Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Mineração de opinião em textos opinativos utilizando algoritmos de classificação

Fernando Santos

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Orientador
Prof. Dr. Marcelo Ladeira

Brasília
2013

Universidade de Brasília — UnB
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Bacharelado em Ciência da Computação

Coordenador: Prof.ª Dr.ª Maristela Terto de Holanda

Banca examinadora composta por:

Prof. Dr. Marcelo Ladeira (Orientador) — CIC/UnB
Prof. Dr. Rommel Novaes Carvalho — CIC/UnB
Prof. Dr. Hercules Antonio do Prado — CIC/UCB

CIP — Catalogação Internacional na Publicação

Santos, Fernando.

Mineração de opinião em textos opinativos utilizando algoritmos de classificação / Fernando Santos. Brasília : UnB, 2013.
70 p. : il. ; 29,5 cm.

Monografia (Graduação) — Universidade de Brasília, Brasília, 2013.

1. mineração de opinião, 2. análise de sentimento, 3. mineração de texto, 4. processamento de dados em larga escala

CDU 004.4

Endereço: Universidade de Brasília
Campus Universitário Darcy Ribeiro — Asa Norte
CEP 70910-900
Brasília–DF — Brasil

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Mineração de opinião em textos opinativos utilizando algoritmos de classificação

Fernando Santos

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Prof. Dr. Marcelo Ladeira (Orientador)
CIC/UnB

Prof. Dr. Rommel Novaes Carvalho
CIC/UnB

Prof. Dr. Hercules Antonio do Prado
CIC/UCB

Prof.ª Dr.ª Maristela Terto de Holanda
Coordenador do Bacharelado em Ciência da Computação

Brasília, 20 de dezembro de 2013

Dedicatória

Dedico este trabalho à todos que me fizeram crescer de forma pessoal e profissional até hoje. Em especial, dedico a meus pais, que sempre fizeram o possível e o impossível para que eu tivesse uma boa educação e pudesse concluir essa importante etapa da minha vida.

Agradecimentos

Agradeço em primeiro lugar à Deus, pois foi ele quem me proporcionou tudo que tenho e a possibilidade de chegar onde estou hoje.

Agradeço aos meus pais e à toda minha família, por estarem sempre presentes em minha vida e por terem me apoiado em tudo que precisei.

Agraço ao meu orientador Marcelo Ladeira, pela atenção e o apoio que prestou durante os meus dois últimos anos de UnB. Mesmo quando eu estive no exterior, ele esteve sempre presente através de reuniões pela internet com muita dedicação.

Agradeço ao meu amigo Lucas Braga, que iniciou esta pesquisa comigo e me acompanha desde o início do curso.

Agradeço à CJR, a todos os professores do CiC e aos meus colegas de curso, que contribuíram bastante para a minha formação acadêmica.

Agradeço aos meus amigos mais próximos que estiveram sempre unidos e presentes desde os nossos tempos de colégio.

Resumo

Este trabalho descreve uma análise de mineração de opinião realizada sobre uma base de dados extraída da intenet e composta de comentários que contêm gírias, abreviações e outros jargões da internet. A mineração de opinião é a área de estudos que tenta identificar e classificar a subjetividade, como opiniões, emoções ou sentimentos na linguagem natural. Nesta pesquisa, 759 mil comentários em português foram extraídos da loja de aplicativos Google Play. Devido à grande quantidade de comentários, foram necessárias técnicas de processamento distribuído, envolvendo ferramentas poderosas como o Hadoop e o Mahout. O trabalho teve como principal constatação a verificação da baixa eficácia do pré-processamento em textos para a tarefa de mineração de opinião no domínio tratado. O trabalho também contribuiu com a criação de um corpus composto por 759 mil comentários e um dicionário de gírias, abreviações da Internet.

Palavras-chave: mineração de opinião, análise de sentimento, mineração de texto, processamento de dados em larga escala

Abstract

This work describes an opinion mining aplication over a dataset extracted from the *web* and composed of reviews with lots of internet slangs, abbreviations and typo errors. Opinion mining is a study field that tries to identify and classify subjectivity, such as opinions, emotions or sentiments in natural language. In this research, 759 thousand portuguese reviews were extracted from the app store Google Play. Due to the large amount of reviews, large scale processing techniques were needed, envolving powerful frameworks such as Hadoop and Mahout. The main contribution of this research was to verify the low efficiency of pre-processing techniques in the opinion mining task for the treated domain. The work also contributed to the creation of a corpus consisting of 759 thousand reviews and a dictionary of slangs and abbreviations commonly used in the Internet.

Keywords: opiniong mining, sentiment analysis, text mining, large scale data processing

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Pesquisas feitas em 2008 sobre hábitos de compra e comportamento de usuários americanos da *web*, abrangendo mais de 2000 adultos mostraram que:

- 81% dos usuários da internet já utilizaram a rede para pesquisar sobre um produto.
- 20% dos usuários da internet fazem essas pesquisas em um dia típico em que tenham acessado a internet.
- Dentre os leitores de recomendações online de restaurantes, hotéis e outros serviços, entre 73% e 87% informaram que tais relatos tiveram uma significante influência em sua compra.

Diante de tais dados, considerando a *web* como uma fonte de informação e de comunicação para uma parcela significativa das pessoas, é natural que se pense em uma maneira de identificar qual é o posicionamento majoritário das opiniões expressas nessa mídia, se tratando de um produto, uma empresa, um serviço ou até um candidato político. Cada opinião expressa na rede pode ser positiva (favorável) ou negativa (desfavorável) ou possuir um grau variando entre esses dois extremos. Uma vez que a quantidade de comentários na internet é muito grande, se faz necessário e muito útil a sumarização de vários comentários disponíveis em um simples resultado. Isso poupa tempo e ajuda os usuários a realizarem decisões com maior certeza.

Este trabalho tem como foco o desenvolvimento de uma aplicação de mineração de opinião. A mineração de opinião, ou análise de sentimento, como também é denominada, estuda formas de identificar e extrair informações subjetivas de dados não estruturados. A *web* está repleta de comentários, avaliações e textos em geral que expressam a opinião das pessoas. Dessa forma, pode-se pensar em dividir essas opiniões em categorias, como por exemplo, "positivo", "negativo" ou "neutro" e assim, tentar identificar, por exemplo, um comentário como sendo um comentário positivo ou negativo sobre alguém.

Neste trabalho, uma grande quantidade de comentários feitos sobre aplicativos para *smartphones* são analisados. Foram extraídos da loja de aplicativos online Google Play ao todo 759 mil comentários escritos na língua portuguesa. Todavia, esses comentários se caracterizam de forma diferente. Por serem extraídos de uma plataforma online onde os próprios usuários fazem os comentários e os mesmos não são revisados, há uma grande incidência de erros ortográficos, gírias, abreviações e jargões da língua portuguesa, o que tende a dificultar a análise quando comparado a um texto formalmente bem escrito.

Dessa forma, diferentes fases de pré-processamento são aplicadas na base com o intuito de melhorar a performance dos algoritmos de classificação utilizados.

Dada a grande quantidade de texto analisada, ferramentas de mineração de dados mais robustas se fazem necessárias. Assim, modelos de aprendizagem de máquina são criados utilizando-se do processamento distribuído em um *cluster* de computadores.

1.1 Objetivo

O objetivo deste trabalho é desenvolver uma aplicação de mineração de opinião capaz de avaliar a eficácia do pré-processamento de textos em uma base de textos do português formada por gírias, abreviações e termos comumente utilizados na internet. De modo a alcançar esse objetivo geral, o projeto possui os seguintes objetivos específicos:

- Avaliar a eficácia de métodos de pré-processamento de texto na tarefa de mineração de opinião.
- Classificar a orientação da opinião em textos da lingua portuguesa.
- Tratar o problema da mineração de opinião para uma grande quantidade de dados.

1.2 Estrutura do documento

Os próximos capítulos deste documento são compostos da seguinte forma:

- Capítulo 2: apresenta conceitos básicos da mineração de opinião e de processamento distribuído.
- Capítulo 3: apresenta a teoria envolvida nas técnicas de mineração de dados utilizadas nesse trabalho.
- Capítulo 4: apresenta os experimentos realizados, a fase de preparação, a metodologia e os resultados.
- Capítulo 5: apresenta as conclusões obtidas no decorrer desse projeto e sugestões de trabalhos futuros.

Capítulo 2

Fundamentação teórica

Este capítulo abordará os principais conceitos de mineração de opinião, indicando diferentes abordagens para esse propósito. Será tratado aqui a importância desse estudo para as pessoas e para as organizações bem como a própria definição de mineração de opinião. São mostrados termos específicos utilizados nesse campo de estudo e seus significados. Em seguida, faremos uma introdução ao conceito de processamento distribuído utilizando-se do modelo *map-reduce* implementado no framework Hadoop. É explicada a teoria envolvida no modelo *map-reduce* bem como o seu funcionamento em conjunto com um sistema de arquivos distribuído (*Distributed File System*). Por fim, são apresentados trabalhos correlatos e aplicações existentes sobre o domínio em questão.

2.1 Mineração de opinião

A mineração de opinião é um campo da mineração de dados relativamente recente, mas que tem despertado bastante interesse da comunidade científica. A mineração de opinião também é denominada como extração de opinião, análise de sentimento ou mineração de sentimento.

Bing Liu [26] define a mineração de opinião da seguinte maneira:

> "Dado um conjunto de documentos de texto avaliativos D que contêm opiniões (ou sentimentos) sobre um objeto, a mineração de opinião tem como objetivo extrair atributos e componentes do objeto que foi comentado em cada documento $d \in D$ e determinar se esses comentários são positivos, negativos ou neutros."

Analisando essa definição pode-se perceber alguns fatores que valem a pena ser comentados. Bing Liu se referencia a "opiniões" e também da mesma forma se referencia à "sentimentos". Esses são dois conceitos que apesar de parecerem distinstos, para a mineração de opinião, são equivalentes. Isso vale porque nem sempre a "opinião" que avaliamos em um texto ou frase é realmente uma opinião. Na frase "O passeio foi ótimo" é possível perceber claramente a expressão de uma opinião, que por sinal, é uma opinião positiva em relação ao objeto "passeio". Agora ao se observar a frase "A nova novela teve um nível de audiência muito baixo." é possível perceber que a frase não se trata de uma opinião e sim de um fato, porém mesmo assim, fica claro que esse fato possui uma orientação, ou seja um sentimento à respeito do objeto "nova novela" e que no caso é uma orientação negativa, uma vez que diz que poucas pessoas assistiram à novela.

2.1.1 Motivação

A nível mundial há um grande aumento do número de usuários da Internet e, em consequência, também há uma disponibilidade de grandes quantidades de informações novas na *web*. Grande parte da informação contida na *web* hoje é composta por opiniões e isso se deve principalmente ao advento de ferramentas tais como blogs, redes sociais e fóruns que permitem aos usuários leigos em informática expressarem suas opiniões na Internet.

A comunicação pode ser classificada de acordo com os participantes envolvidos. O telefone constitui um simples meio de comunicação um-para-um. Com o passar do tempo foram surgindo meios de comunicação um-para-muitos como, por exemplo, televisão, cinema, rádio, jornais, revistas e etc. A partir do século XXI, as mídias sociais e tecnologia da informação propiciaram o surgimento de novas formas de se comunicar, concretizando uma comunicação muitos-para-muitos. Andreas Kaplan e Michael Haenlein [22] definem mídias sociais como "um grupo de aplicações para Internet construídas com base nos fundamentos ideológicos e tecnológicos da Web 2.0, e que permitem a criação e troca de conteúdo gerado pelo usuário". As mídias sociais definem a interação interpessoal eletrônica produzindo conteúdo de muitos para muitos. Muitas vezes o termo mídias sociais e redes sociais são tratados da mesma forma, mas, segundo Giselle Stazauskas [36], as mídias sociais são na verdade as ações que acontecem dentro das redes sociais. As redes sociais são as formas de se representar os relacionamentos afetivos ou profissionais entre os seres enquanto o termo mídias sociais define a dinâmica desses relacionamentos.

A opinião de uma única pessoa reflete apenas a opinião dela e não uma tendência, mas o resumo da opinião de muitas pessoas sobre um assunto determinado gera um conhecimento que pode ser muito útil para qualquer organização que tenha interesse no assunto em questão. Por exemplo, um candidato em campanha política que está interessado em avaliar a sua popularidade ou uma produtora de um novo programa cultural que está interessado em saber a aceitação dele. Essa é uma tarefa que pode ser realizada analisando o que as pessoas têm comentado sobre o assunto nas mídias sociais.

Diversas organizações têm dado uma atenção especial para essa forma de pesquisar opiniões por ser mais econômica, rápida e dinâmica em relação à pesquisa de opinião convencional baseada em uma amostragem do público alvo. De fato, pelo método tradicional, se uma organização, por exemplo, quer saber a opinião do público sobre um novo celular lançado, poderá perguntar a opinião sobre aquele produto para alguns usuários selecionados segundo alguma técnica estatística. Porém isso demanda tempo, dinheiro e contato com as pessoas a serem entrevistadas, as quais podem não ter interesse em participar desse tipo de pesquisa. É justamente nesse contexto que a mineração de opinião tem uma importância potencial, pois pode substituir, com economia, a pesquisa tradicional por uma análise das opiniões expressas sobre o assunto em questão por usuários da internet, que por sua vez, participam da pesquisa sem mesmo saber que estão contribuindo.

2.1.2 Terminologia

A seguir serão passados alguns conceitos importantes sobre mineração de opinião que ajudarão no entendimento do mesmo.

- **Objeto e aspectos**: Um objeto O é uma entidade qualquer que esteja sendo analisada e/ou comentada, como por exemplo, um produto, um serviço, uma organização, um evento e etc. Formalmente, um objeto O é associado à um par (T,A) onde T representa uma hierarquia de subcomponentes de O e A representa o conjunto de atributos de O.

  Para facilitar o entendimento, suponha um objeto "restaurante". Um restaurante pode ter um grupo de subcomponentes como estacionamento, comida, ambiente e também um grupo de atributos como conforto e qualidade de atendimento. Da mesma forma, cada componente ou subcomponente tem também seus atributos, ou seja, o estacionamento pode ser considerado lotado ou perigoso.

  Em geral, essa hierarquia de componentes é representada como uma árvore onde o objeto O é a raiz. Porém, para diminuir a complexidade no tratamento de objetos, esse par de componentes e atributos é geralmente representado pelo termo "aspecto" [27].

  Vale ressaltar que um "aspecto" em uma frase pode ser explícito ou implícito. Este será explícito quando realmente estiver expresso na frase e implícito, caso contrário. Por exemplo, na frase "Essa casa é muito bonita." percebe-se que a casa é um aspecto explícito, pois ele está sendo citado na própria frase. Agora na frase "O mar está gelado" existe um aspecto implícito que não aparece na frase, que é a temperatura da água do mar.

- **Passagem de opinião sobre um aspecto**: A passagem de opinião sobre um aspecto de algum objeto analisado em um documento se dá por um grupo de sentenças presentes nesse documento, que expressam uma opinião negativa ou positiva sobre o aspecto.

- **Detentor da opinião**: O detendor da opinião é uma organização ou pessoa que possui a opinião em questão. Geralmente em dados providos de blogs, foruns e redes sociais, o detentor da opinião é o próprio autor do comentário. Isso nem sempre acontece em outras fontes como notícias jornalísticas, onde ocorrem situações em que o jornal expressa opiniões de outras pessoas sobre fatos. Por exemplo na seguinte frase o detentor da opinião é o presidente: "O presidente mostrou sua satisfação para com o lançamento do novo programa federal".

- **Opinião ou sentimento**: O psicólogo Klaus Scherer define uma tipologia para os estados afetivos divida em cinco tipos [35]:

  - Emoção (nervoso, triste, orgulhoso ...)
  - Humor (abatido, animado, irritado ...)
  - Posturas interpessoais (amigável, frio, insolente ...)
  - **Atitudes (gostando, odiando, amando, desejando ...)**
  - Traços de personalidade (nervoso, ansioso, hostil ...)

  Essa divisão é bem interessante para à análise de sentimento, pois através dela, podemos verificar onde as opiniões ou sentimentos estão presentes. Dentre as cinco divisões de estados afetivos, o sentimento está presente justamente nas atitudes ou

posturas (tradução mais coerente para o portugûes). Quando se detecta uma postura, deve-se detectar quem detém aquela postura (detentor da opinião), sobre quem é aquela postura (aspecto) e qual a orientação dessa postura (negativo, positivo ou neutro).

Pelos exemplos citados acima, pode-se perceber que todos os termos citados em Atidudes (gostando, odiando, amando, desejando) tem uma relação entre duas ou mais entidades, ou seja, é sempre necessário haver a fonte do sentimento e o destino (aspecto), enquanto que os outros tipos de estados afetivos nem sempre dependem de uma relação dupla. Os estados nervoso, triste, irritado, por exemplo, só dependem necessariamente de uma fonte.

- **Opinião direta**: É uma opinião que faz referências sobre um aspecto ou objeto expressando diretamente um sentimento positivo ou negativo sobre ele.
- **Opinião comparativa**: É uma opinião que compara o objeto em questão com algum outro objeto similar, enfatizando as semelhanças ou diferenças dos dois objetos que estão sendo comparados. Em geral, sentenças comparativas são expressas utilizando um advérbio de comparação.
- **Sentenças subjetivas e objetivas**: Uma sentença que trata da subjetividade do autor ou da interpessoalidade do indivíduo, pode ser definida como uma sentença que possua subjetividade. Sentenças objetivas são sentenças factuais, que expressam fatos, acontecimentos concretos, enquanto que sentenças subjetivas expressam opiniões, sentimentos ou considerações sobre alguma coisa. A subjetividade está relacionada com o interior do ser humano.
- **Sentenças opinativas**: São sentenças que, independentemente da subjetividade ou objetividade, expressam opiniões, mesmo que implicitamente. Na maioria dos casos, sentenças opinativas são também subjetivas, mas pode-se facilmente verificar casos onde estão expressas opiniões mesmo em um contexto factual. Por exemplo, na frase "Esse carro é muito lento" é citado um fato sobre o motor do carro, porém mesmo assim, percebe-se que o autor da frase está opinando negativamente sobre o mesmo.

2.1.3 Conceitos

Níveis de classificação

Varios autores [21] [18] [26] dividem a classificação de sentimentos em três níveis: documento, sentença e aspecto. No presente trabalho é feita uma classificação em nível de documento, ou seja, um documento inteiro é considerado como uma unidade básica de dados. Dessa forma, todas as opiniões do documento são sumarizadas em um nível de granularidade mais geral, assumindo que o documento trate apenas de um único objeto, o que pode ser verdade ou não.

Ao analisar o documento em um nível mais aprofundado, pode-se realizar uma classificação em nível de sentença. Essa classificação é feita em duas partes [18]: classificação da subjetividade e classificação do sentimento. Inicialmente, cada sentença é classificada em objetiva ou subjetiva para determinar se a mesma exprime ou não uma opinião. A

seguir, a sentença é classificada como positiva ou negativa, caso seja uma sentença subjetiva. Algumas premissas geralmente são assumidas nesse nível de classificação, como por exemplo, a de que o objeto tratado na sentença é conhecido e que cada sentença possui somente uma opinião sobre esse objeto.

Em um último nível de granularidade, é feita a classificação baseada em aspecto. Nesse nível assume-se que o documento possui diversos aspectos e opiniões. O foco então, é relacionar as opiniões presentes com seus respectivos aspectos. Por exemplo, na frase "A casa está velha, porém o bairro é muito agradável" o autor deprecia a "casa" porém tem uma opinião positiva à respeito do "bairro". A tarefa de classificação em nível de aspectos é feita inicialmente com a extração dos aspectos presentes no documento e posteriormente determinando a orientação correspondente a cada aspecto.

Abordagens

Existem três diferentes abordagens para a implementação de uma tarefa de mineração de opinião [21]: métodos baseados em aprendizagem de máquina (utilizado neste trabalho), métodos léxicos e métodos baseados em análise linguística. A seguir falaremos um pouco de cada um deles.

