billion in 2014. **eMarketer website**, 2014. Disponivel em: <http://www.emarketer.com/Article/Smartphone-Users-Worldwide-Will-Total-175-Billion-2014/1010536>. Acesso em: 25 Março 2015.

2. FORBES MAGAZINE. Candy Crush Maker King Digital Prices IPO At $22.50 Per Share, 2014. Disponivel em: <http://www.forbes.com/sites/maggiemcgrath/2014/03/25/candy-crush-maker-prices-ipo-at-22-50-per-share/>. Acesso em: 28 Março 2015.

3. GOOGLE. Ant Smasher, Best Free Game. **Google Play Android**, 2015. Disponivel em: <https://play.google.com/store/apps/details?id=com.bestcoolfungames.antsmasher&hl=en>. Acesso em: 02 Abril 2015.

4. REVISTA ISTOE. Os milionários dos aplicativos. **ISTOE Economia & Negócios**, 2012. Disponivel em: <http://www.istoe.com.br/reportagens/194216_OS+MILIONARIOS+DOS+APLICATIVOS>. Acesso em: 10 Abril 2015.

5. YU, X. et al. Mining Online Reviews for Predicting Sales Performance: A Case Study in the Movie Domain. **Knowledge and Data Engineering, IEEE Transactions on**, v. 24, n. 4, p. 720-734, Abril 2012.

6. GHOSE, A.; IPEIROTIS, P. G. Estimating the Helpfulness and Economic Impact of Product Reviews: Mining Text and Reviewer Characteristics. **Knowledge and Data Engineering, IEEE Transactions on**, v. 23, n. 10, p. 1498-1512, Outubro 2011.

7. CARREÑO, G.; WINBLADH, K. **Analysis of user comments:** an approach for software requirements evolution. ICSE ’13 Proceedings of the 2013 International Conference on Software Engineering. [S.l.]: IEEE Press. 2013. p. 582-591.

8. JIA, M. H. Y.; ZHANG, Y. **App store mining and analysis:** MSR. In Proc. of Working Conference on Mining Software Repositories - MSR ’12. [S.l.]: [s.n.]. 2012. p. 108-111.

9. PAGANO, D.; MAALEJ, W. **User feedback in the appstore:** an empirical study. Proc. of the International Conference on Requirements Engineering - RE ’13. [S.l.]: [s.n.]. 2013. p. 125-134.

10. MAALEJ, D.; PAGANO, D. **On the Socialness of Software**. 2011 IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing. [S.l.]: [s.n.]. 2011. p. 864-871.

11. GRAF, N. S. F.; MAIDEN, N. **Using mobile review tools to give end-users their own voice**. Requirements Engineering Conference (RE), 2010 18th IEEE International. [S.l.]: [s.n.]. 2010. p. 37-46.

12. TECHCRUNCH. Apple's App Store Rankings Algorithm Changed To Consider Ratings, And Possibly Engagement. **Techcrunch news**, 2013. Disponivel em: <http://techcrunch.com/2013/08/23/apples-app-store-rankings-algorithm-changed-to-favor-ratings-and-possibly-engagement/>. Acesso em: 05 Abril 2015.

13. PING ZINE. Pinterest Users Buy More Items Than Facebook Users, According to Survey. **Ping! Zine WEB HOSTING MAGAZINE**, 2012. Disponivel em: <http://www.pingzine.com/pinterest-users-buy-more-items-than-facebook-users-according-to-survey-15335/>. Acesso em: 28 Março 2015.

14. DZOGANG, F.; MARIE-EANNE, L.; MARIA, R. **Expressions of Graduality for Sentiments Analysis - A Survey**. Fuzzy Systems (FUZZ), 2010 IEEE International Conference. [S.l.]: [s.n.]. 2010.

15. CAMBRIA, E. et al. New Avenues in Opinion Mining and Sentiment Analysis. **Intelligent Systems, IEEE**, v. 28, n. 2, p. 15-21, Março-Abril 2013.

16. NEETHU, M. S.; RAJASREE, R. **Sentiment analysis in twitter using machine learning techniques**. Computing, Communications and Networking Technologies (ICCCNT),2013 Fourth International Conference on. [S.l.]: [s.n.]. 2013. p. 1-5.

17. LIN, Z. et al. **Make It Possible:** Multilingual Sentiment Analysis without Much Prior Knowledge. Web Intelligence (WI) and Intelligent Agent Technologies (IAT), 2014 IEEE/WIC/ACM International Joint Conferences on. [S.l.]: [s.n.]. 2014. p. 79-86.

18. TECHCRUNCH. Adeven's New App Store Sentiment Analysis Tool Helps Developers Draw Useful Conclusions From App Reviews, 2012. Disponivel em: <http://techcrunch.com/2012/12/04/adevens-new-app-store-sentiment-analysis-tool-helps-developers-draw-useful-conclusions-from-app-reviews/>. Acesso em: 29 Março 2015.

