Emerson Freitas Cardoso

Filtragem Automática de Opiniões Falsas – Comparação Compreensiva dos Métodos baseados em Conteúdo

Sorocaba, SP

4 de Agosto de 2017

Emerson Freitas Cardoso

Filtragem Automática de Opiniões Falsas – Comparação Compreensiva dos Métodos baseados em Conteúdo

Dissertação de mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação (PPGCC-So) da Universidade Federal de São Carlos como parte dos requisitos exigidos para a obtenção do título de Mestre em Ciência da Computação. Linha de pesquisa: Inteligência Artificial e Banco de Dados.

Universidade Federal de São Carlos – UFSCar

Centro de Ciências em Gestão e Tecnologia – CCGT

Programa de Pós-Graduação em Ciência da Computação – PPGCC-So

Orientador: Prof. Dr. Tiago Agostinho de Almeida

Sorocaba, SP

4 de Agosto de 2017

Freitas Cardoso, Emerson

Filtragem Automática de Opiniões Falsas – Comparação Compreensiva dos Métodos baseados em Conteúdo / Emerson Freitas Cardoso. -- 2017.
88 f. : 30 cm.

Dissertação (mestrado)-Universidade Federal de São Carlos, campus Sorocaba, Sorocaba
Orientador: Prof. Dr. Tiago Agostinho de Almeida
Banca examinadora: Prof. Dr. André Carlos Ponce de Leon Ferreira de Carvalho, Profa. Dra. Sahudy Montenegro González
Bibliografia

1. Opiniões falsas. 2. Classificação. 3. Aprendizado de máquina. I. Orientador. II. Universidade Federal de São Carlos. III. Título.

Ficha catalográfica elaborada pelo Programa de Geração Automática da Secretaria Geral de Informática (SIn).
DADOS FORNECIDOS PELO(A) AUTOR(A)

UNIVERSIDADE FEDERAL DE SÃO CARLOS

Centro de Ciências em Gestão e Tecnologia

Programa de Pós-Graduação em Ciência da Computação

Folha de Aprovação

Assinaturas dos membros da comissão examinadora que avaliou e aprovou a defesa de dissertação de mestrado do candidato Emerson Freitas Cardoso, realizada em 04/08/2017:

Prof. Dr. Tiago Agostinho de Almeida
UFSCar

Prof. Dr. André Carlos Ponce de Leon Ferreira de Carvalho
USP

Profa. Dra. Sahudy Montenegro González
UFSCar

Certifico que a sessão de defesa foi realizada com a participação à distância do membro Prof. Dr. André Carlos Ponce de Leon Ferreira de Carvalho e, depois das arguições e deliberações realizadas, o participante à distância está de acordo com o conteúdo do parecer da comissão examinadora redigido no relatório de defesa do aluno Emerson Freitas Cardoso.

Prof. Dr. Tiago Agostinho de Almeida
Presidente da Comissão Examinadora
UFSCar

Dedico esta dissertação à minha esposa Rita e à minha mãe Mariza;
por todo o amor e motivação que eu precisei para concluir este trabalho.

Agradecimentos

Agradeço,

a Deus, primeiramente, por me guiar em todos os passos da minha vida, concedendo força, coragem e sabedoria para tomar as decisões corretas todos os dias;

a todo o corpo docente da UFSCar Sorocaba por ministrar as disciplinas que agregaram enorme valor para a minha vida profissional, acadêmica e pessoal;

ao professor e orientador Tiago Almeida, por todo o enorme apoio e conselhos durante toda as etapas do Mestrado;

ao pesquisador Renato Moraes Silva, pelo auxílio durante as etapas finais de revisão desta dissertação;

à minha amada esposa Rita por todo o amor, carinho, inspiração e por sempre me incentivar e não deixar que eu desistisse;

à minha querida mãe Mariza, por sempre se importar com minhas tarefas e preocupações, por me motivar e por me ensinar a ser uma pessoa esforçada e capaz.