A mineração de opinião através de aprendizagem de máquina é feita utilizando-se de algoritmos de classificação. Os documentos são convertidos em matrizes que representam a frequência de cada palavra pertencente ao documento (denominadas matrizes de termo-documento). Essa é uma tarefa de aprendizagem supervisionada, portanto, é necessária uma base de treino com documentos rotulados com a respectiva orientação (positiva, negativa ou neutra). Um algoritmo de classificação é treinado a partir dessa base para que então possa ser utilizado para classificar novos documentos.

Grande parte da performance desse processo de classificação depende da conversão do texto em uma matriz termo-documento. Diversos pré-processamentos e filtros podem ser aplicados durante essa fase. Palavras são corrigidas, abreviações são expandidas, palavras muito frequêntes e sem significado (*stopwords*) são removidas e etc. Além disso, novos termos denominados *n-grams* podem ser criados e tratados como novas "palavras" do documento. Um *3-gram*, como por exemplo "água de coco", é um termo de 3 palavras que aparecem em sequência no documento. Ainda entre os termos restantes, algoritmos de seleção de atributos podem ser executados para selecionar os termos que melhor representam cada classe de orientação.

Métodos léxicos utilizam bases de palavras previamente definidas com uma certa polaridade, como por exemplo, a base SentiWordNet [7]. Nesta abordagem, cada palavra do documento é avaliada quanto à sua polaridade, e então, diferentes métodos podem ser aplicados para determinar a classificação geral do documento, sentença ou aspecto [13].

Métodos baseados em análise linguística utilizam características sintáticas das palavras e sentenças para determinar a orientação de um texto, ou seja, cada palavra precisa ser classificada de acordo com a sua função sintática. Estudos anteriores [37] já mostraram, por exemplo, que adjetivos e advérbios são classes gramaticais bastante subjetivas que carregam opinião. Nesta abordagem, a classificação é feita a partir das classes gramaticais de cada palavra e da posição em que elas aparecem. Ferramentas denominadas *POS Taggers* (*Part-of-speech Taggers*) conseguem rotular as palavras de acordo com a sua posição e atuação na estrutura da frase. Essas análises podem ser morfológicas (analisando

cada termo individualmente), sintáticas (analisando o termo inserido em uma sentença) ou a combinação das duas (morfossintática). Essa área de estudo que visa estabelecer uma comunicação entre a linguagem natural e a linguagem de máquina denomina-se processamento de linguagem natural (ou linguística computacional).

Dificuldades na mineração de opinião

Textos opinativos, em geral, se diferenciam de outros textos por passarem a informação de forma mais sutil. Considere uma tarefa mais simples, como a categorização, onde um texto é atribuído à uma classe de acordo com o domínio do texto, como por exemplo, esporte, política, culinária e etc. Em tal tarefa, podemos pensar em palavras-chave frequêntes para cada domínio. Em um texto sobre política, por exemplo, palavras como "presidente", "parlamentar" e "república" representam fortes indícios de que o texto trata sobre política. O mesmo não necessáriamente ocorre para a mineração de opinião, onde muitas vezes, opiniões podem ser expressas através de figuras de linguagem, como por exemplo, metáforas, ironias e etc. Considere os seguintes exemplos:

- "Nossa, que belo filme! Caí no sono em 15 minutos..."
- "Se você está pensando em comprar esse jogo, desista e vá ler um livro!"

Para nós, seres humanos, é fácil perceber a orientação negativa de tais comentários. Entretanto, percebe-se que opiniões são expressas mesmo sem o uso de palavras com orientação negativa. Na verdade, o primeiro comentário, possui inclusive uma palavra positiva, "belo", que torna o problema ainda mais complexo.

Outro problema a ser tratado é a identificação da subjetividade. Muitos frases em um texto não possuem ou não expressam nenhuma opinião. Porém, essa identificação de frases não subjetivas é bem complexa. Além disso, orientações podem ser expressas de forma indireta através de frases objetivas. Considere novamente os exemplos a seguir:

- "A Petrobras dispensará 20 mil funcionários em programa de redução de custos."
- "A bateria do Iphone não tem uma vida longa."

Apesar de expressarem fatos, conseguimos perceber que o fato de uma empresa demitir funcionários para reduzir custos, em geral, não é algo positivo para a empresa. Da mesma forma, sabemos que um celular com bateria de vida curta é algo ruim. Entretanto, essas são inferências que concluímos a partir do nosso raciocínio. Inserir esse tipo de "conhecimento" em uma máquina como o computador é uma tarefa muito complexa.

A tempolaridade é outro fator que influência na mineração de opinião. Em alguns domínios, as opiniões à respeito de algum assunto podem mudar com o tempo. Um produto pode ser lançado com pouco sucesso em determinado momento, e futuras versões do mesmo produto podem melhorar. Mesmo possuindo versões diferentes, muitas vezes as opiniões serão tratadas como se fossem sobre o mesmo produto. Um político, por exemplo, pode receber diversos elogios e após a descoberta de seu envolvimento com corrupção pode receber muitas avaliações negativas. Dessa forma, o contexto temporal em que uma avaliação ou comentário é obtida deve ser levado em consideração.

Muitas vezes a tarefa de identificar o objeto de uma avaliação não é simples. Como saber se a frase "Lula viaja esta semana para o Japão" faz referência à um presidente

do Brasil ou à um molusco marinho? O reconhecimento de entidades (*named entity recognition*) é outra tarefa importante dessa área de estudo.

Outros problemas mais simples também são relevantes. Textos, em geral, não possuem nenhuma estrutura. Não necessariamente seguem um padrão formal de frases constituído por sujeito e predicado. Textos extraídos da internet, em geral, de mídias sociais, possuem muitos erros gramaticais, gírias e abreviações.

2.2 Processamento distribuído

Big Data analysis é um termo que tem recebido muita atenção nos últimos anos. O termo se refere à analise de imensas quantidades de dados que estão disponíveis hoje através da internet. Em muitos casos, os dados e a análise a ser realizada fornece uma oportunidade para explorar o paralelismo no processamento dos dados.

A seguir são citados dois importantes exemplos explorados por Jeffrey Ulman et al. [33]:

- Ordenação de páginas *web* por importância, onde são feitas operações de multiplicação de matrizes de dimensões na casa dos bilhões.
- Pesquisa em redes de "amigos" em sites de relacionamento, que utilizam grafos com milhões de nós e bilhões de arestas.

Para tratar problemas de tal magnitude, novos sistemas com paradigmas diferentes foram desenvolvidos. Esses sistemas utilizam o paralelismo não através de um "mega computador" mas sim através de um *cluster* de computadores, ou seja, uma grande quantidade de simples (e de baixo custo) computadores conectados em rede. Esses sistemas são baseados em um novo formato de arquivo, um formato de arquivo distribuído (*distributed file systems*). Sistemas de arquivo distribuídos fornecem blocos de disco maiores, melhorando a performance de leituras e gravações em disco.

Sobre esse novo sistema, uma nova forma de programação chamada *map-reduce* é utilizada. Implementações de *map-reduce* viabilizam a computação de diversos cálculos sobre grandes quantidades de dados (*large-scale data*) de forma eficiente e tolerante à falhas [33].

2.2.1 Sistema de arquivos distribuído

Até então, a computação distribuída era feita, principalmente, através de computadores especiais com vários processadores e *hardware* especializado. Entrentando, o aumento da quantidade de dados espalhados pela *web* fez crescer o processamento distribuído entre diferentes nós de computação, o que nos trouxe à uma nova geração de sistemas de computação. Esses sistemas trabalham de forma simultânea com vários computadores e por isso devem possuir certa tolerância à falhas, dado que qualquer um dos computadores pode falhar a qualquer momento.

Essa nova arquitetura de computação, chamada de *cluster* é organizada de forma que computadores são agrupados em blocos (*racks*) e blocos são conectados em uma rede Ethernet por um *switch*. A figura 2.1[33] ilustra essa arquitetura enquanto a figura 2.2[1] mostra um cluster de computadores da Yahoo.

[1] http://kiiro.io/yahoos-hadoop-cluster/

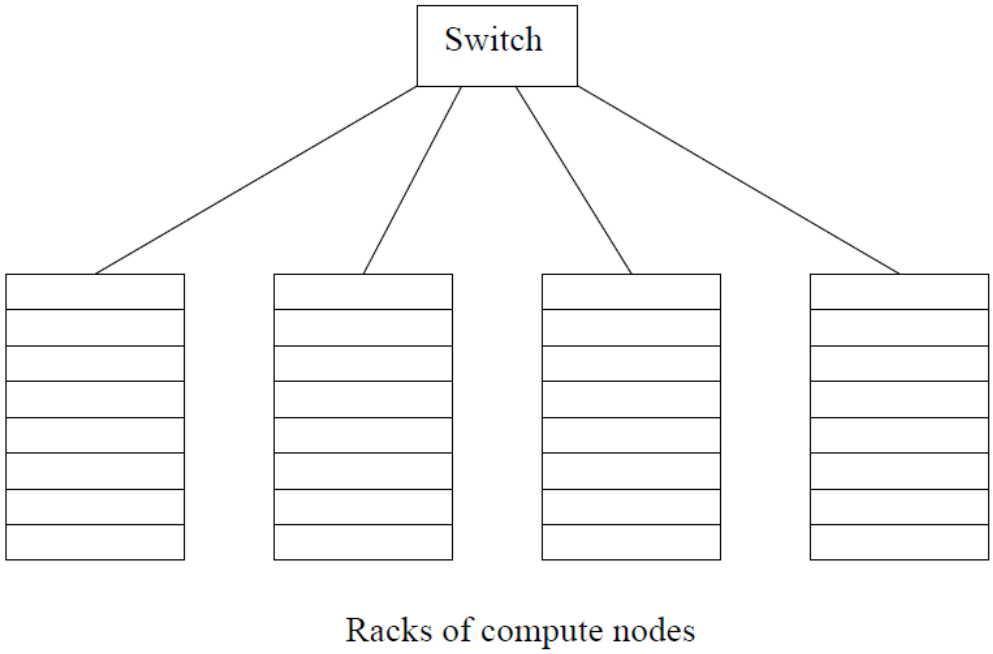

Figura 2.1: Arquitetura de um *cluster* de computadores

Figura 2.2: *Cluster* de computadores da Yahoo utilizando o Hadoop

Alguns sistemas de arquivos existentes e utilizados hoje são:

- *Google File System* (GFS): O primeiro a ser desenvolvido.
- *Hadoop Distributed File System* (HDFS): Um sistema de arquivos distribuído de código aberto distribuído pela fundação Apache. (Utilizado neste trabalho)

Em situações reais, *hardwares* podem falhar, fazendo parar a computação de um único computador ou até de todo um bloco. Em cálculos complexos, que podem durar horas, a

simples falha de um computador poderia fazer com que todo o cálculo fosse refeito. Por isso alguns procedimentos são necessários [33]:

- Arquivos devem ser salvos de forma redundante. Se os arquivos não forem salvos em vários locais, a falha do computador que contém o arquivo iria prejudicar todo o processamento.
- A computação deve ser dividida em tarefas. Dessa forma, se uma tarefa não é executada devido à alguma falha, as outras tarefas podem continuar sua execução de forma independente.

O sistema de arquivo utilizado possui algumas características específicas:

- Arquivos podem ser enormes (terabytes, por exemplo).
- Arquivos são raramente atualizados. Em geral, são lidos para alguma computação e possivelmente concatenados com mais dados.
- Arquivos são divididos em pedaços (*chunks*), comumente de tamanho igual a 64 MB, para que possam ser espalhados replicados e espalhados por diferentes nós de computação.

2.2.2 Map-Reduce

Map-reduce é uma forma de programação implementada em vários sistemas, como por exemplo, o Hadoop. O usuário somente precisa programar duas funções, *Map* e *Reduce*. O sistema irá gerenciar a execução em paralelo, decidindo quando cada uma dessas funções será utilizada, além de tratar a possibilidade de alguma dessas tarefas falhar. Jeffrey Ulman et al [33] explicam a execução do *map-reduce* da seguinte forma:

- Tarefas de mapeamento (*map*) recebem um ou mais *chunks* do sitema de arquivos distribuído. Essas tarefas convertem cada *chunk* em uma sequência de pares chave-valor. Esses pares são produzidos a partir dos dados de entrada e são determinados de acordo com o código fornecido pelo usuário na função *Map*.
- Os pares chave-valor de cada tarefa de mapeamento são coletados por um controlador e ordenados por suas chaves. As chaves são divididas entre todas as tarefas de redução (*reduce*), de forma que todo par chave-valor com a mesma chave é passado para a mesma tarefa de redução.
- Tarefas de redução (*reduce*) trabalham com uma chave por vez, e combinam, de alguma forma, todos os valores associados àquela chave. A maneira como esses valores serão combinados é determinada pelo usuário através da função *Reduce*.

A figura 2.3[33] ilustra um esquema da execução do *map-reduce*.

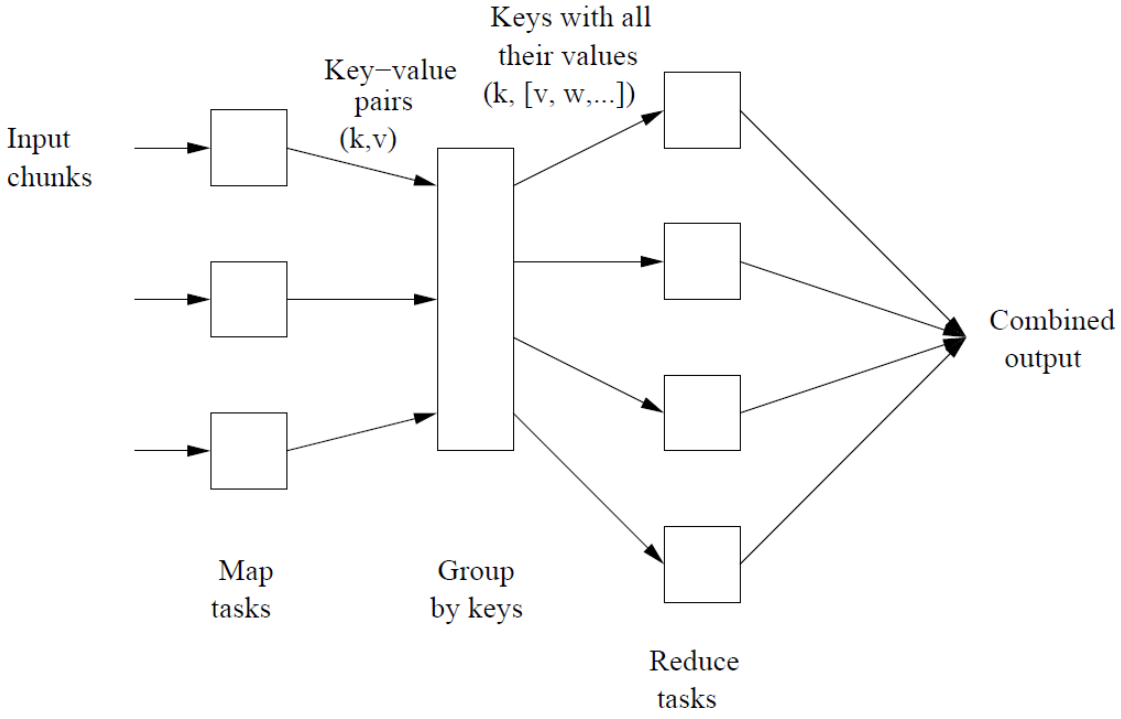

Figura 2.3: Execução do *map-reduce*

Tarefas de mapeamento

Tarefas de mapeamento recebem como argumento dados a serem processados e produzem como saída um ou mais pares chave-valor. Apesar da palavra chave, as chaves não precisam necessariamente ser únicas. O padrão desse par chave-valor é arbitrário e é definido pelo usuário através da função *Map*.

Para facilitar o entendimento, usaremos o clássico exemplo adotado para explicação do *map-reduce*: contar a frequência de cada palavra em um conjunto de documentos. Nesse exemplo, chaves são *strings* (as palavras a serem contadas) e os valores são inteiros (as frequências). A função *Map* para esse exemplo irá ler um documento, separá-lo em sequências de palavras $p_1, p_2, \ldots, p_n$ e atribuir a frequência 1 para cada palavra. Dessa forma, a saída da tarefa de mapeamento será:

$$(p_1, 1), (p_2, 1), \ldots, (p_n, 1)$$

Durante a execução, uma mesma tarefa de mapeamento pode ser responsável por ler vários documentos. Assim, caso a palavra p apareça m vezes entre vários documentos, a saída da tarefa possuirá m pares $(p, 1)$.

Agrupamento

O agrupamento é feito sempre da mesma forma, independentemente das funções *Map* e *Reduce*. O controlador irá aplicar uma função de *hashing* sobre as chaves recebidas como saída da tarefa de mapeamento, fazendo com que chaves iguais sempre sejam atribuídas à uma mesma tarefa de redução. Assim, após a finalização das tarefas de mapeamento,

o controlador irá passar para as funções de redução os pares em um formato de chave e lista de valores. Isto é, para cada chave c, a entrada para a tarefa de redução responsável pela chave c é um par $(c, [v_1, v_2, \ldots, v_n])$, onde $(c, v_1), (c, v_2), \ldots, (c, v_n)$ são todos os pares chave-valor com chave c.

Tarefas de redução

As tarefas de redução recebem como argumento pares com uma chave e uma lista de valores associados à esta chave. No exemplo de contagem das palavras, a função *Reduce* irá somar todos os valores de cada chave e retornará uma sequência de pares (p, n), onde p é uma palavra que aparece ao menos uma vez entre todos os documentos e n é a quantidade de vezes que p aparece.

Exemplo

Para entender a real utilização do *map-reduce*, será também exemplificado brevemente um cálculo em um domínio mais comum. O exemplo descrito por Steve Krenzel [4] explica como realizar a tarefa de encontrar amigos em comum em uma rede social, como por exemplo, o Facebook. Suponha a seguinte lista de amizades:

- A -> B C D (lê-se A é amigo de B, C e D)
- B -> A C D E (lê-se B é amigo de A, C, D e E)

A função de mapeamento será responsável por transformar essas listas de amizades em um atributo do tipo chave e valor, onde a chave é a ligação de amizade entre duas pessoas e o atributo é a lista de amigos. Dessa forma, obtêm-se os seguintes pares da função de mapeamento:
Para a entrada A -> B C D:

- (A B) -> B C D
- (A C) -> B C D
- (A D) -> B C D

Para a entrada B -> A C D E:

- (A B) -> A C D E
- (B C) -> A C D E
- (B D) -> A C D E
- (B E) -> A C D E

O framework é responsável por agrupar os pares por meio de suas chaves, antes de passá-los para as funções de redução:

- (A B) -> (A C D E) (B C D)

Por fim, a função de redução é responsável por calcular a interseção das listas de amigos de cada agrupamento, resultando na lista de amigos em comum entre cada amizade:

- (A B) -> (C D) (lê-se: os amigos em comum de A e B são C e D)

Através desse simples exemplo, pode-se entender melhor como uma tarefa qualquer, como a de identificação de amigos em comum em um grafo de amizades, pode ser executada utilizando-se do paradigma *map-reduce*.

Organização das tarefas

Durante a execução do *map-reduce*, o usuário cria um processo mestre (*master*) e um determinado número de processos trabalhores (*workers*). Um *worker* pode ficar responsável por tarefas de mapeamento ou de redução, mas não ambos. O processo mestre é responsável por criar tarefas de mapeamento e redução e atribuí-las aos *workers*. O processo mestre também controla a situação de cada *worker* (ociosa, ocupada, etc). A quantidade de tarefas é definida pelo usuário e é de grande importância para a performance do *cluster*. As tarefas de redução são limitadas, pois para cada redução, é necessário um arquivo independente. Se a quantidade de tarefas de redução for muito grande, serão necessários muitos arquivos e isso poderá ocasionar em custos desnecessários de leitura (*overhead*). A quantidade de tarefas de mapeamento dependem da quantidade de dados e da capacidade de processamento de cada nó de computação. A figura 2.4[33] mostra em mais detalhes como funciona um programa executando *map-reduce*.