19. ADJUST. Get on top of your app store reviews, 2014. Disponivel em: <https://www.adjust.com/company/overview/2014/07/31/apptrace-app-store-stats-review-sentiment-analysis/>. Acesso em: 12 Abril 2015.

20. TRACKUR. What can Trackur Do For You?, 2015. Disponivel em: <http://www.trackur.com/social-media-monitoring>. Acesso em: 08 Abril 2015.

21. GOOGLE. Esmaga Formigas Jogo Grátis. **Google Play**, 2015. Disponivel em: <https://play.google.com/store/apps/details?id=com.bestcoolfungames.antsmasher&hl=pt_BR>. Acesso em: 15 Setembro 2015.

22. VALAKUNDE, N. D.; PATWARDHAN, M. S. **Multi-aspect and Multi-class Based Document Sentiment Analysis of Educational Data Catering Accreditation Process**. Cloud & Ubiquitous Computing & Emerging Technologies (CUBE), 2013 International Conference on. [S.l.]: [s.n.]. 2013. p. 188-192.

23 LAWRENCE, D. K.; PENNOCK, D. M. **Mining the peanut gallery:** opinion extraction and semantic classification of product reviews. In Proceedings of the 12th international WWW conference. Budapest: [s.n.]. 2003. p. 519-528.

24 LU, Y.; C., Z.; N., S. **Rated aspect summarization of short comments**. In Proceedings of International Conference on World Wide Web. [S.l.]: [s.n.]. 2009.

25 VINCENT, N.; DASGUPTA, S.; ARFIN, S. M. N. **Examining the role of linguistic knowledge sources in the automatic identification and classification of reviews.** Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics. Sydney: [s.n.]. 2006. p. 611-618.

26 PANG et al. **Thumbs up? Sentiment classification using machine learning techniques**. EMNLP. [S.l.]: [s.n.]. 2002. p. 79-86.

27 WIEBE, B. R. F. . O. T. P. **Development and Use of a Gold-Standard Data Set for Subjectivity Classifications**. Annual Meeting of the Association for Computational Linguistics. [S.l.]: [s.n.]. 1999. p. 246-253.

28 WIEBE, J.; WILSON, T.; BELL, M. **Identifying Collocations for Recognizing Opinions**. Proceedings of the ACL Workshop on Collocation. Toulouse: [s.n.]. 2001.

29 YU, H.; HATZIVASSILOGLOU, V. **Towards Answering Opinion Questions:** Separating Facts from Opinions and Identifying the Polarity of Opinion Sentences. EMNLP-03, 8th Conference on Empirical Methods in Natural Language Processing. Sapporo, Japan: [s.n.]. 2003. p. 129-136.

30 KIM, S.; HOVY, E. **Determining the sentiment of opinions**. Proceedings of Interntional Conference on Computational Linguistic. [S.l.]: [s.n.]. 2004.

31 CARENINI, G.; PAULS, A. **Multi-document summarization of evaluative text**. Proceedings of the European Chapter of the Association for Computational Linguistics. [S.l.]: [s.n.]. 2006.

32 DING; LIU, B.; YU, P. **A holistic lexicon-based approach to opinion mining**. Proceedings of the Conference on Web Search and Web Data Mining. [S.l.]: [s.n.]. 2008.

33 HU, M.; LIU, B. **Mining and summarizing customer reviews**. Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. [S.l.]: [s.n.]. 2004.

34 SINGH, V. K. et al. **Sentiment analysis of movie reviews:** A new feature-based heuristic for aspect-level sentiment classification. Automation, Computing, Communication, Control and Compressed Sensing (iMac4s), 2013 International Multi-Conference on. [S.l.]: [s.n.]. 2013. p. 712-717.

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

36 MAUÁ, D. D.; COZMAN, F. G. Representing and classifying user reviews, 2009.
.

37 SANGANI, C.; ANANTHANARAYANAN, S. **Sentiment Analysis of App Store Reviews**. [S.l.].
.

38 HOVY, E.; LIN, C.-Y. Automated Text Summarization in SUMMARIST. **Advances in**
. **Automatic Text Summarization**, v. I, 1999.

39 YI, J. et al. **Sentiment analyzer:** extracting sentiments about a given topic using natural
. language processing techniques. Data Mining, 2003. ICDM 2003. Third IEEE International Conference on. [S.l.]: [s.n.]. 2003. p. 427-434.

40 FARHADLOO, M.; ROLLAND, E. **Multi-Class Sentiment Analysis with Clustering and Score**
. **Representation**. Data Mining Workshops (ICDMW), 2013 IEEE 13th International Conference on. [S.l.]: [s.n.]. 2013. p. 904-912.