"Ensina a criança no Caminho em que deve andar,
e mesmo quando for idoso não se desviará dele!"
(Provérbios 22:6)

Resumo

Antes de comprar um produto ou escolher um destino de viagem, muitas pessoas costumam buscar por opiniões alheias para obter uma visão da qualidade daquilo que se deseja adquirir. Assim, as opiniões sempre exerceram grande influência na decisão de compra. Com o avanço da Internet e aumento no volume de informações trafegadas, surgiram redes sociais que possibilitam compartilhar e visualizar informações de todo o tipo, fazendo com que pessoas passassem a buscar também por opiniões na *Web*. Atualmente, sites especializados, como TripAdvisor e Yelp, oferecem um sistema de compartilhamento de opiniões *online* (*reviews*) de maneira fácil, pois possibilitam que usuários publiquem suas opiniões de qualquer lugar através de *smartphones*, assim como também permitem que fabricantes de produtos e prestadores de serviços obtenham *feedbacks* relevantes de maneira centralizada e rápida. Em virtude disso, estudos indicam que atualmente a maioria dos usuários confia tanto em recomendações pessoais quanto em *reviews online*. No entanto, a competição entre prestadores de serviços e fabricantes de produtos também aumentou nas redes sociais, o que levou aos primeiros casos de *spam reviews*: opiniões enganosas publicadas por pessoas contratadas que tentam promover ou difamar produtos ou serviços. Esses *reviews* são escritos cuidadosamente para parecerem autênticos, o que dificulta sua detecção por humanos ou por métodos automáticos. Assim, eles são usados para tentar, de maneira enganosa, controlar a opinião geral, podendo causar prejuízos para empresas e usuários. Diversas abordagens para a detecção de *spam reviews* vêm sendo propostas, sendo que a grande maioria emprega técnicas de aprendizado de máquina e processamento de linguagem natural. No entanto, apesar dos avanços já realizados, ainda há questionamentos relevantes que permanecem em aberto e demandam uma análise criteriosa para serem respondidos. Por exemplo, não há um consenso se o desempenho de métodos tradicionais de classificação pode ser afetado em cenários que demandam aprendizado incremental ou por mudanças nas características dos *reviews* devido ao fator cronológico, assim como também não há um consenso se existe diferença estatística entre os desempenhos dos métodos baseados no conteúdo das mensagens. Neste cenário, esta dissertação oferece uma análise e comparação compreensiva dos métodos tradicionais de aprendizado de máquina, aplicados na detecção de *spam reviews*. A comparação é realizada em múltiplos cenários, empregando-se diferentes tipos de aprendizado e bases de dados. Os experimentos realizados, juntamente com análise estatística dos resultados, corroboram a oferecer respostas adequadas para os questionamentos existentes. Além disso, os resultados obtidos podem ser usados como *baseline* para comparações futuras.

Palavras-chaves: Opiniões falsas; Classificação; Processamento de linguagem natural; Aprendizado de máquina.

Abstract

Before buying a product or choosing for a trip destination, people often seek other people's opinions to obtain a vision of the quality of what they want to acquire. Given that, opinions always had great influence on the purchase decision. Following the enhancements of the Internet and a huge increase in the volume of data traffic, social networks were created to help users post and view all kinds of information, and this caused people to also search for opinions on the Web. Sites like TripAdvisor and Yelp make it easier to share online reviews, since they help users to post their opinions from anywhere via smartphones and enable product manufacturers to gain relevant feedback quickly in a centralized way. As a result, most people nowadays trust personal recommendations as much as online reviews. However, competition between service providers and product manufacturers have also increased in social media, leading to the first cases of spam reviews: deceptive opinions published by hired people that try to promote or defame products or businesses. These reviews are carefully written in order to look like authentic ones, making it difficult to be detected by humans or automatic methods. Thus, they are used, in a misleading way, in attempt to control the general opinion, causing financial harm to business owners and users. Several approaches have been proposed for spam review detection and most of them use techniques involving machine learning and natural language processing. However, despite all progress made, there are still relevant questions that remain open, which require a criterious analysis in order to be properly answered. For instance, there is no consensus whether the performance of traditional classification methods can be affected by incremental learning or changes in reviews' features over time; also, there is no consensus whether there is statistical difference between performances of content-based classification methods. In this scenario, this work offers a comprehensive comparison between traditional machine learning methods applied in spam review detection. This comparison is made in multiple setups, employing different types of learning and data sets. The experiments performed along with statistical analysis of the results corroborate offering appropriate answers to the existing questions. In addition, all results obtained can be used as baseline for future comparisons.