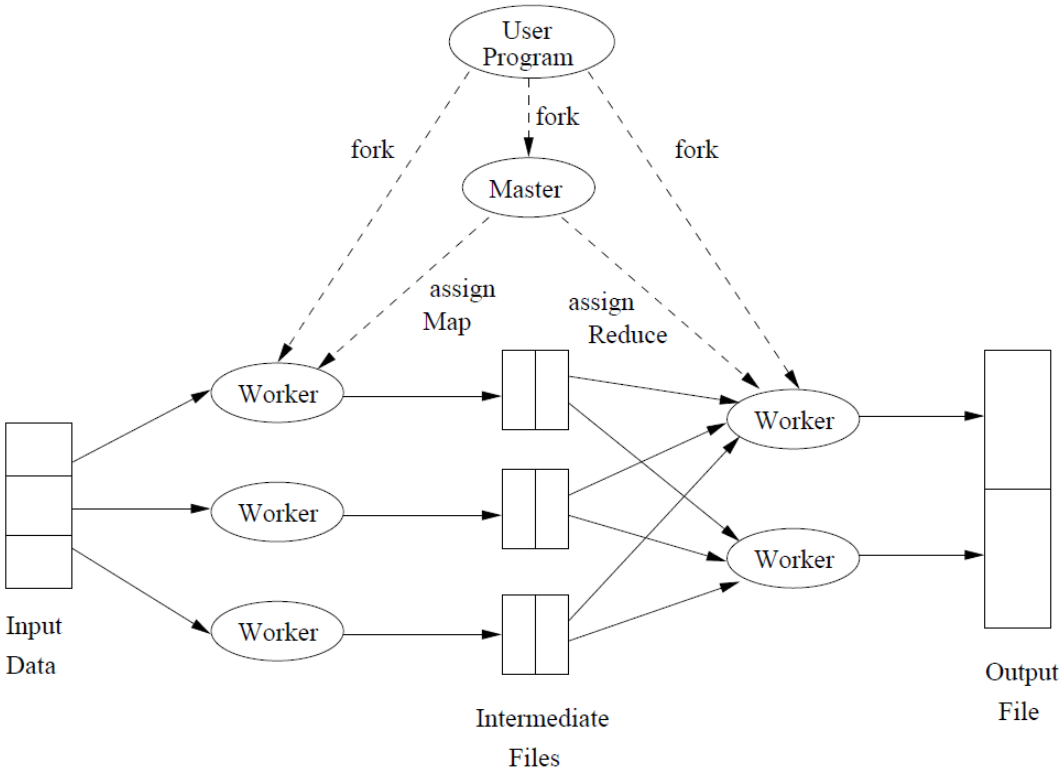

Figura 2.4: Execução detalhada do *map-reduce*

2.2.3 Hadoop e Mahout

O Hadoop[2] é um framework de código aberto para processamento em larga escala de grandes quantidades de dados em *clusters* de computadores de baixo de custo. O projeto é desenvolvido e mantido pela fundação Apache[3]. O framework é composto pelos seguintes módulos:

- Hadoop Common: conjunto de utilitários que suportam os outros módulos
- Hadoop Distributed File System (HDFS): sistemas de arquivos distribuído responsável pelo armazenamento de dados entre os computadores e por proporcionar alta largura de banda para comunicação dentro do *cluster*
- Hadoop YARN: framework responsável pela gestão dos recursos de computação dentro do *cluster* bem como gerência e agendamento dos processos
- Hadoop MapReduce: implementação de um modelo de programação para processamento de dados em larga escala

A estrutura de processamento fornecida pelo Hadoop gerou oportunidades para que diversos outros projetos fossem criados apoiando-se em sua estrutura. Entre esses projetos, temos o Mahout[4]: um conjunto de implementações de aprendizagem de máquina utilizando-se do paradigma *map-reduce*. O foco principal desse projeto é criar implementações de alta performance para:

- Classificação
- Clusterização
- *Collaborative Filtering*[5]

2.3 Trabalhos correlatos

O OpSys [28], resultado de um trabalho de graduação, trata da orientação semântica de textos de notícias extraídos de portais de investimentos para analisar notícias sobre organizações com ações negociadas na bolsa de valores. O autor extrai notícias das páginas de jornais como o Estado de São Paulo, Folha de São Paulo, GoogleNews, Globo, ou o portal Exame e utiliza técnicas de processamento de linguagem natural (PLN) para definir a polaridade para a organização que está sendo analisada. Esse mesmo autor analisou a polaridade de notícias do mercado de ações [17], onde são buscadas palavras que façam referência diretamente ao nome de ações como, por exemplo, PETR4 (Petrobrás PN). O autor utilizou algoritmos como o PMI (Pointwise Mutual Information) para identificar a polaridade das notícias. O PMI é uma medida utilizada na teoria da informação para

[2] http://hadoop.apache.org/

[3] http://www.apache.org/

[4] http://mahout.apache.org/

[5] *Collaborative Filtering* é uma técnica utilizada em sistemas de recomendação. Em geral, utilizam-se grandes quantidades de dados para encontrar padrões e similaridades em usuários de algum sistema. Dessa forma, recomendações podem ser feitas para usuários similares. Tome como exemplo a recomendação de livros feitas pela Amazon ou a recomendação de filmes feitas pelo Netflix.

medir a relação entre palavras dentro de um texto com a comparação entre as probabilidades de duas palavras estarem juntas ou separadas no texto, ajudando a identificar uma associação verdadeira [17].

Também utilizando um algoritmo similar ao PMI, em [39] é apresentada uma ferramenta para extração de *n-grams* de tamanhos variáveis úteis à mineração de textos. O algoritmo analisa as posições em que as palavras ocorrem com relação às outras e a especificidade de cada uma para decidir se determinado par de palavras forma realmente uma palavra composta com significado interessante. Algoritmos como esse são úteis para a mineração de opinião visto que através deles novas palavras e expressões podem surgir e ajudar na performance da classificação. Por exemplo, as palavras "show", "de" e "bola" separadamente não representam nenhuma subjetividade, porém juntas ("show de bola") se tornam uma expressão que indica forte orientação positiva à algum objeto.

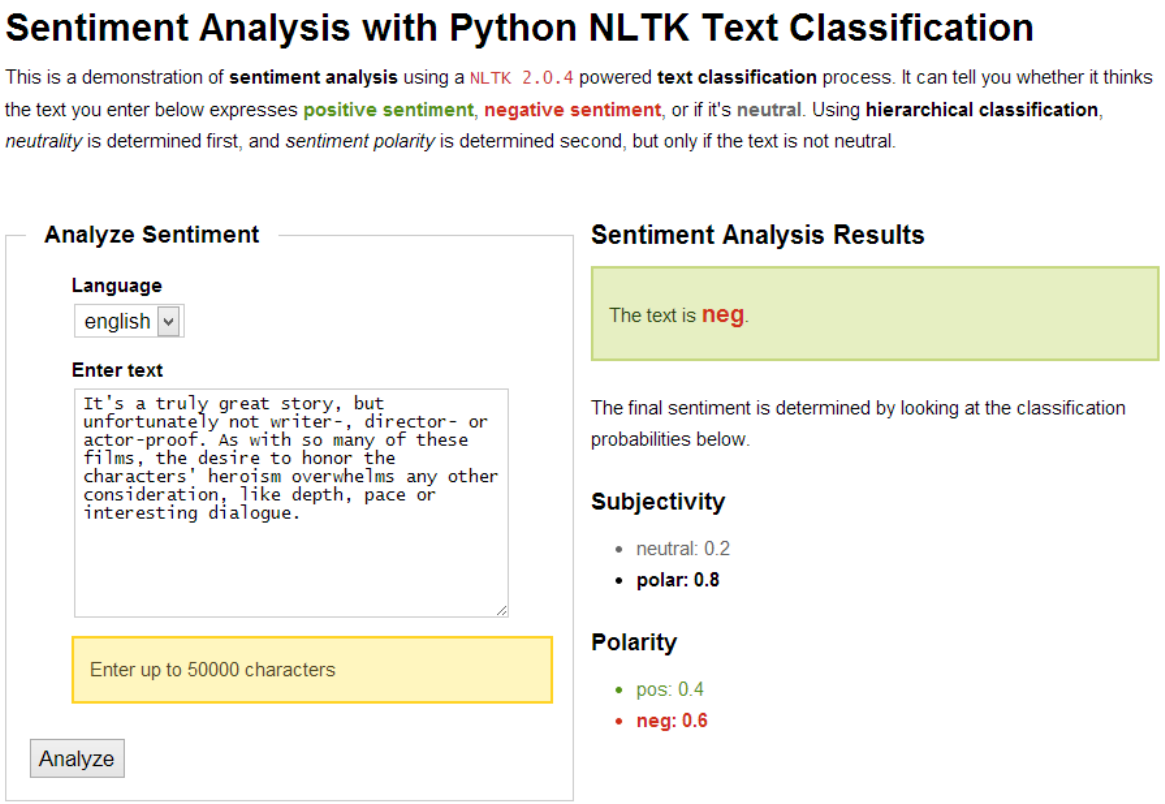

Figura 2.5: Classificação de avaliação de filme com o Python NLTK Text Classification

O site *Sentiment Analysis with Python NLTK Text Classification* [6] permite que os usuários escrevam um texto que será analisado. O analisador foi treinado sobre um *corpus* disponibilizado pelas pesquisadoras Bo Pang e Lilliam Lee [32]. Esse *corpus* é baseado em revisões de comentários sobre filmes variados. Os comentários são encontrados em uma base de dados do portal IMDb[6] que agrupa informações de filmes de todo o mundo. Devido ao fato das ferramentas serem treinadas com revisões de filmes, os resultados são mais acurados para textos no formato de resenhas de filmes e devem necessariamente ser escritos na língua inglesa. A figura 2.5 mostra uma avaliação de filme sendo analisada.

[6]http://www.imdb.com/

A ferramenta é capaz de identificar, além da orientação, uma nota para a polaridade e para a subjetividade do texto apresentado. Em sua pesquisa sobre análise de sentimentos aplicada a avaliação de filmes, as autoras Bo Pang e Lilliam Lee [32] fizeram diversos testes utilizando-se de algoritmos como *Naive Bayes*, *Maximum Entropy* e *Support Vector Machine*, concluindo que o *Support Vector Machine* obteve a melhor performance entre os três.

Baseado em avaliações de restaurantes retiradas da plataforma Twitter[7], Farley Fernandes [18] fez um estudo similar. A implementação do sistema *HowGood*, proposta pelo autor, analisa os comentários e os classifica entre positivos e negativos. Para a realização dessa tarefa são utilizadas técnicas de processamento de linguagem natural além de alguns algoritmos da teoria da informação, como o *Pointwise Mutual Information* (PMI). O trabalho atingiu um resultado relevante, sendo possível fazer uma classificação com boa acurácia. Entretanto, essa implementação é dependente de trabalho manual. Durante a classificação das avaliações de restaurantes, o próprio usuário do sistema deve, manualmente, classificar quais são os aspectos a serem avaliados (comida, ambiente, estacionamento, etc ...) e classificar grande parte dos adjetivos e advérbios entre positivos e negativos. Em seu trabalho, Farley [18] utiliza um padrão apresentado por [37] para identificar conjuntos de palavras que representem aspectos e seus respectivos atributos, como por exemplo, "belo carro". O padrão apresentado, tabela 2.1, é uma lista de classes gramaticais que quando ocorridas em sequência, tem grande chance de representar um par de atributo e objeto. A tabela 2.2 mostra o significado de cada um dos rótulos.

Primeira palavra	**Segunda palavra**	**Terceira palavra**
JJ	NN ou NS	Qualquer uma
RB, RBR, RBS	JJ	Não NN, nem NNS
JJ	JJ	Não NN, nem NNS
NN ou NNS	JJ	Não NN, nem NNS
RB, RBR ou RBS	VB, VBD, VBN ou VGB	Qualquer uma

Tabela 2.1: Padrão para extração de pares atributo e objeto

Como continuação do trabalho de Farley [18], Nelson Silva [13] realizou um estudo baseado no sistema HowGood com o intuito de automatizar o processo de classificação manual das polaridades das palavras. Para isso, foi utilizada a base *SentiWordNet* [7], que consiste em um conjunto de mais de 100 mil palavras em inglês já classificadas com notas de 0 a 1 para as orientações positivas, negativas e neutras. O sistema implementado é denominado *BestChoice* e analisa os mesmos comentários do *HowGood*, que são em português. Para decidir a polaridade das palavras encontradas, uma tradução através do tradutor automático da Google[8] é feita e o resultado é buscado no banco de palavras do *SentiWordNet*. O autor provou que mesmo automatizando grande parte do processo, conseguiu, em geral, a mesma acurácia nas classificações da abordagem manual utilizada no sistema *HowGood*.

[7] www.twitter.com

[8] www.google.com

Rótulo	Descrição
JJ	Adjetivo
NN	Substantivo
RB	Advérbio
VB	Verbo no infinitivo
NNS	Substantivo no plural
RBR	Advérbio comparativo
RBS	Advérbio superlativo
VBD	Verbo no passado
VBN	Verbo no particípio passado
VBG	Verbo no gerúndio

Tabela 2.2: Rótulos gerados por um *POS-Tagger*

Em [40], uma base de *tweets* foi avaliada utilizando técnicas de mineração de texto. A partir da base de *tweets* disponíveis publicamente já rotulados, foram selecionados *tweets* relacionados à Microsoft e ao Iphone. Os algoritmos *J48* e *Naive Bayes* foram aplicados utilizando-se de uma lista de atributos composta por 931 palavras positivas e 1838 palavras negativas. Na mesma abordagem, os autores incluíram *emoticons* positivos e negativos nessa lista de atributos e refizeram os testes. Foi concluído que algoritmos baseados em árvores de decisão, como o *J48*, obtiveram performance superior ao *Naive Bayes* e que a adição de emoticons ao conjunto de atributos trouxeram impactos levemente negativos à performance dos classificadores.

Groot R. [15] aplicou diversas técnicas de mineração de dados em uma base de tweets. Novamente, diversos classificadores foram testados e foi concluído que o SVM obteve a melhor performance. Algoritmos de aprendizagem não supervisionada também foram testados nesse trabalho, porém sem resultados expressivos. Groot R. fez testes com *n-grams* e obteve melhores resultados com *uni-grams* do que com *n-grams* de ordens mais altas. Por fim, concluiu-se que apesar de obter resultados satisfatórios quando testados dentro da amostra, a generalização do modelo para novos tweets nunca vistos obteve resultados insatisfatórios.

Em [23], grandes quantidades de *tweets* (até 300.000) foram analisados no intuito de construir um sistema de mineração de opinião de larga escala em tempo real. Para o treinamento dos classificadores, foram utilizados *tweets* com *emoticons*, partindo do princípio de que *emoticons* felizes, em geral, aparecem em comentários positivos, e *emoticons* negativos em comentários negativos. As ferramentas utilizadas para o processamento em larga escala foram as mesmas das utilizadas nesta pesquisa: Mahout e Hadoop. Os autores concluíram que foi possível obter um bom tempo de resposta com as tecnologias envolvidas e atingiram acurácia de 73.7% na classificação da polaridade dos tweets.

Também sobre uma grande base de *tweets*, Lin et al. [25] realizaram estudos em análise de sentimento. Foram utilizados o Hadoop em conjunto com o framework Pig. O Pig é uma plataforma de alto nível para criação de programas utilizando o *map-reduce* com o Hadoop. O pig fornece uma linguagem chamada Pig Latin que abstrai o paradigma de map-reduce à um nível similar à linguagens de consulta como SQL (*Structured Query Language*). Dessa forma, a plataforma fornece suporte para análise de grandes quantidades de dados e também fornece alguns algoritmos de aprendizagem de máquina, como a regressão logística. Em sua pesquisa [25], Lin et al. perceberam que após uma certa quantidade, o ganho na performance de classificações não é tão alto. Foram feitos testes com quantidades de 1 milhão, 10 milhões e 100 milhões de *tweets*. A diferença no resultado desses testes foi muito pouca. De 1 para 10 milhões houve um aumento da acurácia dos classificadores de aproximadamente 0.5%, enquanto que de 10 para 100 milhões esse aumento foi ainda menor.

Capítulo 3

Mineração de dados

Segundo Berry M. e Linoff G. [9], mineração de dados é a exploração e a análise, por meio automático ou semi-automático, de grandes quantidades de dados, a fim de descobrir padrões e regras significativos. Outra definição mais simples encontrada em [38] classifica a mineração de dados como a aplicação de algoritmos específicos para extração de padrões em dados. Independentemente da definição, o objetivo final de qualquer processo de mineração de dados se baseia em auxiliar na tomada de decisão.

De forma similar à mineração de dados, a mineração de texto é o processo de descoberta de conhecimento e padrões em bases de texto. Entre as tarefas de mineração de texto temos categorização e clusterização de textos, reconhecimento de entidades, sumarização de documentos, análise de sentimentos (foco desse trabalho) e muitas outras. A mineração de textos surgiu com o intuito de aplicar as mesmas técnicas de mineração de dados em dados e informações não-estruturadas, considerando-se o alto nível de complexidade envolvido quando os dados estão representados em forma de linguagem natural.

Nas seções seguintes, alguns conceitos sobre mineração de dados e texto bem como etapas desses processos utilizadas nessa pesquisa são explicados com mais detalhes.

3.1 O Corpus

Um *corpus*, na linguística, é um conjunto de documentos textuais utilizado como base de dados para uma análise. O foco dessa pesquisa é a identificação da polaridade em comentários sobre aplicativos para dispositivos móveis disponíveis na Google Play (`https://play.google.com/store`), loja de aplicativos oficial do sistema Android, da Google.

O *corpus* é composto por comentários extraídos dessa plataforma. Na primeira extração realizada, foram extraídos no total 190.095 comentários, todos da lingua portuguesa. Passados alguns meses, uma nova extração realizada gerou um total de 759.176 instâncias de comentários. Percebe-se que são comentários em sua totalidade do português brasileiro, mas também alguns comentários do português de Portugal foram capturados devido às similaridades dos dois idiomas.

Os comentários analisados pertencem a um mesmo domínio de conhecimento. Analisar comentários de diferentes domínios, como produtos de beleza e peças de automóveis, geraria resultados conflitantes, pois os termos que definem boas características para um não necessariamente são bons para o outro. Assim, para a composição do *corpus*, só

foram extraídos comentários sobre aplicativos de jogos. Dessa forma, tem-se a certeza de que o objeto sobre o qual o comentário está sendo feito é, de forma direta ou indireta, o jogo. Caso contrário, se analisássemos um *corpus* composto por, por exemplo, notícias de jornal, cada notícia teria um objeto diferente, o que tornaria a análise mais complexa.

A loja de aplicativos Google play é o meio oficial do usuário do sistema Android adquirir aplicativos. O sistema Android é voltado para dispositivos móveis, roda sobre um núcleo Linux e, em setembro de 2012, possuía uma média de 1.3 milhões de novos dispositivos eletrônicos sendo ativados por dia. A loja Google play, antiga Android Market, atingiu em setembro de 2013 a marca de 850 mil aplicativos ativos. A figura 3.1[1] mostra um gráfico do crescimento do total de aplicativos android no ano de 2013.