41 NEVIAROUSKAYA, A.; PRENDINGER, H.; ISHIZUKA, M. **SentiFul:** Generating a reliable lexicon
. for sentiment analysis. Affective Computing and Intelligent Interaction and Workshops, 2009. ACII 2009. 3rd International Conference on. [S.l.]: [s.n.]. 2009. p. 1-6.

42 MOUTHAMI, K.; DEVI, K. N.; BHASKARAN, V. M. **"Sentiment analysis and classification**
. **based on textual reviews**. Information Communication and Embedded Systems (ICICES), 2013 International Conference on. [S.l.]: [s.n.]. 2013. p. 271-276.

43 HU, M.; LIU, B. Mining opinion features in customer reviews. **AAAI**, Chicago, v. 4, n. 4, p.
. 750-760, 2004.

44 SUBRAHMANIAN, V. S.; REFORGIATO, D. AVA: Adjective-Verb-Adverb Combinations for
. Sentiment Analysis. **Intelligent Systems, IEEE**, v. 23, n. 4, p. 43-50, Julho-Agosto 2008.

45 ISLAM, M. R. **Numeric rating of Apps on Google Play Store by sentiment analysis on user**
. **reviews**. Electrical Engineering and Information & Communication Technology (ICEEICT), 2014 International Conference on. [S.l.]: [s.n.]. 2014. p. 1-4.

46 GUZMAN, E.; MAALEJ, W. **How Do Users Like This Feature? A Fine Grained Sentiment**
. **Analysis of App Reviews**. Requirements Engineering Conference (RE), 2014 IEEE 22nd International. [S.l.]: [s.n.]. 2014. p. 153-162.

47 TECHCRUNCH. Study: Users Both Mostly Positive And Inconsistent In Reviewing iOS App
. Store Titles, 2012. Disponivel em: <http://techcrunch.com/2012/10/05/apple-app-store-reviews-reviewers-bias-positive/>. Acesso em: 10 Abril 2015.

48 STAT Counter. **Top 5 Desktop, Tablet & Console Browsers from Sept 2014 to Aug 2015**,
. 2015. Disponivel em: <http://gs.statcounter.com/>. Acesso em: 12 Junho 2015.

49 YANG, Y.; LIU, X. **A re-examination of text categorization methods**. Proceedings of SIGIR.

. [S.l.]: [s.n.]. 1999.

50 LEWIS, D. D. **Naive (Bayes) at forty:** The independence assumption in information retrieval.
. Proceedings of ECML. [S.l.]: [s.n.]. 1998.

51 JOACHIMS, T. **Text categorization with support vector machines:** Learning with many
. relevant features. Proceedings of ECML. [S.l.]: [s.n.]. 1998.

52 ZHANG, T.; OLES, F. J. Text categorization based on regularized linear classification
. methods. **Information Retrieval**, n. 4, p. 5-31.

53 DUDA, R. O.; HART, P. E. **Pattern classification and scene**. [S.l.]: Wiley and Sons, Inc., 1973.
.

54 HECKERMAN, D. **A tutorial on learning with bayesian network**. Microsoft Research.
. Redmon, WA. 1995. (MSR-TR-95-06).

55 ZHANG, H. **The optimality of Naive Bayes**. Faculty of Computer Science - University of New
. Brunswick. Fredericton, New Brunswick, Canada.

56 MANNING, C. D.; RAGHAVAN, P.; SCHÜTZE, H. **Introduction to Information Retrieval**. [S.l.]:
. Cambridge University Press, 2008.

57 TEAM, R. C. **R:** A Language and Environment for Statistical Computing. Vienna, Austria: R
. Foundation for Statistical Computing, 2015.

58 VENABLES, W. N.; RIPLEY, B. D. **Modern Applied Statistics with S**. 4. ed. New York:
. Springer, 2002.

59 SHARMA, D. Stemming Algorithms: A Comparative Study and their Analysis. **International**
. **Journal of Applied Information Systems**, New York, USA, v. 4, n. 3, 2012.

60 PARSONS, J. Martin Porter's stemmer for node.js. **Github**, 2015. Disponivel em:
. <https://github.com/jedp/porter-stemmer>. Acesso em: 22 ago. 2015.

61 CONTRIBUTORS, W. List of emoticons. **Wikipedia, The Free Encyclopedia.**, 2015. Disponivel
. em: <https://en.wikipedia.org/w/index.php?title=List_of_emoticons&oldid=685449036>.
Acesso em: 2 out. 2015.

62 SAIF, H.; MIRIAN FERNANDEZ, Y. H. H. A. **On Stopwords, Filtering and Data Sparsity for**
. **Sentiment Analysis of Twitter**. International Conference on Language Resources and
Evaluation. Reykjavik : ELRA. 2014.