Key-words: Spam reviews; Classification; Natural language processing; Machine learning.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

ARI	*Automated Readability Index* (Índice de Legibilidade Automatizado)
ATS	*Average Travel Speed* (Velocidade Média de Viagem)
AUC	*Area Under the Curve* (Área sob a curva)
AVP	*Amazon Verified Purchase* (Compra Verificada pela Amazon)
B.NB	Naïve Bayes Bernoulli
CD	*Critical Difference* (Diferença Crítica)
DT	*Decision Tree* (Árvore de Decisão)
FM	F-*Measure* (Medida-F)
FPR	*False Positive Rate* (Taxa de falsos positivos)
GSRank	*Group Spam Rank*
IP	*Internet Protocol* (Protocolo de Internet)
IQR	*Interquartile Range* (Amplitude Interquartil)
KNN	*k-Nearest Neighbors* (k-Vizinhos mais próximos)
M.NB	Naïve Bayes Multinomial
MDL	*Minimum Description Length* (Princípio da Descrição mais Simples)
NLP	*Natural Language Processing* (Processamento de Linguagem Natural)
PC	*Personal Computer* (Computador Pessoal)
RF	*Random Forest* (Floresta Aleatória)
SGD	*Stochastic Gradient Descent* (Gradiente Descendente Estocástico)
SVM	*Support Vector Machines* (Máquinas de Vetores de Suporte)
TF	*Term Frequency* (Frequência do termo)
TF-IDF	*Term Frequency-Inverse Document Frequency* (Frequência do termo-inverso da frequência nos documentos)
TPR	*True Positive Rate* (Taxa de verdadeiros positivos)

Lista de símbolos

$\hat{\alpha}$	Fator multiplicador de Lagrange utilizado no SVM
α	Grau de confiança utilizado no teste de Bonferroni-Dunn
β	Peso condicional do termo
Θ_N	Fator de crescimento de uma árvore
λ	Fator de regularização do SGD
$\vec{\mu}$	Centróide
ϕ	Função de perda de um classificador
Φ_Z	Função de transformação para um espaço de maior dimensionalidade
χ^2_F	Valor calculado no teste de Friedman
Ω	Fator de regularização do MDLText
ξ	Erro de classificação de uma amostra
$\|A\|$	Número de árvores de decisão do método Floresta Aleatória
b	Constante de viés
c	Classe disponível para classificação
$\mathcal{C}$	Conjunto de classes para classificação
$\|\mathcal{C}\|$	Número total de classes para classificação
C	Fator de regularização do SVM
d	Documento de texto
$\|d\|$	Número de termos de um documento
$\|\bar{d}\|$	Número médio de termos de um documento
$\|\bar{d}\|_{treino}$	Número médio de termos do documento de treinamento
$\|\bar{d}\|_{teste}$	Número médio de termos do documento de teste
$\mathcal{D}$	Conjunto de documentos

$\|\mathcal{D}\|$	Número total de documentos
$\mathcal{D}_c$	Conjunto de documentos pertencentes à classe c
$\|\mathcal{D}_c\|$	Número de documentos pertencentes à classe c
$\|\mathcal{D}\|_{treino}$	Número de documentos de treinamento
$\|\mathcal{D}^t\|$	Número de documentos em que o termo t aparece
$\mathcal{E}$	Valor esperado com relação a amostras e seus rótulos
E	Número de épocas de treinamento
f_p	Número de falsos positivos
f_n	Número de falsos negativos
F_F	Valor calculado no teste de Friedman para a distribuição F
F	Distribuição F
$F_{critico}$	Valor crítico encontrado na tabela de distribuição F
F_M	Função monotônica do SVM
g	Grau de liberdade utilizado no teste de Friedman
k	Número de vizinhos mais próximos
$\hat{k}$	Número de métodos de classificação
K	Função de *kernel* do SVM
$\hat{K}$	Função de penalidade do termo utilizada no MDLText
l	Taxa de aprendizado
L_ϕ	Função de perda do classificador com fator de regularização do SGD
$L(d\|c_i)$	Tamanho da descrição do documento quando descrito pela i-ésima classe
$L(t_j\|c_i)$	Tamanho da descrição do i-ésimo termo quando descrito pela i-ésima classe
$\mathcal{M}$	Mediana do número de termos por amostra
N	Número de bases de dados
$p(x)$	Função classificador

P	Precisão de um classificador
q_α	Parâmetro estatístico tabelado de acordo com o grau de confiança utilizado no teste de Bonferroni-Dunn
Q	Qualidade de um classificador
R	Revocação de um classificador
$\bar{R}$	*Ranking* médio de um método de classificação
$\hat{S}$	Função de penalidade do documento utilizada no MDLText
S	Constante de convergência do SGD
t_j	j-ésimo termo de um documento de texto
t_p	Número de verdadeiros positivos
t_n	Número de verdadeiros negativos
T	Número de ocorrências do termo
$\vec{v}$	Vetor normalizado de um documento
$\mathcal{V}$	Vocabulário de termos
$\|\mathcal{V}\|$	Número total de termos do vocabulário
$\|\vec{V}_{sup}\|$	Número de vetores de suporte do SVM
$\vec{w}$	Vetor de pesos
W	Soma de pesos TF-IDF
$\vec{x}$	Amostra representada como vetor de atributos
x_i	i-ésimo atributo de uma amostra
$\|\vec{x}\|$	Número total de atributos da amostra
X	Matriz de amostras
y_{true}	Rótulo correto da amostra
$\hat{y}$	Rótulo retornado pelo classificador
Y	Matriz de rótulos
Z	Espaço de maior ou infinita dimensionalidade