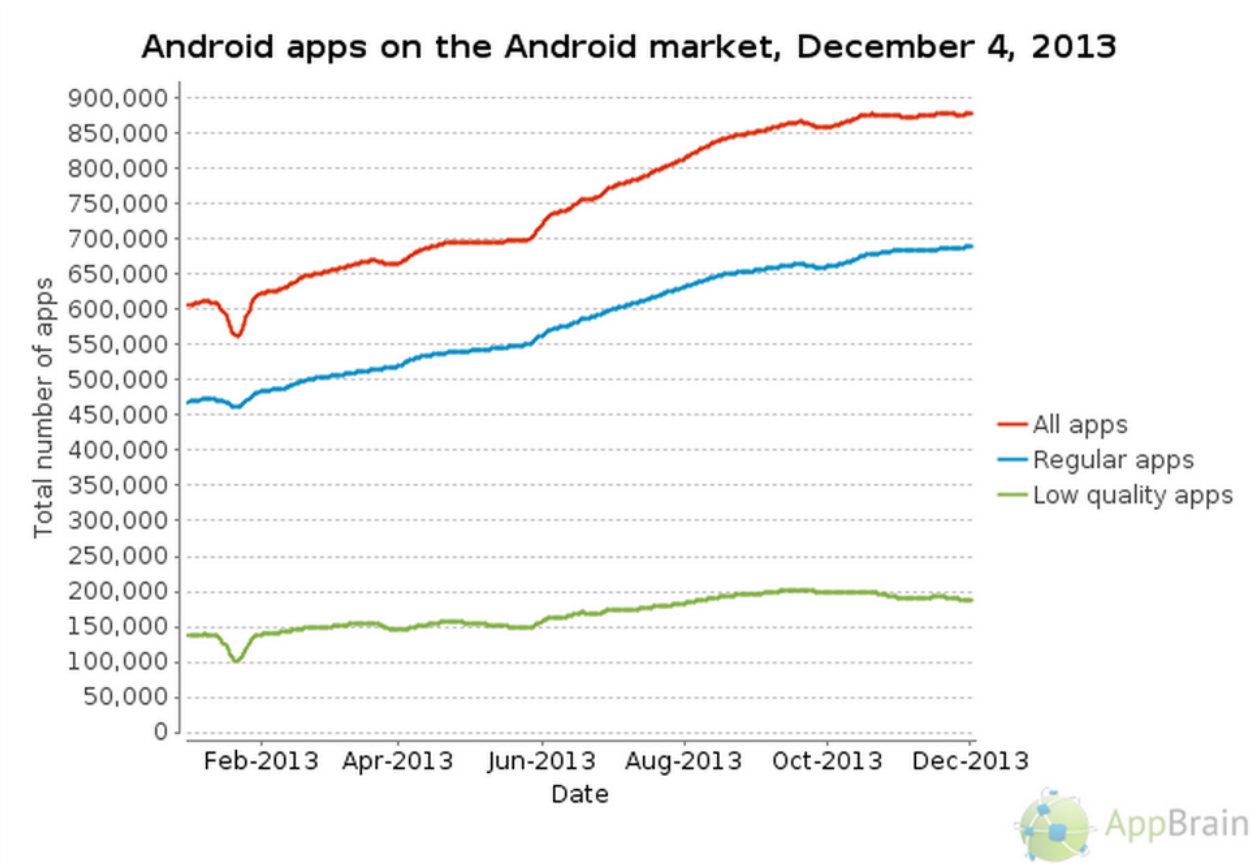

Figura 3.1: Quantidade de aplicativos android em 2013

Como a oferta de aplicativos é muito grande, o usuário necessita de informações para avaliar se determinado aplicativo será uma boa aquisição. Uma das principais fontes de onde se pode verificar a satisfação sobre uma aquisição no Google play são os comentários de outros usuários que já adquiriram e testaram os aplicativos. Quando um usuário instala um aplicativo em seu dispositivo, seja ele gratuito ou pago, ele pode deixar um comentário escrito e uma avaliação num ranking de 1 a 5 estrelas. O comentário escrito é separado em título, com no máximo 50 caracteres e o comentário contendo a opinião do usuário, com no máximo 1200 caracteres. Como não há nenhum tipo de restrição quanto ao que o usuário pode escrever, existem muitos comentários com erros de grafia e utilização de

[1] http://www.appbrain.com/stats/

gírias e neologismos. A figura 3.2[2] mostra exemplos de comentários existentes no Google Play.

Figura 3.2: Exemplos de comentários sobre o jogo Angry Birds

Nos exemplos mostrados pela figura 3.2 pode-se verificar os tipos de gírias e abreviações presentes nos comentários. Durante a fase de pré-processamento dos dados esse problema é tratado de forma a tentar minimizar o impacto de gírias e erros ortográficos nos resultados finais.

É natural desejar conhecer a opinião geral de todos os comentários, tanto para o usuário que quer adquirir o aplicativo quanto para a empresa que o programou e almeja maximizar suas vendas. Essa tarefa se torna complexa quando levamos em conta a quantidade de comentários, a má formação dos textos, a subjetividade, a ironia e demais problemas inerentes à tarefa de mineração de opinião.

3.2 Pré-processamento

O pré-processamento é uma etapa extremamente importante para o processamento de dados, principalmente em aplicações que utilizam dados não estruturados, como textos. O crescimento de pesquisas em dados do Twitter, Facebook e outras plataformas, gerou de forma proporcional o crescimento no foco em pré-processamento. Grande parte do texto encontrado em tais plataformas não são bem formulados, isto é, possuem erros, gírias, abreviações e ignoram muitas regras gramaticais. Dessa forma, o pré-processamento se

[2]https://play.google.com/store

faz necessário para corrigir possíveis erros e até dar mais estrutura ao texto, de forma a melhorar a performance de futuras ferramentas que venham a utilizar o texto pré-processado.

Apesar da importância do pré-processamento, essa é uma tarefa simples. Palavras digitadas erroneamente podem ser facilmente corrigidas com um dicionário. Abreviações também podem ser substituídas pelo seu significado real através de um dicionário pré-definido. Outra técnica comumente utilizada é a remoção de *stopwords*. *Stopwords* são palavras que estão presentes em grande parte dos documentos analisados. Além de palavras comuns da própria linguagem natural, como artigos e preposições, também são consideradas *stopwords* palavras sem subjetividade e muito utilizadas dentro do domínio tratado. Por exemplo, em uma análise sobre avaliações de filmes, palavras como "filme", "cinema" e "sinopse" seriam descartadas. A remoção de tais palavras, diminui a dimensionalidade do problema, diminuindo o tempo de resposta dos algoritmos de classificação e aumentando sua performance.

De forma similar, números e pontuações podem ser removidos dos documentos. Palavras acentuadas são transformadas em não acentuadas. Outra técnica bastante utilizada no pré-processamento é o *stemming*. Essa técnica reduz cada palavra ao seu radical, ou seja, ao morfema que corresponde ao sentido básico da palavra. Por exemplo, palavras como "adorei", "adorou" e "adoro" são todas reduzidas à "ador". Essas técnicas tendem a mesclar palavras que tem o mesmo significado em uma só, aumentando a frequência daquele termo e diminuindo a dimensão geral do problema.

A tarefa de pré-processamento é muito dependente da linguagem natural que está sendo analisada. Por isso, dados pertencentes à um único idioma se tornam menos complexos. A maioria dos estudos de mineração de opinião atualmente é feita sobre a língua inglesa.

3.3 Extração de atributos

A extração de atributos é o processo que envolve a extração de características dos dados que serão úteis para a classificação. Muitas vezes, usar todos os dados é algo muito custoso para a tarefa de classificação e por isso é necessário extrair os atributos que melhor representam cada classe. Na mineração de textos, é importante definir como cada palavra será convertida para um valor numerico. Alguns conceitos importantes relacionados à essa tarefa serão descritos a seguir.

3.3.1 Matriz termo-documento

A matriz termo-documento é a estrutura utilizada para representar documentos de forma estruturada. Uma matriz termo-documento consiste em uma matriz matemática $M_{d,t}$ composta de d documentos e t termos, onde o valor $m_{d,t}$ da matriz representa o peso (*weight*) do termo t para o documento d.

$$M = \begin{bmatrix} m_{documento1,termo1} & m_{documento1,termo2} & \cdots \\ m_{documento2,termo1} & m_{documento1,termo2} & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix}$$

Existem três principais formas de se calcular o valor para de $m_{d,t}$:

- **TP (Term presence)**: Recebe 1 caso o termo esteja presente no documento, ou 0 caso contrário.
- **TF (Term frequency)**: Recebe a quantidade n de vezes que o termo aparece no documento.
- **TF-IDF (Term Frequency - Inverse Document Frequency)**: Recebe um valor relativo ao **TF** porém depreciado de acordo com a quantidade **DF (Document frequency)** de documentos que contenham o termo . Esse valor é dado por:

$$TF - IDF = TF * Log(N/DF)$$

onde N representa o número total de documentos.

Essa medida leva em consideração o seguinte raciocínio: Se um termo t que ocorre em d, aparece em diversos documentos, então t não é um bom represente de d. Por outro lado, se o termo t aparecer várias vezes em d e aparecer poucas vezes em outros documentos, então t é um bom representante do documento d, e por isso recebe um valor TF-IDF alto.

Em [21], estudos comparativos entre os três métodos acima são realizados durante a tarefa de classificação de avaliações de filmes. E, em geral, o TF-IDF obteve maior performance.

3.3.2 N-grams

O uso de *n-grams* é um método comumente utilizado durante o processo de extração de atributos. Um *n-gram* nada mais é do que um sintagma, isto é, um grupo de palavras que aparecem sequencialmente no texto. O *n-gram* mais simples possível é um *unigram*, ou seja, uma única palavra. *N-grams* de ordem $n = 2$ (*bigram*) e $n = 3$ (*trigram*) também são bastante utilizados na literatura. Quanto maior for o valor de n maior é a quantidade de termos extraídos do documento, sendo necessário que somente os k termos mais representativos sejam selecionados para a matriz termo-documento.

O tamanho ideal de *n-grams* e o método de extração de atributos em uma tarefa de mineração de texto dependem principalmente dos dados e do domínio tratado. Bo Pang e Lilliam Lee [32], por exemplo, obtiveram melhores resultados utilizando-se de *unigrams* com **TP** (*Term presence*) do que ao utilizar *unigrams* com **TF** (*Term frequency*). Elas também realizaram testes com *bigrams* porém não obtiveram nenhuma melhora na performance. Já em [20], durante a classificação de *tweets* com *emoticons*, foi obtida uma melhor performance através da combinação de *unigrams* e *bigrams*. Parece intuitivo o fato de que a utilização de *n-grams* de ordens superiores nos leva a uma performance melhor, porém isso nem sempre se concretiza. Para um bom resultado final, são necessários diferentes testes com várias configurações possíveis, visto que cada tipo de dado e domínio se ajustam de forma diferente à cada modelo.

3.4 Classificação

Classificação consiste em atribuir uma classe a um documento a partir dos seus dados de entrada. Essa classificação é feita através de algoritmos de aprendizagem. Esses algoritmos conseguem "criar conhecimento" sobre o domínio tratado através de dados de entrada. Quando esses dados de entrada já possuem suas classes atribuídas dizemos que essa é uma aprendizagem supervisionada. Dessa forma, o algoritmo consegue criar uma relação entre as características de cada entrada com a sua classe final, gerando um "conhecimento" para fazer previsões sobre entradas que ainda não possuem uma classe definida. Esses dados de entrada já classificados são chamados de base de treino enquanto que os documentos que serão classificados posteriormente ao treino são denominados como a base de teste.

Quando a base de treino não possui uma classificação pré-definida são utilizados algoritmos de aprendizagem não-supervisionada. Nesse caso, os algoritmos tentam agrupar entradas com características semelhantes e classificá-los por grupos. Esse tipo de classificação é chamada de clusterização.

Típicos exemplos de classificação são por exemplo, determinar se um email é spam ou não baseado em seu conteúdo. Determinar se uma transação online é fraudulenta ou legítima baseado nos dados da transação ou determinar se um tumor é maligno ou benigno baseado em características das células.

Nesta pesquisa foram utilizados principalmente algoritmos de aprendizagem supervisionada. Alguns deles são explicados em mais detalhes nas subseções seguintes.

3.4.1 Logistic Regression

Regressão logística, apesar de ser denominada como regressão, é um tipo de modelo probabilístico de classificação. Para entender melhor o funcionamento desse algoritmo é necessário o entendimento de uma regressão linear. Uma regressão linear simples é um modelo que relaciona uma variável y com uma variável explicativa x através de uma hipótese h. Dada uma sequencia de pontos (x, y) a regressão linear define uma reta $h(x) = \theta_1 + \theta_2 * x$ com a mínima distância possível de todos os pontos (x, y). Essa minimização é feita através da média dos erros ao quadrado, onde o erro é a distância mínima entre cada ponto e a reta. A figura 3.3[3] ilustra um típico exemplo de regressão linear, utilizado para avaliar o preço de uma casa (variável y) baseando-se no seu tamanho (variável x).

Apesar de ser muito utilizada, a regressão linear não se aplica bem para casos de classificação. Um simples motivo para esse fato é que a regressão linear $h(x)$ pode gerar como saída qualquer valor numérico e para uma tarefa de classificação binária, por exemplo, precisamos de saídas categóricas como 0 e 1.

A regressão logística pode ser definida da seguinte forma:

$$h(x) = g(\theta^T x)$$

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

[3] http://www.libresoft.es/node/325

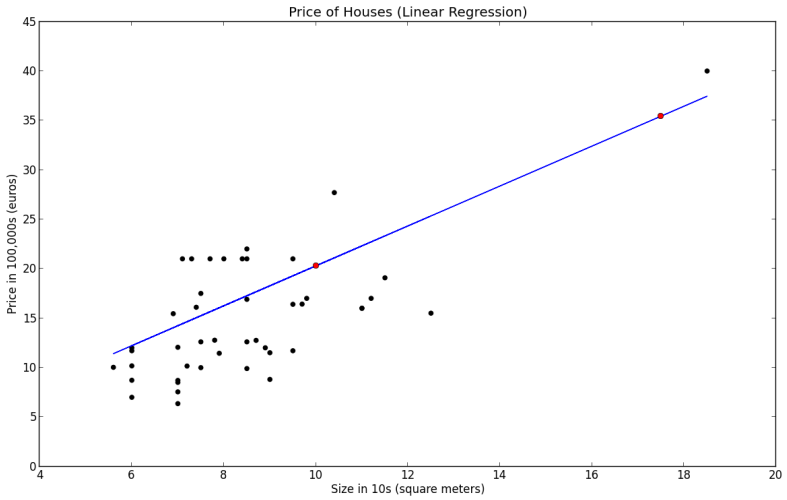

Figura 3.3: Regressão linear sobre o exemplo de precificação de casas

, onde g representa uma função logística (ou *Sigmoid*) e θ representa um vetor de parâmetros. Uma função logística é uma função que tem como saída sempre valores entre 0 e 1, ilustrada na figura 3.4[4].

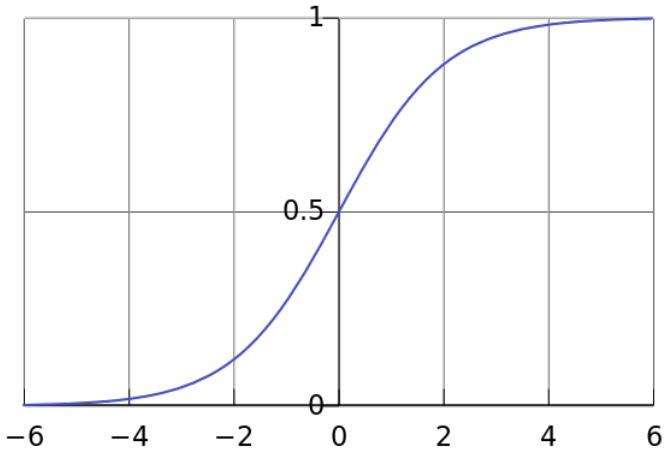

Figura 3.4: Função logística

Para entender a ideia desse algoritmo. Suponha o conjunto de dados não linear apresentado na figura 3.5[5] e a seguinte hipótese:

$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2)$$

[4]http://deeplearning.net/software/theano/tutorial/examples.html
[5]https://www.coursera.org/course/ml

$$\theta = \begin{bmatrix} -1 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

O algoritmo terá saída $y = 1$ caso $g(\theta^T x) >= 0.5$. Se $g(\theta^T x) >= 0.5$ então $\theta^T x > 0$. Portanto, ao substituir θ, temos que o algoritmo irá gerar a saída $y = 1$ se $-1+x_1^2+x_2^2 >= 0$. Essa fórmula se equivale à $x_1^2 + x_2^2 = 1$, ou seja, uma circunferência de raio = 1. Dessa forma, o algoritmo consegue criar uma função não linear que divide a base de treino em dois grupos.

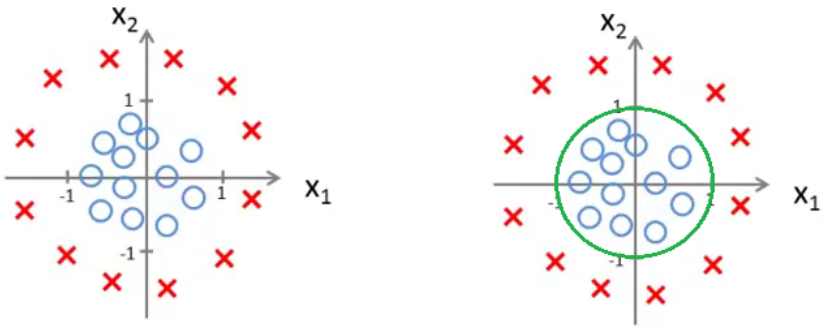

Figura 3.5: Curva de decisão gerada pela regressão logística

O vetor de parâmetros θ é iniciado com algum valor arbitrário e a cada nova entrada da base de treino, θ é ajustado com novos valores de forma a minimizar a função de custo do algoritmo. Esse método para estimar valores para o vetor é chamado de *maximum likelihood estimation*. Assim, ao passar por todos os exemplos de treino, θ possuirá os valores que melhor se ajustam a hipótese $h(x)$ proposta. Uma descrição mais matemática desse modelo pode ser encontrada em [10].

3.4.2 Naive Bayes

Naive Bayes é um algoritmo probabilístico de classificação relativamente simples. O algoritmo se baseia no teorema de bayes (equação 3.1) e recebe o nome *naive* (ingênuo) por assumir algumas fortes hipóteses de independência entre as variáveis.

$$P(\mathbf{A}|\mathbf{B}) = \frac{P(\mathbf{B}|\mathbf{A})P(\mathbf{A})}{P(\mathbf{B})} \tag{3.1}$$

No contexto de classificação de documentos, queremos atribuir uma classe c para um documento d. Portanto, o que precisamos saber é a probablidade de uma classe c, dado um documento d:

$$P(\mathbf{c}|\mathbf{d}) = \frac{P(\mathbf{d}|\mathbf{c})P(\mathbf{c})}{P(\mathbf{d})}$$

Em um classificador, a classe $c_{NB} \in C$, onde C representa o conjunto possível de classes, é selecionada como a melhor classe caso ela maximize a sua probabilidade dada o documento d:

$$c_{NB} = arg\,max_{c \in C} P(\mathbf{c}|\mathbf{d}) \tag{3.2}$$

Aplicando-se o teorema de Bayes,

$$c_{NB} = arg\,max_{c \in C} \frac{P(\mathbf{d}|\mathbf{c})P(\mathbf{c})}{P(\mathbf{d})} \tag{3.3}$$

podemos eliminar $P(d)$ dado que a probabilidade do documento é constante para todas as classes.

$$c_{NB} = arg\,max_{c \in C} P(\mathbf{d}|\mathbf{c})P(\mathbf{c}) \tag{3.4}$$

A probabilidade de um documento d pertencer à classe c ($P(d|c)$) pode ser expressa como a probabilidade de cada um de seus atributos dada a classe c: $P(x_1, x_2, \ldots, x_n|c)$. Portanto podemos reescrever a equação 3.4 da seguinte forma:

$$c_{NB} = arg\,max_{c \in C} P(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n|\mathbf{c})P(\mathbf{c}) \tag{3.5}$$

Para diminuir a quantidade de parâmetros calculados, assumimos algumas hipóteses, que apesar de incorretas, simplificam bastante o modelo e ainda assim nos geram resultados satisfatórios

1. Assume-se que a posição em que as palavras aparecem no texto não importa;
2. Assume-se que as probabilidades de cada atributo $P(x_i|c_j)$ são independentes dada a classe c.