63 LOPES, A. Portuguese stop words. **GitHubGist**, 2015. Disponivel em:
. <https://gist.github.com/alopes/5358189>. Acesso em: 22 set. 2015.

64 TEXTFIXER.COM. Common English Words. **TextFixer**, 2015. Disponivel em:

. <http://www.textfixer.com/resources/common-english-words.txt>. Acesso em: 25 set. 2015.

65 UMBEL, C.; ROB ELLIS, R. M. general natural language facilities for node. **GitHub**, 2012. . Disponivel em: <https://github.com/NaturalNode/natural>. Acesso em: 30 ago. 2015.

66 PRINCETON UNIVERSITY. About WordNet. **Wrodnet**, 2010. Disponivel em: . <http://wordnet.princeton.edu>. Acesso em: 23 jul. 2015.

67 GOMAA, W. H.; FAHMY, A. A. A survey of Text Similarity Approaches. **International Journal** . **of Computer Applications**, v. 68, n. 13, Abril 2013.

68 WINKLER, W. E. **String comparator metrics and enhanced decision rules in the** . **fellegisunter model of record linkage**. Proceedings of the Section on Survey Research. [S.l.]: [s.n.]. 1990. p. 354-359.

69 FORMAN, G.; SCHOLZ, M. **Apples-to-Apples in Cross-Validation Studies: Pitfalls in** . **Classifier Performance Measurement**. HP Laboratories. [S.l.]. 2009.

Anexo A - Código fonte

O código-fonte utilizado para o desenvolvimento do sistema deste Trabalho de Conclusão de Curso encontra-se disponível no seguinte endereço:

https://www.dropbox.com/sh/dwl8iwfluix8mye/AAD7G_Sp8L9rVjDJGqCLhwtFa?dl=0

A pasta organiza-se da seguinte forma:

- Pasta jsProject – contém o projeto em Nodejs responsável pela leitura do arquivo fornecido pela empresa Best Cool & Fun Games, separação do arquivo nos k-folds para a validação cruzada, algoritmo Naive-Bayes, algoritmo banco de palavras e todas as técnicas de pré-processamento de texto; possui também a função responsável por gerar o relatório CSV a partir
- Pasta scrapper – contém o código em javascript da extensão do Google Chrome descrito na seção 5.1 .
- Pasta opiniões – contém os sets de treinamento e validação utilizados na validação cruzada para treinamento e análise dos diferentes algoritmos
- Arquivo R_script – contém o script em R responsável pelo algoritmo de regressão logística multinomial.

Anexo B – Manual de Usuário do Sistema de Análise de Sentimentos

0) Pré-requisitos

O computador deve possuir o navegador Google Chrome instalado e Nodejs.

O programa deve ter sido previamente treinado com um algoritmo de aprendizado de máquina

1) Funcionalidade

O sistema possui três funções básicas

a) analyzeReview: a partir de um texto de avaliação (string) retorna o sentimento. Esta função está disponível na extensão Google Chrome e no projeto em nodejs.

b) replyReviewsSemiAutomatic: analisa a página da Google Play onde estão as avaliações dos usuários e classifica todas as revirews em positivo, negativo ou neutro. Caso alguma avaliação tenha o texto classificado como positivo mas o rating (estrelas) seja baixo, o sistema gera automaticamente uma mensagem perguntando porque o usuário não deu 5 estrelas.

c) generateCsvReport: a partir de um arquivo CSV no padrão da Google Play (contendo o texto de uma opinião e o idioma), o sistema gera um relatório com a quantidade de opiniões postivas, negativas e neutras.

2) Modo de usar extensão Google Chrome

- Baixar a pasta reviewsScrapper
- Abrir o Google Chrome
- Ir para a aba de extensões do Google Chrome “chrome://extensions/”,
- Ativar a opção “Modo de Desenvolvedor”
- Clicar na opção “Carregar extensão expandida...”
- Selecionar a pasta “reviewsScrapper” baixada
- Ir até o site da Google Play (https://play.google.com/apps/publish/)

- Entrar na seção “Ratings and Opiniões” do app
- Abrir o developer console do google Chrome (Ctrl + Shitf + I)
- Digitar ‘replyReviewsSemiAutomatic()’, chamando a função descrita no item anterior
- Clicar na página da web da google play
- O sistema vai analisar as opiniões e completar com o texto de resposta automático a partir das regras descritas no item anterior

3) Modo de usar projeto Nodejs

- Baixar a pasta jsProject
- Abrir o terminal e ir para o diretório ‘jsProject’
- No terminal digitar
    - $ node app.js <nome do arquivo CSV com as opiniões>
- O sistema vai retornar as estatísticas para o csv escolhido.