Sumário

Prefácio

Hoje em dia, com o avanço da Internet e o uso crescente de *smartphones*, é muito fácil postar e obter opiniões praticamente sobre qualquer coisa: produtos, serviços, estabelecimentos e até mesmo sobre pessoas ou fatos. As redes sociais alavancaram a facilidade de publicação, consulta e também o volume desse tipo de informação. Consequentemente, consumidores em potencial passaram a consultar opiniões alheias *online* antes de comprar um produto. Além disso, fabricantes de produtos passaram a se interessar por *feedbacks* com opiniões publicadas por consumidores.

São muitos os benefícios das opiniões manifestadas *online* (*reviews*), o que motivou o surgimento de sites especializados no compartilhamento dessas opiniões, como por exemplo o TripAdvisor[1] e o Yelp[2]. O sucesso desses sites pode ser justificado pela alta confiança que os usuários têm nas opiniões *online*. De acordo com pesquisas recentes, mais de 84% dos usuários confiam tanto em *reviews online* quanto em recomendações pessoais[3].

O sucesso das opiniões *online* não atraiu apenas a atenção dos usuários intencionados em fazer boas compras. Pelo contrário, a possibilidade de influenciar a opinião de um grande volume de pessoas facilmente também atraiu a atenção de grupos mal-intencionados que publicam *reviews* falsos. Esses *reviews* são escritos de maneira que pareçam autênticos e reflitam uma opinião positiva ou negativa sobre o objeto-alvo.

Essa prática está tornando-se cada vez mais comum e ficou conhecida como *spam* de opinião ou *spam review*. Em alguns casos, os autores (denominados *spammers*) são contratados por empresas para difamarem seus concorrentes ou então elevar indevidamente a popularidade de seus produtos.

A publicação de *reviews* falsos pode impactar rápida e drasticamente a imagem de produtos e serviços, podendo resultar em perdas financeiras. Neste cenário, dado que o volume de mensagens postadas diariamente frequentemente é altíssimo, são necessárias maneiras automáticas para se detectar este tipo de atividade fraudulenta, para que os usuários não sejam influenciados por *reviews* enganosos.

A detecção automática desse tipo de mensagem apresenta severas dificuldades, uma vez que até mesmo seres humanos frequentemente não percebem quando um *review* é enganoso. Harris (2012) demonstrou que as pessoas dificilmente conseguem diferenciar *spam reviews* de opiniões legítimas, obtendo taxa de acerto pouco maior do que 50%.

---

1 *TripAdvisor*. Disponível em: <https://www.tripadvisor.com>, acessado em 24/01/2017.
2 *Yelp*. Disponível em: <https://www.yelp.com>, acessado em 24/01/2017.
3 *Local Consumer Review Survey*. Disponível em <https://www.brightlocal.com/learn/local-consumer-review-survey/>, acessado em 24/01/2017.

Nos últimos anos, surgiram estudos na literatura que exploram este tema, propondo aplicar métodos de aprendizado de máquina e técnicas de processamento de linguagem natural na tarefa de classificar automaticamente *spam reviews* com base no conteúdo das mensagens. Entretanto, ainda há uma série de questões importantes em aberto que demandam estudos criteriosos para serem respondidas. Dentre elas, destacam-se:

Q1: O desempenho dos métodos pode ser afetado pela polaridade das opiniões (reclamações × elogios)?

Q2: Uma vez que grande parte dos trabalhos da literatura utilizaram bases artificiais de *spam*, o desempenho dos métodos é preservado na classificação de amostras reais?

Q3: O desempenho dos métodos pode ser afetado pela diversidade de objetos de análise (treinamento com opiniões de múltiplos tipos de serviço/mercadoria × um único tipo de serviço/mercadoria)?

Q4: O desempenho dos métodos pode ser afetado pelas mudanças de características dos *reviews* por causa do fator cronológico?

Q5: O desempenho dos métodos pode ser afetado em cenários que demandam aprendizado incremental?

Q6: O desempenho dos métodos pode ser afetado ao se utilizar bases de dados desbalanceadas?