Assumindo tais hipóteses temos que:

$$P(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n|\mathbf{c}) = P(\mathbf{x}_1|\mathbf{c}) * P(\mathbf{x}_2|\mathbf{c}) * \ldots * P(\mathbf{x}_n|\mathbf{c}) \tag{3.6}$$

$$c_{NB} = arg\,max_{c \in C} P(\mathbf{c}_j) \prod_{x \in X} P(\mathbf{x}|\mathbf{c}) \tag{3.7}$$

A partir da equação 3.7, podemos enteder melhor como um classificador *Naive Bayes* irá selecionar uma classe. Suponha um documento d que constitui os atributos $x_1, x_2, \ldots, x_n$ e duas classes c_1 e c_2. Cada atributo x_i pode ser uma palavra do documento. Para cada classe, $P(c_j)$ é simplesmente a frequência de documentos da classe c_j e $\prod_{x \in X} P(x|c_j)$ é o produto da probabilidade de cada palavra aparecer em documentos da classe c_j.

Assim como na regressão logística, o treinamento desse modelo é feito através do método de *maximum likelihood estimation*, ou seja, as probabilidades são calculadas simplesmente através das frequências das palavras. Por exemplo, as probabiliades de uma classe c_j e de uma palavra p_i na classe c_j são dadas, respectivamente, pelas equações 3.8 e 3.9.

$$P(c_j) = \frac{\text{Número de documentos da classe } c_j}{\text{Número total de documentos}} \tag{3.8}$$

$$P(p_i|c_j) = \frac{\text{Número de ocorrências de } p_i \text{ em documentos da classe } c_j}{\text{Número total de palavras em documentos da classe } c_j} \tag{3.9}$$

3.4.3 Support Vector Machine

Support Vector Machine (SVM) é um método matemático capaz de classificar documentos de forma supervisionada [11][30][15]. Para simplificar o entendimento, suponha uma base de treino composta por somente dois atributos e duas classes possíveis. Nesse caso, teremos um plano bidimensional, e por isso, em um caso ideal, é possível a separação linear entre essas duas classes. Em outras palavras, é possível traçar uma reta que separe perfeitamente o grupo de uma classe do grupo da outra classe. Na verdade, podemos traçar várias retas, porém a melhor reta possível é a que separa os dois grupos de forma a maximizar a distância entre a reta e todos os pontos do plano. A figura 3.6 mostra um plano bidimensional sendo separado por uma reta ideal. Os vetores sobre os pontos vermelhos são os chamados *Support Vectors*.

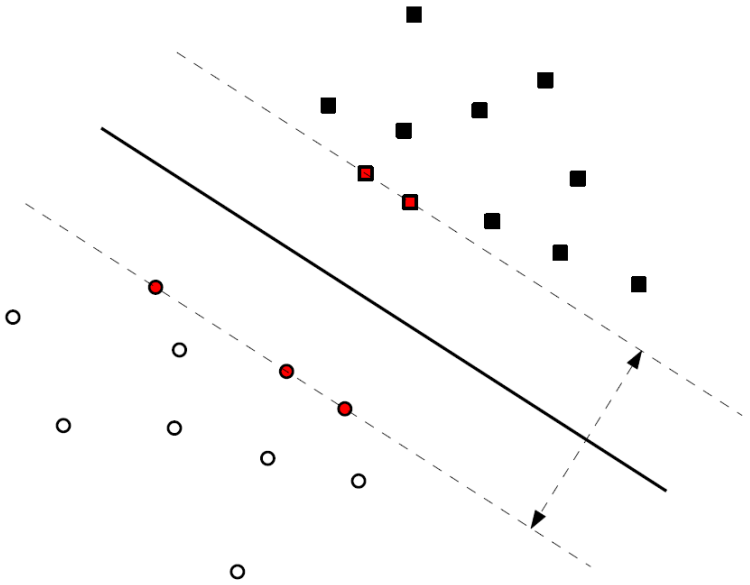

Figura 3.6: SVM linear com a margem maximizada

Em casos reais de classificação, os dados raramente são linearmente separáveis. Neste caso, uma função ϕ é utilizada para mapear os atributos em uma dimensão de ordem maior. Em um espaço dimensional maior, os dados poderão ser novamente separados de forma linear. Essa função ϕ responsável por essa conversão é chamada de kernel, e existem diversas funções diferentes que executam essa tarefa. A figura 3.7 ilustra o mapeamento que possibilita a separação linear dos dados em um espaço dimensional superior.

Mesmo com a utilização de kernels, dados reais, muitas vezes, possuem amostras fora da curva, ou seja, pontos que fogem do padrão esperado (*outliers*). Em geral, é possível fazer um ajuste perfeito, utilizando-se de kernels e outras parametrizações que façam o algoritmo se ajustar perfeitamente aos dados. Entretando, isso nem sempre é

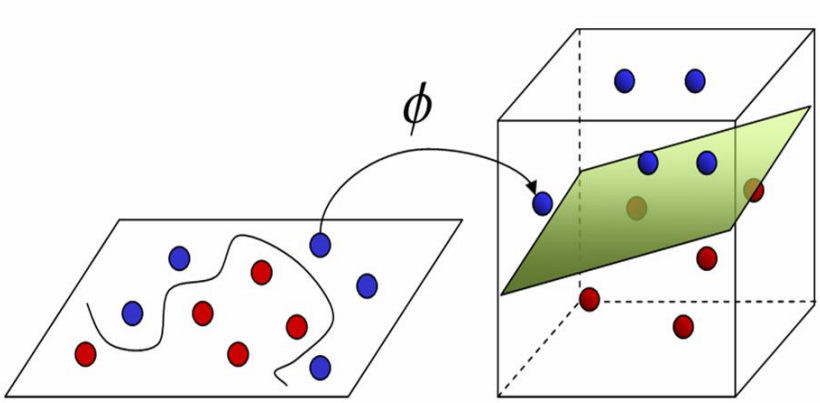

Figura 3.7: Função ϕ mapeando atributos em uma dimensão de ordem superior

desejado, pois pode levar à ocorrência do fenômeno *overfitting*. Esse fenômeno ocorre quando o modelo criado está tão bem ajustado aos dados, que não consegue generalizar bem para novas amostras. Esse fenômeno pode ser melhor entendido ao analisar a figura 3.8. Suponha quatro possíveis funções polinomiais de grau M igual a 0, 1 3 e 9 que tentam se adequar aos pontos mostrados no gráfico. Percebe-se que a função com $M = 3$ consegue se ajustar bem aos dados, apesar de errar em alguns pontos. Já a função com $M = 9$ passa perfeitamente por todos os pontos da base de treino, porém fica claro que a função (linha vermelha) não está seguindo o padrão dos dados, e dessa forma, não irá generalizar bem para novos pontos.

O *overfitting* ocorre pois o modelo se ajusta para tratar casos discrepantes em relação ao resto dos dados. O modelo ideal deve possuir um balanceamento entre estar bem ajustado com a base de teste e estar sendo bem generalizado para novos dados [15]. No SVM, esse balanceamento é feito através do parâmetro C, que controla o peso que um erro tem para o modelo. Um C alto, penaliza severamente cada erro, fazendo com que o modelo se ajuste mais aos dados para diminuir o erro. Uma descrição mais matemática do SVM pode ser encontrada em [11], onde Vapnik et al. descreveram pela primeira vez a sua formulação inicial.

3.5 Avaliação

Quando se trabalha com classificadores, é necessário se trabalhar também com medidas que possam avaliar se a classificação realizada foi eficaz, ou seja, é necessário saber qual foi a taxa de acerto e erro do classificador. Entre as medidas de desempenho existentes, serão citadas aqui algumas comumente utilizadas na literatura e também utilizadas nessa pesquisa. Para melhor entendimento das subseções seguintes, considere a tabela 3.1.

A tabela 3.1 mostra uma tabela de contingência que relaciona a classe real de um documento (se é positivo ou negativo) com o julgamento que o classificador realizou (se julgou ser positivo ou se julgou ser negativo).

Para as quatro situações possíveis existem quatro nomenclaturas:

- **Verdadeiro-positivo (VP)**: Número de documentos que foram classificados como positivos e realmente são da classe de positivos.

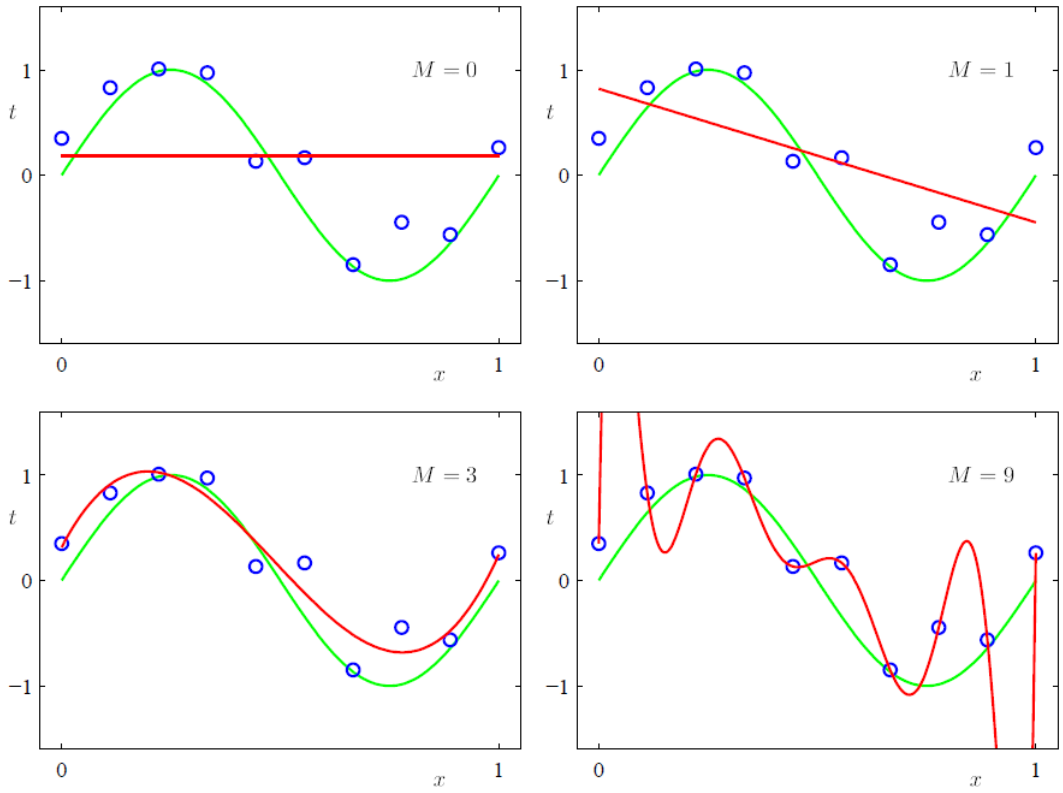

Figura 3.8: Função poliminial de grau $M = 0, 1, 3$ e 9 respectivamente

- **Falso-positivo (FP)**: Número de documentos que foram classificados como positivos e são da classe de negativos.
- **Falso-negativo (FN)**: Número de documentos que foram classificados como negativos e são da classe dos positivos.
- **Verdadeiro-negativo (VN)**: Número de documentos que foram classificados como negativos e realmente são da classe de negativos.

Com essas definições pode-se perceber que os Verdadeiro-positivos e os Verdadeiro-negativos são realmente as classificações feitas corretamente, enquanto que os Falso-negativos e os Falso-positivos são classificações feitas erroneamente. Feitas essas definições, serão explicadas a seguir algumas medidas de desempenho [29].

3.5.1 Acurácia

Acurácia é uma medida de desempenho dos acertos realizados pelo classificador, sendo definida pela seguinte fórmula:

$$\text{Acurácia} = \frac{VP + VN}{VP + FP + FN + VN}$$

A princípio essa medida parece ser bem interessante, porém ela falha em alguns casos. Suponha uma base com 1000 comentários. Destes 1000 comentários 995 são negativos e 5

Classificação real	**Positivo**	**Negativo**
Classificado como Positivo	Verdadeiro-positivo	Falso-positivo
Classificado como Negativo	Falso-negativo	Verdadeiro-negativo

Tabela 3.1: Matriz de confusão

são positivos. Suponha que a classe de interesse é a positiva e que um classificador julgou os 995 negativos corretamente, porém também classificou os outros 5 como negativos.

Essa medida gera uma taxa de acurácia de 99,5% para um classificador que não classificou corretamente nenhum documento da classe positiva, a classe de interesse. Isso ocorreu pois essa medida não levou em consideração os 5 comentários que deveriam ter sido classificados como positivos. E é justamente nesse ponto que as próximas medidas a serem mostradas se diferem da acurácia.

3.5.2 Precisão

A medida de precisão mensura a quantidade de documentos que foram corretamente classificados como positivos dentre todos os documentos que foram julgados como positivos. Para isso, a medida se baseia em uma divisão do número de acertos (VP) pelo número de documentos classificados como positivos ($VP + FP$), ou seja:

$$\text{Precisão} = \frac{VP}{VP + FP}$$

3.5.3 Recall

A medida de recall (ou sensibilidade), ao contrário da precisão, é a relação entre a quantidade de documentos que foram corretamente classificados como positivos dentre todos os documentos que são realmente da classe de positivos.

$$\text{Recall} = \frac{VP}{VP + FN}$$

Exemplificando essas medidas de precisão e recall, no exemplo anterior dos 1000 comentários, teríamos uma Precisão = 0% e um Recall = 0%, o que é totalmente cabível, uma vez que o classificador realmente não fez nenhuma classificação positiva correta e ainda classificou erroneamente os 5 que eram positivos.

Suponha um outro classificador que gerou as seguintes estatísticas:

$$VP = 4, FP = 30, FN = 1, VN = 965$$

Dessa forma teríamos uma precisão = 11,8% e um recall = 80%, ou seja, obtém-se uma taxa alta de recall, pois de 5 documentos que eram realmente positivos, o classificador

acertou 4, e obtém-se uma taxa baixa de precisão pois de 34 documentos que foram classificados como positivos, somente 4 eram realmente positivos.

3.5.4 F-Measure

Mesmo com as medidas de precisão e recall sendo de grande utilidade para avaliações de desempenho de classificadores, às vezes se faz necessária uma medida única, para que se possa comparar de forma direta dois ou mais classificadores. E essa é a ideia da medida F_β que é uma medida baseada na média harmônica dos valores de precisão e recall.

$$F_\beta = \frac{(1+\beta^2) * \text{Precisão} * \text{Recall}}{\beta^2 * \text{Precisão} + \text{Recall}}$$

Na fórmula apresentada percebe-se que dependendo do valor da constante β, a medida F_β estará dando maior importância para a medida de precisão ($0 < \beta < 1$) ou para a medida de recall ($\beta > 1$).

Um caso particular e muito utilizado dessa medida F é a medida F_1, com $\beta = 1$, isto é, a medida F que dá a mesma importância para Precisão e Recall, resultando na fórmula:

$$F_1 = \frac{2 * \text{Precisão} * \text{Recall}}{\text{Precisão} + \text{Recall}}$$

3.5.5 Um-contra-todos

As medidas de desempenho citadas nas seções anteriores estão baseadas na matriz de confusão mostrada na tabela 3.1, portanto, são utilizadas para mensurar um classificador que trabalha com duas classes: positivo e negativo. Entretanto, nessa pesquisa é utilizada, além das classes positivo e negativo, a classe neutro, e, portanto, se faz necessária uma pequena mudança na abordagem dessas medidas.

Nessa nova abordagem cada uma das classes existentes é comparada com todas as outras e por isso é considerada uma nova matriz de confusão para cada classe. Tome como exemplo o cálculo do recall de um classificador que trabalha com as classes positivo, negativo e neutro. Nesse cálculo é necessária uma matriz de confusão que trabalha com a classe "Positivo" e a classe "Não Positivo", uma matriz com "Negativo" e "Não Negativo" e outra matriz com "Neutro" e "Não Neutro". Para cada matriz é calculado um valor de Recall, da forma explicada na subseção 3.5.3, e por fim, calcula-se um valor final de recall para o classificador da seguinte forma:

$$\text{Recall}(final) = P(neg) * \text{Recall}(neg) + P(pos) * \text{Recall}(pos) + P(neu) * \text{Recall}(neu),$$

onde $P(neg)$ é a probabilidade de documentos classificados como negativos dentre a quantidade total de documentos e Recall(neg) é o valor do recall calculado da matriz de confusão referente às classes "Negativo" e "Não Negativo". O mesmo vale para positivo e neutro.

A tabela 3.2 mostra a matriz de confusão de três classes, enquanto as tabelas 3.3, 3.4 e 3.5 mostram, respectivamente, uma nova matriz de confusão para cada uma das classes.

	Positivo	**Negativo**	**Neutro**
Classificado como positivo	10	5	3
Classificado como negativo	8	17	20
Classificado como neutro	2	8	7

Tabela 3.2: Matriz de confusão com três classes

	Positivo	**Não positivo**
Classificado como positivo	10	8
Não classificado como positivo	10	52

Tabela 3.3: Matriz de confusão para a classe positivo

Com essas novas matrizes de confusão divididas em duas classes, pode-se calcular a Acurácia, Precisão, Recall e F-measure como explicado nas seções anteriores. Após esse cálculo, realiza-se a soma da medida de cada classe ponderada com a sua probabilidade para obter uma medida final, resultado nas seguintes fórmulas:

Acurácia(final)=P(neg)*Acurácia(neg)+P(pos)*Acurácia(pos)+P(neu)*Acurácia(neu)

Precisão(final)=P(neg)*Precisão(neg)+P(pos)*Precisão(pos)+P(neu)*Precisão(neu)

Recall(final)=P(neg)*Recall(neg)+P(pos)*Recall(pos)+P(neu)*Recall(neu)

F-measure(final)=P(neg)*F-measure(neg)+P(pos)*F-measure(pos)+P(neu)*F-measure(neu)

3.5.6 K-fold Cross Validation

Em muitos casos onde a quantidade de dados não é muito grande, utilizar 40% ou até um terço da base, para testar um modelo, pode acabar por prejudicar a análise, uma vez que a base para treino pode ficar pequena, e então não representar bem os dados reais. Nesses casos, uma técnica comumente utilizada é a chamada *k-fold cross validation*. Nesse método, o *corpus* é inicialmente dividido randomicamente em k partes iguais (ou de tamanho parecido) com aproximadamente a mesma distribuição de classes em cada parte. Feito isso, são executados k turnos de treinamento e validação, onde, em cada

	Negativo	**Não negativo**
Classificado como negativo	17	28
Não classificado como negativo	13	22

Tabela 3.4: Matriz de confusão para a classe negativo

	Neutro	**Não neutro**
Classificado como neutro	7	10
Não classificado como neutro	23	40

Tabela 3.5: Matriz de confusão para a classe neutro

turno, uma parte diferente é escolhida para validação enquanto as outras $k-1$ partes são utilizadas para treinamento/aprendizado. Assim, uma medida final para a performance do classificador pode ser obtida pela média calculada sobre os k testes. A figura 3.9[6] ilustra um exemplo de um *k-fold cross validation* com $k = 3$. Uma definição mais detalhada desse método, bem como suas vantagens e desvantagens podem ser encontradas em [34].

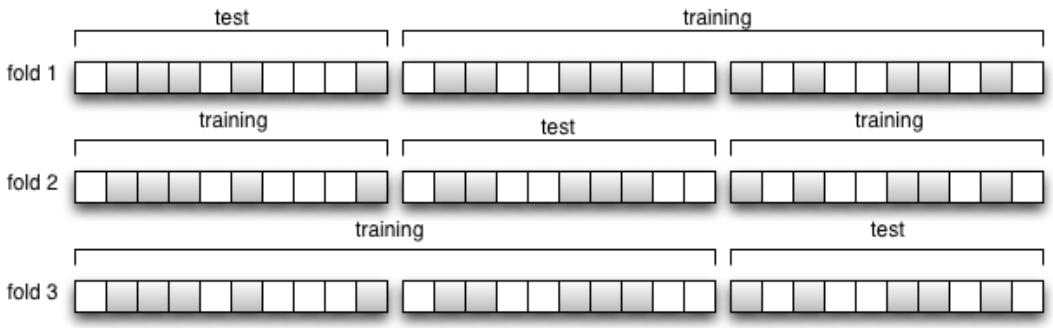

Figura 3.9: *3-fold cross validation*

3.6 CRISP-DM

Na resolução de tarefas em mineração de opinião foi também utilizada a metodologia CRISP-DM para mineração de dados, por mineração de opiniões e mineração de dados serem atividades correlatas. A metodologia *Cross-Industry Standard Process for Data Mining* - CRISP-DM segue um modelo hierárquico (figura 3.10[7]), que vai de um conjunto

[6] http://homepages.inf.ed.ac.uk/pmartin/tutorial/case$_s$*tudies.html*
[7] http://www.blue-granite.com/blog/?Tag=CRISP-DM

de tarefas mais gerais para um conjunto de tarefas mais específicas, seguindo a hierarquia de fases, tarefas genéricas, tarefas especializadas e, por fim, instâncias do processos.

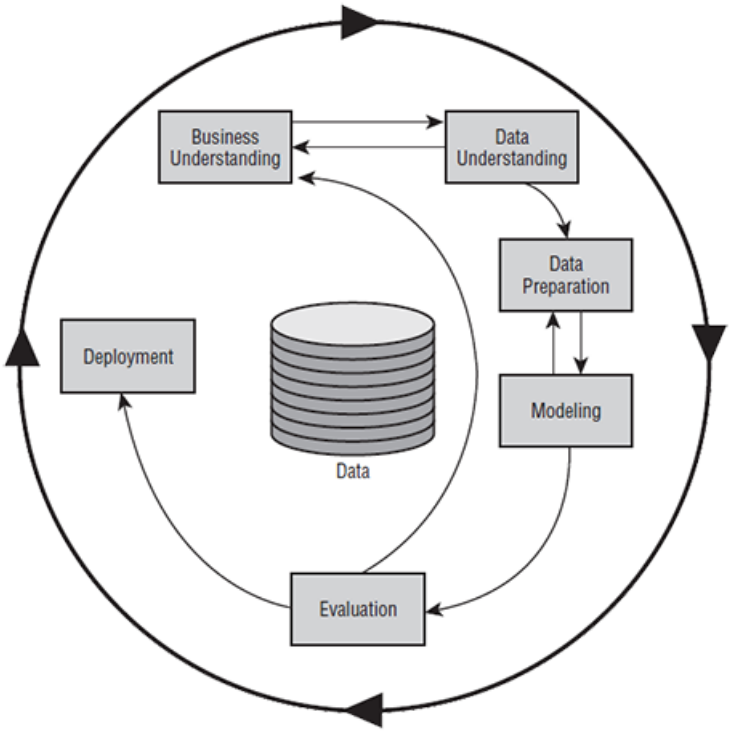

Figura 3.10: Fases do modelo CRISP-DM

As fases do CRISP-DM, em tradução livre, são: entendimento do negócio, entendimento dos dados, preparação dos dados, modelagem, avaliação, e utilização. Cada uma das fases, bem como suas tarefas, será explicada nas sessões adiante [16].

A fase de entendimento do negócio (*Business Understanding*) se refere ao entendimento dos objetivos do projeto e dos requisitos necessários para a sua realização na perspectiva do domínio do projeto. Nessa fase o projeto é mapeado como um problema de mineração de dados e é traçado um plano que guiará o projeto durante o seu desenvolvimento. São feitos cronogramas, listas de requisitos, seleção de ferramentas e técnicas de mineração de dados e etc.

A fase de entendimento dos dados (*Data Understanding*) se baseia na familiarização com os dados. Essa fase dá inicio a um processo de extração, verificação e análise dos dados coletados. Os dados devem ser verificados quanto à sua qualidade e se serão realmente úteis ao que é proposto pelo projeto. Nessa fase são feitos relatórios de descrição dos dados, da qualidade, da coleta inicial e etc.

A próxima fase é a de preparação dos dados (*Data Preparation*). Uma vez que os dados foram avaliados e considerados satisfatórios pela fase de entendimento, aqui os dados coletados passam por um processo de preparação que transforma os dados brutos iniciais em dados prontos para serem aplicados às ferramentas de mineração de dados. Nessa fase são realizadas tarefas como seleção de atributos, limpeza, construção e formatação dos dados.

Na fase de modelagem (Modelling) são selecionadas técnicas de modelagem para aplicação dos dados. Nessa fase também são escolhidos valores ótimos de parametrização para serem utilizados nessas técnicas de forma a obter os melhores resultados possíveis.

Na fase de avaliação (Evaluation) os modelos obtidos na fase de modelagem são avaliados para se ter a certeza de que esse modelo representa de forma suficiente os objetivos do negócio. Nessa fase os modelos são avaliados quanto a sua qualidade utilizando dados de teste diferentes dos dados utilizados para o treinamento e os resultados são verificados de acordo com os critérios adotados no entendimento do negócio.

A última fase é a de colocação em uso (*Deployment*). Nessa fase os resultados obtidos são utilizados por um analista para auxiliar em tomadas de decisões. Além disso, o modelo desenvolvido pode ser utilizado em outras bases de dados.

Capítulo 4

Experimentação

Neste capítulo serão descritos os dados analisados, os pré-processamentos utilizados, os experimentos e os resultados obtidos. O estudo de caso proposto nesta pesquisa envolve duas etapas. Em um primeiro momento, o estudo foi realizado utilizando-se uma pequena fração do *corpus* total obtido. A principal ferramenta utilizada, o Weka [8], não foi capaz de trabalhar com modelos de mineração de dados que envolvessem a quantidade total de documentos extraídos da *Google Play*. Logo, em um segundo momento, o uso de uma nova ferramenta, o Hadoop em conjunto com o Mahout, possibilitou a análise completa do *corpus* de forma à validar as hipóteses feitas na primeira análise e também de forma à iniciar estudos e experimentos em uma recente área de pesquisa: *Big Data*.

4.1 Obtenção do Corpus

A extração da base de dados utilizada nessa pesquisa foi realizada em conjunto com o aluno de graduação da Universidade de Brasília, Lucas Braga. Em sua pesquisa, um trabalho similar foi desenvolvido, porém foi utilizada, na mesma base, uma abordagem voltada para o processamento da linguagem natural, utilizando-se de *POS-Taggers* e outras técnicas.

Como mostrado na seção 3.1, os comentários sobre aplicativos para o sistema operacional *Android* da Google, estão dispostos em páginas da internet. Para processar devidamente os comentários, precisamos adquiri-los e armazená-los em um local propício para a modelagem desenvolvida nesse trabalho. Apesar do tamanho da loja e sua importância financeira para o Google, até dezembro de 2013 não foi disponibilizada uma API da própria Google para a extração destes dados. Por isso foi desenvolvido uma ferramenta (*crawler*) em JAVA capaz de extrair tais comentários e armazená-los em arquivos XML. *Web crawlers* são programas capazes de recuperar e fazer o *download* de páginas web de forma automática. Um *crawler* pode rapidamente visitar milhares de sites e buscar pelas informações desejadas, para que sejam posteiormente analisadas e/ou mineradas. Uma aplicação livre denominada Android-Market-Api [1] também foi utilizada. Esta API possibilita a extração de comentários de aplicativos através do ID do aplicativo. A partir disso, o *crawler* busca o máximo de IDs disponíveis relacionados às categorias de interesse para esse trabalho — jogos, e para cada aplicativo, busca o máximo de comentários disponíveis utilizando a Android-Market-Api.

O programa faz uma lista com os IDs de aplicativos encontrados e estes IDs são usados para buscar os comentários respectivos a cada aplicativo. Feito isso, a ferramenta retorna um arquivo XML organizado por aplicativos contendo o comentário, a data de publicação desse comentário e a nota de 1 a 5 dada pelo autor do comentário.

Os comentários disponíveis na página da loja estão em diversos idiomas. Porém, esse trabalho está focado somente em analisar comentários do idioma português. Para essa filtragem foi utilizada outra ferramenta livre *Language Detection* [3], que identifica o idioma predominante em um texto qualquer. Dessa forma, uma funcionalidade extra foi adicionada ao *crawler*, de forma a desprezar comentários não pertencentes à língua portuguesa. Como dito anteriormente, pela similaridade dos dois idiomas, alguns comentários em português de Portugal também foram entendidos pela ferramenta como português brasileiro.

A extração dos dados ocorreu em dois momentos. Durante o início da pesquisa, 700 aplicativos foram buscados (350 pagos e 350 gratuitos). A partir desses aplicativos, 190.095 comentários foram obtidos. Porém, como mostrado na seção 3.1, a quantidade de aplicativos e comentários na plataforma Google Play tem um crescimento muito rápido. Por isso, uma nova extração foi feita meses depois para que o máximo de dados fosse obtido para a segunda análise. Nessa segunda e última extração, comentários de 4708 aplicativos (1468 pagos e 3240 gratuitos) foram buscados, gerando um total de 759.176 comentários. Um teste X^2 (Chi-Quadrado) foi realizado na base com relação à classificação de estrelas para podermos considerar aplicativos pagos e gratuitos como uma única categoria de aplicativos. Outro teste similar foi aplicado, também com relação à classificação de estrelas, para validar que a quantidade de comentários não é um fator determinante para a distribuição de estrelas sobre os comentários de algum aplicativo. Portanto, foi possível tratar os comentários de todos os aplicativos de uma forma única sem discernimento quanto à qual aplicativo o comentário se refere.

4.2 Análise inicial

A abordagem utilizada para a mineração de texto utilizada nesse trabalho se assemelha e foi inspirada na abordagem proposta em [24], na qual divide o processo de mineração aplicada a textos do português brasileiro em três fases: o pré-processamento ou preparação dos dados, a análise dos dados/extração do conhecimento e a etapa de pós-processamento. Para o desenvolvimento dessa pesquisa, também foram aplicados conceitos do modelo CRISP-DM, explicados na seção 3.6. As etapas desenvolvidas nessa análise inicial do trabalho são explicadas nas subseções seguintes.

4.2.1 Seleção dos dados

Todo comentário disponível na plataforma Google Play possui juntamente com o texto do comentário a nota do aplicativo dada pelo autor que varia de 1 a 5 estrelas. Entretanto, foi observado que nem sempre essa nota está de acordo com o texto do comentário. A tabela 4.1 mostra alguns exemplos extraídos da base onde a nota dada pelo usuário não condiz com o comentário escrito.

Uma vez que a quantidade de comentários é muito grande, aproximadamente 759 mil, não é possível fazer uma seleção manual desses comentários para excluir ou de al-

Comentário	Nota
"*Bem, estou baixando não sei se é bom..*"	*
"*Ruim Nao consigo abrir o jogo.. aff =((*"	*****
"*Demora mt p fzr dowload!*"	**

Tabela 4.1: Exemplos de comentários não condizentes com a nota

guma forma corrigir esses comentários "não condizentes". Para tratar esse problema, uma amostra de 10 mil comentários foi selecionada de forma extratificada, isto é, seguindo as distribuições de cada classe. Esses comentários foram lidos um a um, e manualmente classificados entre as classes "positivo", "negativo" e "neutro". Dessa forma, como veremos nas seções seguintes, através de testes de hipótese, poderemos verificar que a base classificada manualmente possui comportamento igual à uma base classificada a partir das notas em estrelas. Logo, poderemos utilizar a base completa de comentários e suas respectivas notas em estrelas para treinamento e avaliação de algoritmos de aprendizagem de máquina. Esse *corpus* composto por 10 mil comentários classificados manualmente entre as 3 polaridades será tratado como o *corpus A* no decorrer do documento.

O interesse principal aqui é identificar comentários com polaridade positiva ou comentários com polaridade negativa. Assim sendo, a classe "neutra" foi utilizada como uma forma de classificar quaisquer outros comentários. Isso envolve principalmente dois casos. Comentários que não expressam opinião, isto é, comentários sem subjetividade, foram classificados como "neutro". De forma similar, comentários que não apresentam uma polaridade definida também foram classificados como "neutro". Esse caso inclui comentários que, por exemplo, elogiam alguns aspectos e criticam outros, e de forma geral, não deixam clara uma orientação positiva e nem negativa. A tarefa realizada nessa aplicação é uma classificação em nível de documento, ou seja, aspectos não estão sendo tratados separadamente. Logo, o objetivo é definir uma orientação para o comentário como um todo.

A quantia de 10 mil comentários para o *corpus A* foi escolhida por dois motivos:

1. Apesar de ser necessário um certo esforço, 10 mil comentários podem ser lidos e classificados manualmente em um tempo cabível.

2. A ferramenta Weka, utilizada nessa etapa da pesquisa, consegue trabalhar com modelos para 10 mil comentários de forma acessível, dada a quantidade de memória (8GB) da máquina onde foram realizados os experimentos.

Para verificar a hipótese de que a classificação das estrelas é válida, o corpus B foi considerado. O corpus B é composto pelos mesmos comentários presentes no corpus A, porém a sua classificação é feita pelas notas dos usuários e não a partir da classificação manual. As notas em estrelas foram convertidas nas classes "positivo", "negativo" e "neutro" da seguinte maneira: comentários de 1 e 2 estrelas são marcados como da classe "negativo", 3 estrelas marcados como da classe "neutro" e 4 e 5 estrelas como da classe "positivo". Após a leitura dos 10 mil comentários para a classificação manual do corpus A, ficou fácil perceber essa distribuição nos comentários e por isso a classificação foi feita dessa forma.

Por fim, um *corpus C* foi criado, contendo comentários distintos dos 10 mil comentários pertencentes aos *corpora A* e *B* e também utilizando a classificação por estrelas descrita acima. O *corpus C* foi criado no intuito de validar e de garantir que os 10 mil comentários inicialmente selecionados não estejam de certa forma enviesados ou não tornem possível a generalização dos modelos criados. A tabela 4.2 mostra as características dos *corpora* utilizados.

Nome	**Comentários**	**Classificação**
Corpus A	10 mil iniciais	Manual
Corpus B	10 mil iniciais	Por estrelas
Corpus C	10 mil distintos	Por estrelas

Tabela 4.2: *Corpora* utilizados na análise inicial

(%)	**1***	**2***	**3***	**4***	**5***
Comentários	10.29%	2.60%	5.99%	12.77%	68.20%

Tabela 4.3: Distribuição de estrelas entre o *corpus* obtido na primeira extração

(%)	**1***	**2***	**3***	**4***	**5***
Comentários	16.01%	3.87%	8.29%	13.42%	58.38%

Tabela 4.4: Distribuição de estrelas entre o *corpus* completo

Devido a própria natureza dos comentários, o *corpus* total extraído não é bem balanceado. As distribuições de estrelas encontradas nas duas extrações do *corpus* são apresentadas nas tabelas 4.3 e 4.4.

Com o intuito de minimizar os efeitos negativos causados pelo não balanceamento da base, os *corpora A* e *B*, formados a partir de amostras extratificadas dessa proporção, foram modificados de forma a aumentar a proporção das classes de estrelas 1, 2 e 3, enquanto no corpus *C*, essa exata proporção foi mantida para fins de teste e validação do modelo. As distribuições das classes nos corpora *A*, *B* e *C* são apresentadas na tabela 4.5.

Corpus	**Negativo**	**Neutro**	**Positivo**
Corpus A	16.31%	21.92%	61.76%
Corpus B	16.31%	21.92%	61.76%
Corpus C	12.87%	6.13%	80.92%

Tabela 4.5: Distribuição das classes nos *corpora A*, *B* e *C*

4.2.2 Pré-processamento

Três fases de pré-processamento, de forma sequencial, são aplicadas em cada um dos *corpora* descritos anteriormente. Dessa forma, são criados 4 conjuntos de *corpora* diferentes, onde cada um possui um nível a mais de pré-processamento. O primeiro é o corpus sem nenhum tipo de pré-processamento e os 3 seguintes são formados a partir das seguintes tarefas, respectivamente:

- **Padronização dos termos**: eliminação de acentuação, pontuação, caracteres especiais, números e todas as letras são transformadas em letras em caixa baixa.
- **Correção ortográfica**: eliminar possíveis erros ortográficos do texto a ser analisado, utilizando-se de dicionários do idioma tratado.
- **Stemming**: eliminar das palavras variações como, por exemplo, plural, formas de gerúndio, sufixos temporais, com o intuito de reduzir cada palavra para o seu radical.

Dessa forma, temos os *corpora* divididos nos seguintes estágios:

- Estágio 1: *corpus* sem nenhum pré-processamento
- Estágio 2: *corpus* com padronização dos termos
- Estágio 3: *corpus* com padronização dos termos e correção ortográfica
- Estágio 4: *corpus* com padronização dos termos, correção ortográfica e *stemming*

Na metodologia proposta em [24], uma outra etapa de pré-processamento é utilizada: a remoção de stopwords, isto é, eliminação de palavras que ocorrem frequentemente nos textos e que não contribuem para o entendimento ou o descobrimento de padrões nos mesmos. Todavia, nesse trabalho, essa etapa é sempre utilizada, em todas as fases do pré-processamento, visto que é uma técnica já muito reconhecida e já tem sua eficácia validada em muitas outras pesquisas.

Diferentes *corpora* são criados com o intuito de analisar e avaliar até que ponto os pré-processamentos aqui propostos são realmente úteis para a base de dados utilizada. Os resultados finais de cada um dos *corpora* são posteriormente comparados para isso.

Durante a fase de pré-processamento, os corpora são sempre armazenados em arquivos CSV (*Comma Separated Value*) e acessados utilizando a linguagem de programação JAVA. A fase de padronização dos termos é feita utilizando funcionalidades básicas do Java para tratamento de *Strings*.

Um dicionário com 2000 termos, gírias e abreviações em geral foi criado para a fase de correção ortográfica. GNU Aspell [2] é um projeto livre e de código aberto para verificação ortográfica. A partir do dicionário e das regras de associação para formação de palavras do português disponíveis nesse projeto, foi possível gerar um arquivo texto no tamanho de 137MB contendo inúmeras palavras do português. Essa lista de palavras foi comparada com cada palavra presente no *corpus* completo. Após ordenar as frequências obtidas nessas comparações, uma lista das palavras incorretas (de acordo com a língua portuguesa) mais recorrentes foi obtida. Nessa lista, encontram-se palavras e gírias do tipo "vc", "mto", "dahora" e até palavras escritas erroneamente como "exelente", "demaais" ou "interesante". Foram selecionados os 2000 primeiros termos dessa lista e para cada termo,

a sua respectiva correção foi adicionada manualmente, formando assim um dicionário de gírias, abreviações e erros mais comuns para o domínio da base em questão.

Outra abordagem proposta para a correção ortográfica foi a aplicação direta do corretor Aspell na base de comentários. No entanto, não foi verificada uma boa performance. Um dos motivos é que devido ao fato de trabalharmos no domínio de jogos e celulares, termos da língua inglesa como "galaxy S3", "android", "app", "tablet", "multiplayer", "server" são recorrentes, e para esses termos o corretor Aspell inevitavelmente os trata como palavras incorretas e os substitui por outras palavras.

Para o tratamento de Stemming, foi utilizado o PTStemmer. Um stemmer livre e de código aberto para português desenvolvido em JAVA por Pedro Oliveira [5].

4.2.3 Extração do conhecimento

A extração do conhecimento é feita em duas etapas. Primeiro o texto pré-processado é transformado em uma matriz de termo-documento (*Term-document Matrix*) e posteriormente algoritmos de classificação são aplicados nessa matriz.

Essa etapa da pesquisa foi desenvolvida com a utilização do software Weka [8]. O Weka é um software de código aberto, desenvolvido pela universidade de Waikato – Nova Zelândia, voltado para algoritmos de aprendizagem de máquina. É uma ferramenta muito poderosa que inclui algoritmos para pré-processamento de dados, classificação, regressão, clusterização e outras técnicas utilizadas na mineração de dados.

Geração da matriz termo-documento

Durante a geração da matriz termo-documento, alguns parâmetros se fazem importantes para a performance que um classificador terá à partir da matriz gerada. Parâmetros como utilização do TF-IDF, frequência mínima de um termo para que ele seja considerado um termo válido na matriz, normalização da matriz (deixar os valores da matriz em escalas proporcionais), geração de tokens (*n-grams*), número de atributos e outros foram testados para decidir qual seria a melhor parametrização para a formação da matriz. E feito isso, essa mesma parametrização é usada para todos os testes. As seguintes configurações foram utilizadas:

- *TF-IDF*
- *Stopwords*
- *Frequência mínima = 5*
- *Normalização da base*
- *bigrams e unigrans*

Devido ao desbalanceamento dos *corpora* utilizados, foram realizados testes utilizando-se técnicas de *undersampling* e *oversampling* [12] na base. Classes minoritárias foram duplicadas de forma a igualarem-se à classe majoritaria e, de forma similar, classes majoritárias tiveram comentários removidos de forma a igualarem-se à classe minoritária.

Aplicação dos classificadores

Uma vez que a matriz termo-documento é construída, o próximo passo é a aplicação de algoritmos de classificação sobre a mesma. De forma similar à geração da matriz, inicialmente diversos testes com classificadores diferentes e com parametrizações diferentes para cada classificador são realizados e por fim o modelo com melhor performance é então utilizado para todos os testes.

Nessa fase do processamento um filtro de seleção de atributos (*attributeSelection*) foi aplicado à matriz. A seleção de atributos é dividida em duas etapas. Primeiro cada atributo, ou seja, cada termo da matriz, recebe um valor que representa o quão importante é aquele atributo. Para essa etapa foi utilizado o *InfoGainAttributeEvaluator*. Na segunda etapa, um método de busca é utilizado para decidir se o atributo deve ou não ser utilizado baseado na avaliação da primeira etapa. Nesse caso, foi utilizado o método *Ranker*.

Classificadores como redes neurais, redes bayesianas, naive bayes, SVM e outros foram testados para alcançar a melhor performance possível. Testes com meta-classificadores também foram realizados. Meta-classificadores são algoritmos de classificação que utilizam do resultado de outros classificadores para chegar a um resultado final. O *Voting*, por exemplo, é um classificador que pega o resultado de diversos outros classificadores e faz a votação entre eles para determinar o resultado final de cada classificação. Já um *Cost Sensitive Classifier* utiliza um classificador juntamente com uma matriz de custo.[1] Por fim, a melhor performance para um tempo de processamento aceitável foi obtida com o SVM.

Os testes realizados durante essa fase de experimentação de diferentes classificadores e parametrizações foram sempre realizados com o método *k-fold cross validation* com um valor de $k = 5$.

4.2.4 Resultados

Após aplicadas as fases de pré-processamento e construídos os modelos de classificação, obteve-se a melhor performance utilizando-se do algoritmo SVM e sua parametrização padrão do Weka. Os resultados das classificações para os *corpora* A e B são mostrados, respectivamente, nas tabelas 4.6 e 4.7.

	Acurácia	**Precisão**	**Recall**	**F-Measure**	**Área ROC**
Estágio 1	81.6482%	0.818	0.816	0.816	0.869
Estágio 2	81.9538%	0.819	0.82	0.818	0.865
Estágio 3	**82.6644%**	**0.829**	**0.827**	**0.827**	**0.876**
Estágio 4	82.0538%	0.819	0.821	0.819	0.87

Tabela 4.6: Resultados SVM - *corpus A*

[1]Uma matriz de custo define diferentes importâncias para os erros de uma classificação. Imagine o problema da classificação de um tumor em benigno e maligno. Prever que um tumor é benigno, sendo que na verdade ele é maligno, é um erro muito mais grave do que a situação contrária.

	Acurácia	**Precisão**	**Recall**	**F-Measure**	**Área ROC**
Estágio 1	81.2081%	0.782	0.812	0.779	0.775
Estágio 2	**81.3232%**	**0.782**	**0.813**	**0.78**	**0.776**
Estágio 3	81.2031%	0.782	0.812	0.78	0.774
Estágio 4	81.163%	0.78	0.812	0.775	0.771

Tabela 4.7: Resultados SVM - *corpus B*

Ao contrário do esperado, ao analisar as tabelas 4.6 e 4.7, percebeu-se que a melhora na performance entre um estágio e outro não é muito grande. Na tabela 4.6, pode-se perceber uma melhoria gradual até o estágio 3, enquanto que para o *corpus B*, já é observado um leve agravamento de algumas medidas entre o estágio 2 e 3. Esse comportamento não era o esperado, uma vez que a cada novo estágio a quantidade de palavras é reduzida, diminuindo assim, o espaço dimensional tratado pelo algoritmo. Do estágio 1 para o 2, por exemplo, os termos "não" e "nao" se tornariam um só termo, assim como do estágio 2 para o 3, os termos "exelente" e "excelente" também são tratados unicamente. Foi discutido em [31], que "técnicas padrões de aprendizagem de máquina, como *support vector machines* (SVM), podem ser aplicadas diretamente sobre o documento em si", isto é, sem nenhum tipo de pré-processamento. Por tal motivo, Bo Pang e Liliam Lee [32] [31] aplicam tais técnicas sem sequer utilizar algum pré-processamento. Emma Haddi et al. [21], em seus estudos sobre "o papel do pré-processamento em análise de sentimentos", também obtiveram poucas melhorias nos resultados ao aplicar diferentes tipos de pré-processamento. Entretanto, pode-se pensar que mesmo sendo pouco, uma melhora de 1% na acurácia é algo considerável. Como citado na seçao de trabalhos correlatos, Lin et al. [25] apresentaram uma melhora de menos de 1% na acurácia do classificador ao utilizar 100 milhões de *tweets* ao invés de 1 milhão.

Após obtidos os resultados médios das medidas de desempenho de cada *corpora*, um teste de hipótese t-teste (*Student's t-test*) foi aplicado na medida F-Measure, para testarmos a hipótese de que as médias encontradas no *corpus A*, são similares as médias encontradas no *corpus B*. Dessa forma, podemos assumir que as diferenças encontradas quanto à classificação manual e a classificação por estrelas (diferença entre os corpora A e B) são devido ao acaso e não a erros sistemáticos com respeito à essa classificação. A idéia nesse teste, é verificar que a classificação feita pelos usuários com as estrelas, apesar de possuir algumas inconsistências, é válida e funciona de forma similar à classificação feita manualmente nesse trabalho, podendo assim, ser usada para trabalhos futuros como um corpus válido para aplicações de mineração de dados. O teste foi aplicado para os 4 estágios e a hipótese foi aceita para todos. O valor do P-valor (probabilidade de rejeitarmos a hipótese nula, sendo que ela é verdadeira) é dado na tabela 4.8. Para os testes foi usado um nível de confiança de 95% e 3 graus de liberdade.

O *corpus C* foi criado no intuito de validar o modelo criado a partir do corpus *B*. Apesar do *5-fold cross validation* ter sido usado em todos os testes, se faz necessário aplicar o modelo em uma base nunca antes vista para verificar se o modelo pode ser generalizado para um uso prático e real. Dessa forma, testes sobre a base *C* foram realizados e resultados interessantes foram obtidos. Verifica-se que o modelo generalizou

	Estágio 1	**Estágio 2**	**Estágio 3**	**Estágio 4**
P-valor	0.8102079	0.8108347	0.8175913	0.7871466

Tabela 4.8: P-valor (*Student's t-test*)

muito bem para dados novos, inclusive obtendo performance melhores do que a própria base de treino em algumas medidas. Como esperado, novamente percebeu-se o progresso gradual, e não muito eficaz, entre cada um dos níveis de pré-processamento.

	Acurácia	**Precisão**	**Recall**	**F-Measure**	**Área ROC**
Estágio 1	80.56%	0,861	0,806	0,819	0,763
Estágio 2	80.5945%	0,874	0,806	0,828	0,794
Estágio 3	**81.0849%**	**0,876**	**0,811**	**0,832**	**0,798**
Estágio 4	80.1141%	0,781	0,801	0,745	0,554

Tabela 4.9: Resultados SVM - Modelo aplicado à um *corpus C* de teste

Utilizando-se da ferramenta de seleção de atributos do Weka, foi possível ordenar os termos da matriz de termo-documento de acordo com a importância deste termo para determinar uma classe. Essa ordenação se mostra interessante para verificar e validar os termos representativos de cada classe. A tabela 4.10 mostra 10 dos termos mais importantes para a determinação das classes no *corpus A*.

Classe	**Termos**
Positivo	*muito bom, bom, adorei, excelente, viciante, amei, recomendo, otimo, legal, o melhor, perfeito, parabens, massa*
Negativo	*ruim, lixo, trava, erro, lento, porcaria, horrivel, fraco, bug, demora*

Tabela 4.10: Lista de termos representativos do *corpus A*

Visto que a extração da base de dados foi parte dessa pesquisa e por isso nenhum outro estudo ainda foi realizado a partir dela, não temos muitos meios para comparar os resultados diretamente. Entretando, podemos comparar os resultados obtidos com estudos parecidos. Em [18], por exemplo, acurácias finais de 81.25%, 82.6% e 80.76% foram obtidas em avaliações de comentários em português sobre restaurantes. Vale ressaltar que nesse trabalho, uma etapa de trabalho manual é requerida para definir a polaridade de palavras. Em [14], uma análise de sentimento sobre *tweets* em português também foi feita, e a melhor acurácia obtida foi de 75,1053%. Já em análises feitas sobre a língua inglesa, em [32], uma das pesquisas mais citadas na área de mineração de opinião, 82.9% de acurácia foi obtido em avaliações de filmes. Dessa forma, pode-se considerar que resultados satisfatórios foram encontrados nas classificações aqui apresentadas, visto que um nível de performance similar à outros estudos da área foi atingido. Vale ressaltar que

estudos como este ainda são raros na língua portuguesa, e por isso, espera-se que esse trabalho sirva de base para futuras pesquisas e aplicações que possam ser desenvolvidas nessa área.

4.3 Análise final

A segunda análise proposta nessa pesquisa se baseia em validar as conclusões obtidas na primeira análise. Como explicado no início deste capítulo, a ferramenta Weka não foi suficiente para analisar todos os comentários obtidos pelo *crawler*. Diante disso, foi proposto o uso de ferramentas propícias à análise de dados em larga escala, isto é, o Hadoop e o Mahout.

Na verdade, a quantidade de dados utilizada aqui é considerada pequena para o Mahout. Entretando, é considerada grande para o Weka. Utilizando-se o Mahout em conjunto com um cluster suficientemente grande no Hadoop, é possível gerar modelos de aprendizagem de máquina com dados contendo bilhões de amostras. Devido à alta performance desses frameworks, nossos testes foram realizados em tempo razoável com um *cluster* formado por somente 6 computadores.

As etapas realizadas nessa segunda análise estão descritas nas seções a seguir.

4.3.1 Montagem do cluster

Para a realização dos experimentos, um *cluster* composto por 6 máquinas foi montado no laboratório de raciocínio automatizado (LARA) da Universidade de Brasília. As seguintes tecnologias foram usadas:

- 6 máquinas HP (Processador Intel i5, 8GB de memória RAM)
- Sistema operacional Ubuntu 12.04 (x64)
- Hadoop 0.20.205.0
- Mahout 0.8

Um dos computadores (*master*), além de trabalhar como *slave*, recebendo tarefas para serem executadas, também é responsável por gerenciar o dispacho de processos e tarefas para os outros computadores. Os computadores *slave* executam dois processos: *TaskTracker* e *DataNode*. O *TaskTracker* é responsável pelo recebimento de tarefas do *master* enquanto o *DataNode* é responsável pela comunicação com o sistema de arquivos distribuído. O computador *master*, além dos dois processos de *slave*, também executa o *NameNode* e o *JobTracker*. O *NameNode* gerencia o sistema de arquivos de todo o *cluster*, armazenando a àrvore de arquivos e gerenciando onde os dados são armazenados e replicados pelo *cluster*. Já o *JobTracker* é responsável por separar o problema em tarefas de mapeamento e redução e dispachá-las para os *slaves*.

4.3.2 Extração do conhecimento

O *corpus* utilizado nessa análise é composto pelos 759.176 comentários extraídos. A classificação de cada comentário é definida entre "positivo", "negativo" e "neutro", de acordo

com a nota em estrelas dada pelo usuário, da mesma forma como ocorrido nos *corpora* B e C da primeira análise (ver seção 4.2.1 para mais detalhes).

O pré-processamento realizado no *corpus* nesta fase é o mesmo do realizado na primeira análise (ver seção 4.2.2 para mais detalhes):

- Estágio 1: *corpus* sem nenhum pré-processamento
- Estágio 2: *corpus* com padronização dos termos
- Estágio 3: *corpus* com padronização dos termos e correção ortográfica
- Estágio 4: *corpus* com padronização dos termos, correção ortográfica e *stemming*

Assim como o Weka, o Mahout também é implementado em Java. Entretanto, a forma de se trabalhar com os dados e os algoritmos é bem diferente. Na verdade, o Mahout é uma ferramenta muito mais complexa e a curva de aprendizagem para dominar essa ferramenta é muito lenta. O paradigma de *map-reduce* bem como o da programação paralela, tornam a ferramenta mais complexa e também mais robusta. O fato de ser uma ferramenta menos utilizada, em comparação com o Weka, também dificulta bastante o aprendizado, uma vez que é mais difícil encontrar informações disponíveis em fóruns de dúvida e outras fontes de informação.

Uma vez que o Mahout e o Hadoop seguem o paradigma de *map-reduce*, os dados são sempre representados em pares chave-valor (ver seção 2.2.1 para mais detalhes). Os dados ficam sempre armazenados, em arquivos binários (por questões de performance), no sistema de arquivos do Hadoop e não no sistema de arquivos normal do sistema operacional.

Inicialmente, os comentarios sao transformados em arquivos sequencias onde a chave é a classe do comentário e o valor é o comentário em si. O próximo passo é a transformação para um par chave-valor onde a chave é um identificador do comentário e o valor é uma sequência de pesos TF-IDF para cada termo presente no comentário. Durante a conversão, assim como no Weka, algumas configurações são selecionadas, como tamanho dos *n-grams*, frequência mínima e também a frequência máxima de um termo nos comentários (remoção de *stopwords*). Nessa fase, diversos arquivos são gerados:

- df-count: {**chave** = termo, **valor** = numero de documentos onde esse termo ocorre}
- dictionary.file: {**chave** = termo, **valor** = identificador do termo}
- frequency.file: {**chave** = identificador do termo, **valor** = frequência do termo}
- tf-vectors: {**chave** = identificador do comentário, **valor** = peso TF de cada termo presente no comentário}
- tfidf-vectors: {**chave** = identificador do comentário, **valor** = peso TF-IDF de cada termo presente no comentário}
- tokenized-documents: {**chave** = identificador do comentário, **valor** = lista de termos do comentário}
- wordcount: {**chave** = termo, **valor** = frequência do termo}

Uma vez que o vetor tf-idf foi gerado, podemos utilizá-lo para treinamento dos classificadores. A princípio, foi optado pela utilização do *support vector machine*. Entretando, devido a sua alta complexidade, até dezembro de 2013, nenhuma versão do Mahout possui uma implementação *map-reduce* do algoritmo. Algumas implementações[2][3] sequênciais estão disponíveis em forma de *patches*, isto é, pedaços de software desenvolvido por terceiros e disponibilizados como correções de defeitos (*issues*). Alguns desses *patches* foram testados, porém sem muito sucesso. A pouca documentação dos mesmos é um fator que dificultou o seu uso.

O próximo algoritmo testado foi o *naive bayes*. O Mahout possui uma implementação *map-reduce* para esse algoritmo. Testes com diferentes configurações para o Hadoop foram realizados. A quantidade de tarefas de mapeamento e redução, definidas pelo usuário, é uma configuração de grande importância para a performance do *cluster*, uma vez que escolhas ruins para a quantidade dessas tarefas podem gerar custos de comunicação desnecessários entre os nós do *cluster*. Os testes realizados com o *naive bayes* demonstraram resultados interessantes quando testados na própria base de treino, porém resultados muito insatisfatórios quando generalizados para uma base nova. Foi demonstrada em [19], uma potencial deficiência do *naive bayes* para problemas com classes desbalanceadas. Acredita-se então, que essa é a causa dos resultados insatisfatórios encontrados. Como explicado na seção 3.4.2, utiliza-se da probabilidade $P(c)$ de uma classe c para a classificação no *naive bayes*, isto é, a frequência da classe c entre todos os documentos. Uma vez que a frequência da classe positiva é bem maior que as outras, o classificador tende sempre a classificar a maioria das instâncias como positiva. Técnicas de *undersampling* e *oversampling* [12] foram utilizadas para amenizar o desbalanceamento entre as classes na base de treino, porém sem sucesso. Em casos onde a classe minoritária era "reamostrada" (*oversampled*) para se igualar às classes majoritárias, o classificador se tornava fortemente viciado nas classes minoritárias devido às ocorrências repetidas, logo, também resultava em classificações de baixa performance.

Uma dúvida (apresentada no anexo I deste documento) postada em 2011 na lista de emails do Mahout, iniciou uma discussão sobre a diferença de performance sobre uma mesma base de dados sendo testada sobre os algoritmos *Naive Bayes Multinomial* do Weka e o *Naive Bayes* do Mahout. No email, foi citado um estudo de caso, utilizando-se uma base desbalanceada com performance de 92% de acurácia no Weka e 66% no Mahout. Um teste similar foi feito nessa pesquisa, e o mesmo foi concluído. O *naive bayes* implementado no Mahout também apresentou uma performance inferior ao do Weka, sobre os mesmos dados.

O último algoritmo a ser testado foi a regressão logística. A implementação desse algoritmo no Mahout não é feita com *map-reduce*, e sim de forma sequencial. Entretanto, mesmo sendo sequencial, devido à forma como os dados são tratados e ao sistema de arquivos diferenciado, é possível gerar modelos com grandes quantidades de dados assim como no *naive bayes*.

Durante essa fase, os dados foram separados em três grupos: treino, validação e teste. As bases de treino foram usadas para o aprendizado do algoritmo. As bases de validação foram usadas para testar as diferentes configurações possíveis dos classificadores. Uma vez que a melhor configuração foi definida, a base de teste é então utilizada para

[2]https://issues.apache.org/jira/browse/MAHOUT-334
[3]https://issues.apache.org/jira/browse/MAHOUT-232

avaliar os resultados finais. As bases de treino, validação e teste foram separadas com, respectivamente, 60%, 20% e 20% do *corpus* total.

4.3.3 Resultados

Após realizados diversos testes no Mahout, o algoritmo de regressão logística proporcionou os melhores resultados. A tabela 4.11 mostra os resultados obtidos para cada uma das fases de pré-processamento.

	Acurácia	**Precisão**	**Recall**	**F-Measure**
Estágio 1	80.0843%	0,768	0,801	0,776
Estágio 2	82.0331%	0,787	0,820	0,794
Estágio 3	**81.2961%**	**0,786**	**0,813**	**0,796**
Estágio 4	81.4463%	0,776	0,814	0,786

Tabela 4.11: Resultados regressão logística

Como esperado, podemos observar o mesmo padrão obtido na análise inicial. Mesmo utilizando a base completa, o pré-processamento continua sendo pouco eficaz para o classificador. Observa-se uma melhora gradual nas medida F-measure até o estágio 3. Observa-se também que a medida de acurácia diminui em estágios onde a medida F-measure aumenta. Entretanto, visto que a base está bem desbalanceada, deve-se levar em consideração a medida F-measure.

Como dito na primeira análise, não existem outros estudos feitos com a mesma base para que comparações sejam feitas. No entanto, por meras questões comparativas, outros trabalhos que também utilizaram ferramentas de processamento em larga escala [23][25], obtiveram acurácias de aproximadamente 73% e 77%, respectivamente.

É importarte ressaltar que estudos de mineração de opinião em grandes quantidades de dados ainda não são tão frequêntes. Embora a *web* esteja repleta de dados propícios à essa análise, é necessário que os dados estejam previamente rotulados com suas classificações para que o modelo seja treinado. Especialmente em casos onde algoritmos de aprendizagem de máquina são utilizados. Em estudos com amostras menores, o custo manual necessário para a rotulação de 1000 ou 2000 amostras, por exemplo, é aceitável. Já quando se trabalha com centenas de milhares de amostras, a base utilizada precisa possuir alguma característica que possibilite uma rotulação automatizada, como foi o caso da nota em estrelas utilizada nessa pesquisa.

Mesmo não sendo um objetivo desse trabalho, é importante ressaltar algumas comparações entre as ferramentas Weka e Mahout, visto que as duas foram utilizadas de forma paralela para realizar uma mesma tarefa. Quanto à complexidade, pode-se dizer que o Weka é uma ferramenta mais simples, de fácil aprendizado e uso, visto que o Weka possui também uma interface gráfica. O Mahout, por sua vez, possui uma curva de aprendizagem maior, é mais complexo e mais robusto. Quanto à diversidade de algoritmos, verifica-se que o Weka é mais completo, sendo composto por dezenas de algoritmos de classificação e clusterização, enquanto que o Mahout não possui mais do que 5 algoritmos de classificação

implementados oficialmente. No entanto, pode-se dizer que essas duas ferramentas possuem "públicos-alvo" diferentes. O Mahout possui a capacidade de trabalhar com grandes quantidades de dados e por isso requer mais complexidade, uma vez que seus algoritmos estão focados em performance. Para simples mérito de exemplificação, a tarefa de criação da matriz de termo documento da base completa (759.176 comentários), que no Mahout é processada em alguns minutos em um cluster de 6 computadores, não foi concluída no Weka mesmo após o processamento contínuo durante 24 horas. Portanto conclui-se que em problemas onde a quantidade de dados é significante, o Weka se torna inutilizável e o Mahout surge como uma possível solução.

Capítulo 5

Conclusão

Nesse trabalho, técnicas de mineração de texto foram aplicadas em uma base de comentários sobre aplicativos para aparelhos móveis, com o intuito de classificá-los quanto à sua orientação positiva, negativa ou neutra. Uma série de técnicas de pré-processamento foram aplicadas gradualmente nesses comentários a fim de avaliar o impacto de tais técnicas no domínio dos comentários, que por sua vez, são compostos de gírias, abreviações e jargões da internet. Ao contrário do esperado, os métodos de pré-processamento propostos não resultaram em uma significativa melhora na performance.

Foi possível validar o mesmo resultado ao analisar uma quantidade maior de comentários com ferramentas de processamento distribuído. É importante ressaltar que quanto mais o preço para armazenamento e processamento de dados diminui, a quantidade de dados disponível para análise tende a crescer. Dessa forma, se faz necessário cada vez mais progressos em estudos que considerem grandes quantidades de dados, como é o caso desta pesquisa.

Cada comentário extraído para essa pesquisa possui uma nota em estrelas de 1 à 5. Através de testes de hipótese, concluímos que essa nota pode ser usada como rótulo de comentários "positivos", "negativos" e "neutros". Assim, fica como contribuição uma base de dados classificada que poderá ser utilizada em outras pesquisas da área.

Concluímos também que técnicas de mineração de texto e análise de dados se mostraram eficientes para classificar textos com relação à sua orientação, uma vez que foram atingidas medidas de desempenho similares à outros trabalhos da área. Vale ressaltar que a análise de sentimento em textos em português é ainda pouco estudada e por isso considera-se que este foi um estudo relevante para a literatura.

5.1 Principais contribuições

As principais contribuões deste trabalho são:

- Validação da baixa eficácia do pré-processamento de texto na análise de opinião para o domínio tratado.
- Obtenção de um *corpus* composto por 759.176 comentários opinativos da língua portuguesa, propício à novos estudos.
- Obtenção de um dicionário de gírias, abreviações e termos comumente utilizados por usuários da internet.

5.2 Trabalhos futuros

Permanecem abertas algumas questões que dão oportunidades à desenvolvimentos futuros:

- Investigar o motivo pelo qual o pré-processamento não demonstrou significativa melhora na performance dos classificadores.
- Refazer o estudo proposto em uma base com comentários de tamanho maior. Uma possível causa para a ineficácia do pré-processamento pode se relacionar com o fato dos comentários serem relativamente pequenos.
- Refazer o estudo proposto com outras implementações do Mahout. O algoritmo *naive bayes*, por exemplo, é considerado um ótimo algoritmo para mineração de texto, e mesmo assim gerou performances insatisfatórias devido ao desbalanceamento da base. Isso abre caminho para que outras técnicas e algoritmos sejam testados e tenham suas performances comparadas com os resultados apresentados aqui.

Referências

[1] Android-market-api. `https://code.google.com/p/android-market-api/`. Accessed: 2013-12-01. 38

[2] Gnu aspell. `http://aspell.net/`. Accessed: 2013-12-01. 42

[3] Language detection. `https://code.google.com/p/language-detection/`. Accessed: 2013-12-01. 39

[4] Mapreduce example - finding friends. `http://stevekrenzel.com/finding-friends-with-mapreduce`. Accessed: 2013-12-01. 13

[5] Ptstemmer. `https://code.google.com/p/ptstemmer/`. Accessed: 2013-12-01. 43

[6] Sentiment analysis with python nltk text classification. `http://text-processing.com/demo/sentiment`. Accessed: 2013-12-01. 16

[7] Sentiwordnet. `http://sentiwordnet.isti.cnr.it/`. Accessed: 2013-12-01. 7, 17

[8] Weka - data mining software in java. `http://www.cs.waikato.ac.nz/ml/weka/`. Accessed: 2013-12-01. 38, 43

[9] Michael J. Berry and Gordon Linoff. *Data Mining Techniques: For Marketing, Sales, and Customer Support*. John Wiley & Sons, Inc., New York, NY, USA, 1997. 20

[10] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, 1st ed. 2006. corr. 2nd printing 2011 edition, October 2007. 27

[11] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT '92, pages 144–152, New York, NY, USA, 1992. ACM. 29, 30

[12] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. Smote: Synthetic minority over-sampling technique. *J. Artif. Int. Res.*, 2002. 43, 49

[13] Nelson G. Rocha da Silva. Bestchoice: Classificação de sentimento em ferramentas de expressão de opinião. Tese de graduação, Universidade Federal de Pernambuco, Recife, 2010. 7, 17

[14] Emanuel de Barros Albuquerque Ferreira. Análise de sentimento em redes sociais utilizando influência das palavras. Tese de graduação, Universidade Federal de Pernambuco, Recife, 2010. 46

[15] R. de Groot. Data mining for tweet sentiment classification. Master thesis, Utrecht University, Utrecht, 2012. 18, 29, 30

[16] R Dias, M. e Pacheco. Uma metodologia para o desenvolvimento de sistemas de descoberta de conhecimento. *Acta Science Technology*, 2005. 36

[17] Thomas Jefferson P. Lopes et. al. Mineração de opiniões aplicada à análise de investimentos. *Brazilian Computer Society*, 2008. 15, 16

[18] Fernandes F. Um framework para análise de sentimento em comentários sobre produtos em redes sociais. Tese de pós-graduação, Universidade Federal de Pernambuco, Recife, 2010. 6, 17, 46

[19] Eibe Frank and Remco R. Bouckaert. Naive bayes for text classification with unbalanced classes. In *In Proc 10th European Conference on Principles and Practice of Knowledge Discovery in Databases*, pages 503–510, 2006. 49

[20] Alec Go, Richa Bhayani, and Lei Huang. Twitter sentiment classification using distant supervision. *Processing*, pages 1–6, 2009. 24

[21] Emma Haddi, Xiaohui Liu, and Yong Shi. The role of text pre-processing in sentiment analysis. *Procedia Computer Science*, 17:26–32, 2013. 6, 7, 24, 45

[22] Andreas M. Kaplan and Michael Haenlein. Users of the world, unite! The challenges and opportunities of Social Media. *Business Horizons*, 53, 2010. 4

[23] Vinh Ngoc Khuc, Chaitanya Shivade, Rajiv Ramnath, and Jay Ramanathan. Towards building large-scale distributed systems for twitter sentiment analysis. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, SAC '12, pages 459–464, New York, NY, USA, 2012. ACM. 18, 50

[24] Igor Ruiz Gomes e Thiago Oliveira Lêda de Oliveira Monteiro. Etapas do processo de mineração de textos – uma abordagem aplicada a textos em português do brasil. *Anais do XXVI Congresso da SBC*, 2006. 39, 42

[25] Jimmy Lin and Alek Kolcz. Large-scale machine learning at twitter. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, SIGMOD '12, pages 793–804, New York, NY, USA, 2012. ACM. 19, 45, 50

[26] Bing Liu. Opinion mining. In *Encyclopedia of Database Systems*, pages 1986–1990. Springer US, 2009. 3, 6

[27] Bing Liu and Lei Zhang. A survey of opinion mining and sentiment analysis. In *Mining Text Data*, pages 415–463. Springer US, 2012. 5

[28] Thomas Jefferson P. Lopes. Opsys. `https://www.opsys.com.br/`. Accessed: 2013-12-01. 15

[29] Raghavan P. Manning, C and Schütze H. *Introduction to Information Retrieval*. Cambridge University Press, Cambridge, 2009. 31

[30] R. Ormonde. Classificação automática de páginas web multi-label via mdl e support vector machines. Master thesis, Universidade de Brasília, Brasília, 2009. 29

[31] Bo Pang and Lillian Lee. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42Nd Annual Meeting on Association for Computational Linguistics*, ACL '04. Association for Computational Linguistics, 2004. 45

[32] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up?: Sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing - Volume 10*, EMNLP '02, pages 79–86, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics. 16, 17, 24, 45, 46

[33] Anand Rajaraman and Jeffrey David Ullman. *Mining of massive datasets*. Cambridge University Press, Cambridge, 2012. 9, 11, 14

[34] Payam Refaeilzadeh, Lei Tang, and Huan Liu. Cross-validation. *Encyclopedia of Database Systems*, pages 532–538, 2009. 35

[35] Klaus R. Scherer and Marcel R. Zentner. *Emotional effects of music: productions rules*. Oxford University Press, Oxford; New York, 2001. 5

[36] Giselle Stazauskas. Comunicação interna versus mídias sociais. Tese de pós-graduação, Pontifícia Universidade Católica de São Paulo, São Paulo, 2011. 4

[37] Peter D Turney. Thumbs up or thumbs down? semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 417–424. Association for Computational Linguistics, 2002. 7, 17

[38] Fayyad Usama, G Piatetsky-Shapiro, and Padhraic Smyth. From data mining to knowledge discovery in databases. *American Association for Artificial Intelligence*, pages 37–54, 1996. 20

[39] João Ventura and Joaquim Silva. Mining concepts from texts. *Procedia Computer Science*, 9(0):27 – 36, 2012. <ce:title>Proceedings of the International Conference on Computational Science, {ICCS} 2012</ce:title>. 16

[40] Shruti Wakade, Chandra Shekar, Kathy J Liszka, and Chien-Chung Chan. Text mining for sentiment analysis of twitter data. 2012. 18

ANEXO I

mahout-user mailing list archives

Site index · *List index*

Message view « Date » · « Thread »

From Benjamin Rey <benja...@c-optimal.com>
Subject Re: 92% accuracy on Weka NaiveBayesMultinomial vs 66% with Mahout bayes
Date Sat, 17 Sep 2011 08:30:44 GMT

```
Thanks a lot for your interest and time.
I'm computer-less for the coming week, but I'll run a few more
experiments and post the data as soon as I'm back home.

Thanks.

Benjamin


Le 17 sept. 2011 à 00:24, Ted Dunning <ted.dunning@gmail.com> a écrit :

> Benjamin,
>
> Can you post your actual training data on dropbox or some other place so
> that we can replicate the problem?
>
> On Fri, Sep 16, 2011 at 3:38 PM, Benjamin Rey <benjamin.rey@c-optimal.com>wrote:
>
>> Unfortunately CNB gives me the same 66% accuracy.
>>
>> I past the commands for mahout and weka below.
>>
>> I also tried to remove the biggest class, it helps but then it's the 2nd
>> biggest class that is overwhelmingly predicted. Mahout bayes seems to favor
>> a lot the biggest class (more than prior), contrarily to Weka's
>> implementation. Is there any choice in the parameters, or in ways of
>> computing weights that could be  causing this?
>>
>> thanks.
>>
>> benjamin
>>
>> here are the commands:
>> On Mahout:
>> # training set, usual prepare20newsgroup, followed by a subsampling for
>> just
>> a few classes
>> bin/mahout org.apache.mahout.classifier.bayes.PrepareTwentyNewsgroups   -p
>> examples/bin/work/20news-bydate/20news-bydate-train   -o
>> examples/bin/work/20news-bydate/bayes-train-input   -a
>> org.apache.mahout.vectorizer.DefaultAnalyzer   -c UTF-8
>> mkdir examples/bin/work/20news_ss/bayes-train-input/
>> head -400 examples/bin/work/20news-bydate/bayes-train-input/alt.atheism.txt
>>> examples/bin/work/20news_ss/bayes-train-input/alt.atheism.txt
>> head -200
>> examples/bin/work/20news-bydate/bayes-train-input/comp.graphics.txt >
>> examples/bin/work/20news_ss/bayes-train-input/comp.graphics.txt
>> head -100 examples/bin/work/20news-bydate/bayes-train-input/rec.autos.txt >
>> examples/bin/work/20news_ss/bayes-train-input/rec.autos.txt
>> head -100
>> examples/bin/work/20news-bydate/bayes-train-input/rec.sport.hockey.txt >
>> examples/bin/work/20news_ss/bayes-train-input/rec.sport.hockey.txt
>> head -30 examples/bin/work/20news-bydate/bayes-train-input/sci.med.txt >
```

```
>> examples/bin/work/20news_ss/bayes-train-input/sci.med.txt
>> hdput examples/bin/work/20news_ss/bayes-train-input
>> examples/bin/work/20news_ss/bayes-train-input
>>
>>  then same exact thing for testing
>>
>> # actual training:
>> bin/mahout trainclassifier   -i
>> examples/bin/work/20news_ss/bayes-train-input   -o
>> examples/bin/work/20news-bydate/cbayes-model_ss   -type cbayes   -ng 1
>> -source hdfs
>>
>> # testing
>> bin/mahout testclassifier   -d
>> examples/bin/work/20news_ss/bayes-test-input   -m
>> examples/bin/work/20news-bydate/cbayes-model_ss   -type cbayes   -ng 1
>> -source hdfs
>>
>> => 66% accuracy
>>
>> and for weka
>> # create the .arff file from 20news_ss train and test:
>> start the file with appropriate header:
>> -----
>> @relation _home_benjamin_Data_BY_weka
>>
>> @attribute text string
>> @attribute class
>> {alt.atheism,comp.graphics,rec.autos,rec.sport.hockey,sci.med}
>>
>> @data
>> -----
>> # then past the data:
>> cat ~/workspace/mahout-0.5/examples/bin/work/20news_ss/bayes-test-input/* |
>> perl mh2arff.pl >> 20news_ss_test.arff
>> # with m2harff.pl :
>> ----
>> use strict;
>> while(<STDIN>) {
>>   chomp;
>>   $_ =~ s/\'/\\\'/g;
>>   $_ =~ s/ $//;
>>   my ($c,$t) = split("\t",$_);
>>   print "'$t',$c\n";
>> }
>> ---
>> # and the train/test command:
>> java weka.classifiers.meta.FilteredClassifier -t 20news_ss_train.arff -T
>> 20news_ss_test.arff -F
>> "weka.filters.unsupervised.attribute.StringToWordVector -S" -W
>> weka.classifiers.bayes.NaiveBayesMultinomial
>>
>> => 92% accuracy
>>
>>
>>
>>
>>
>>
>> 2011/9/16 Robin Anil <robin.anil@gmail.com>
>>
>>> Did you try complementary naive bayes(CNB). I am guessing the multinomial
>>> naivebayes mentioned here is a CNB like implementation and not NB.
>>>
>>>
```

```
>>> On Fri, Sep 16, 2011 at 5:30 PM, Benjamin Rey <
>> benjamin.rey@c-optimal.com
>>>> wrote:
>>>
>>>> Hello,
>>>>
>>>> I'm giving a try to different classifiers for a classical problem of
>> text
>>>> classification very close to the 20newsgroup one.
>>>> I end up with much better results with Weka NaiveBayesMultinomial than
>>> with
>>>> Mahout bayes.
>>>> The main problem comes from the fact that my data is unbalanced. I know
>>>> bayes has difficulties with that yet I'm surprised by the difference
>>>> between
>>>> weka and mahout.
>>>>
>>>> I went back to the 20newsgroup example, picked 5 classes only and
>>>> subsampled
>>>> those to get 5 classes with 400 200 100 100 and 30 examples and pretty
>>> much
>>>> the same for test set.
>>>> On mahout with bayes 1-gram, I'm getting 66% correctly classified (see
>>>> below
>>>> for confusion matrix)
>>>> On weka, on the same exact data, without any tuning, I'm getting 92%
>>>> correctly classified.
>>>>
>>>> Would anyone know where the difference comes from and if there are ways
>> I
>>>> could tune Mahout to get better results? my data is  small enough for
>> now
>>>> for weka but this won't last.
>>>>
>>>> Many thanks
>>>>
>>>> Benjamin.
>>>>
>>>>
>>>>
>>>> MAHOUT:
>>>> -------------------------------------------------------
>>>> Correctly Classified Instances          :        491       65.5541%
>>>> Incorrectly Classified Instances        :        258       34.4459%
>>>> Total Classified Instances              :        749
>>>>
>>>> =======================================================
>>>> Confusion Matrix
>>>> -------------------------------------------------------
>>>> a        b        c        d        e        f        <--Classified as
>>>> 14       82       0        4        0        0         |  100       a
>>> =
>>>> rec.sport.hockey
>>>> 0        319      0        0        0        0         |  319       b
>>> =
>>>> alt.atheism
>>>> 0        88       3        9        0        0         |  100       c
>>> =
>>>> rec.autos
>>>> 0        45       0        155      0        0         |  200       d
>>> =
>>>> comp.graphics
>>>> 0        25       0        5        0        0         |  30        e
>>> =
>>>> sci.med
```

```
>>>> 0        0        0        0        0        0          |  0          f
>>> =
>>>> unknown
>>>> Default Category: unknown: 5
>>>>
>>>>
>>>> WEKA:
>>>> java weka.classifiers.meta.FilteredClassifier -t 20news_ss_train.arff
>> -T
>>>> 20news_ss_test.arff -F
>>>> "weka.filters.unsupervised.attribute.StringToWordVector -S" -W
>>>> weka.classifiers.bayes.NaiveBayesMultinomial
>>>>
>>>> === Error on test data ===
>>>>
>>>> Correctly Classified Instances         688               91.8558 %
>>>> Incorrectly Classified Instances        61                8.1442 %
>>>> Kappa statistic                          0.8836
>>>> Mean absolute error                      0.0334
>>>> Root mean squared error                  0.1706
>>>> Relative absolute error                 11.9863 %
>>>> Root relative squared error             45.151  %
>>>> Total Number of Instances              749
>>>>
>>>>
>>>> === Confusion Matrix ===
>>>>
>>>>  a   b   c   d   e   <-- classified as
>>>> 308   9   2   0   0 |   a = alt.atheism
>>>>  5 195   0   0   0 |   b = comp.graphics
>>>>  3  11  84   2   0 |   c = rec.autos
>>>>  3   3   0  94   0 |   d = rec.sport.hockey
>>>>  6  11   6   0   7 |   e = sci.med
>>>>
>>>
>>
```

Mime • Unnamed text/plain (inline, Quoted Printable, 7454 bytes)

View raw message

Top « Date » · « Thread »