Q7: Existe diferença estatística entre os desempenhos dos métodos baseados em conteúdo?

Objetivos e contribuições

O principal objetivo desta dissertação é oferecer uma comparação e análise compreensiva do desempenho de diversos métodos considerados o estado-da-arte na literatura em aprendizado de máquina, aplicados na detecção automática de *spam reviews* com base no conteúdo das mensagens. Nesta análise, além de serem avaliados diferentes métodos, o objetivo é realizar também uma avaliação em diferentes cenários, com características distintas de aprendizado e bases de dados, de tal forma que os resultados encontrados possam fornecer subsídios para responder as questões de pesquisa apresentadas acima.

Em resumo, as principais contribuições oferecidas neste trabalho são:

1. apresentar um estudo geral sobre o problema de *spam review*, bem como oferecer uma revisão e análise dos trabalhos existentes na literatura;

2. aplicar e comparar métodos considerados o estado-da-arte em categorização de texto;

3. apresentar resultados experimentais realizados em diferentes cenários, envolvendo aprendizado *offline* e *online*;
4. oferecer resultados *baseline* para auxiliar comparações futuras; e
5. oferecer respostas adequadas para as principais questões em aberto que permeiam o problema estudado.

Organização

Esta dissertação está estruturada da seguinte maneira:

- no Capítulo 1, o problema é caracterizado, introduzido e contextualizado com relação aos principais trabalhos existentes na literatura;
- no Capítulo 2, as principais formas de representação computacional de texto são sucintamente apresentadas;
- no Capítulo 3, são brevemente apresentados os principais métodos de classificação;
- no Capítulo 4, são detalhados a metodologia experimental, bem como os resultados obtidos e a análise estatística realizada;
- por fim, no Capítulo 5, são apresentadas as considerações finais e direcionamentos para trabalhos futuros.

1 As opiniões falsas

Hoje em dia, sites como o TripAdvisor oferecem opiniões (*reviews*) sobre lugares como hotéis, restaurantes, etc. Sites de *e-commerce* como a Amazon[1] também permitem a postagem de *reviews* sobre produtos. Essa facilidade faz com que pessoas confiem cada vez mais em opiniões *online*, assim como também aumenta a competitividade entre fabricantes de produtos e donos de estabelecimentos comerciais, para ganhar visibilidade nessas mídias. Este foi o cenário onde começaram a surgir os primeiros casos de opiniões falsas publicadas *online*: *reviews* falsos publicados em redes sociais por usuários contratados, com o objetivo de difamar ou promover produtos ou serviços.

A Figura 1 demonstra um exemplo real contendo opiniões falsas publicadas na Amazon. Neste exemplo, dois usuários criaram *reviews* falsos para aumentar a popularidade de três produtos específicos, sendo que os autores publicaram *reviews* com avaliações 5 estrelas (nota máxima) para os mesmos produtos, porém em datas distintas.

Figura 1 – Exemplos reais de opiniões falsas na Amazon.

1 of 1 people found the following review helpful: ★★★★★ **Practically FREE music**, December 4, 2004 This review is from: **Audio Xtract (CD-ROM)** I can't believe for $10 (after rebate) I got a program that gets me free unlimited music. I was hoping it did half what was	2 of 2 people found the following review helpful: ★★★★★ **Like a tape recorder...**, December 8, 2004 This review is from: **Audio Xtract (CD-ROM)** This software really rocks. I can set the program to record music all day long and just let it go. I come home and my ....
3 of 8 people found the following review helpful: ★★★★★ **Yes – it really works**, December 4, 2004 This review is from: **Audio Xtract Pro (CD-ROM)** See my review for Audio Xtract - this PRO is even better. This is the solution I've been looking for. After buying iTunes, ...	3 of 10 people found the following review helpful: ★★★★★ **This is even better than...**, December 8, 2004 This review is from: **Audio Xtract Pro (CD-ROM)** Let me tell you, this has to be one of the coolest products ever on the market. Record 8 internet radio stations at once, ...
5 of 5 people found the following review helpful: ★★★★★ **My kids love it**, December 4, 2004 This review is from: **Pond Aquarium 3D Deluxe Edition** This was a bargain at $20 - better than the other ones that have no above water scenes. My kids get a kick out of the	5 of 5 people found the following review helpful: ★★★★★ **For the price you...**, December 8, 2004 This review is from: **Pond Aquarium 3D Deluxe Edition** This is one of the coolest screensavers I have ever seen, the fish move realistically, the environments look real, and the ....

Fonte: Mukherjee, Liu e Glance (2012) (adaptado).

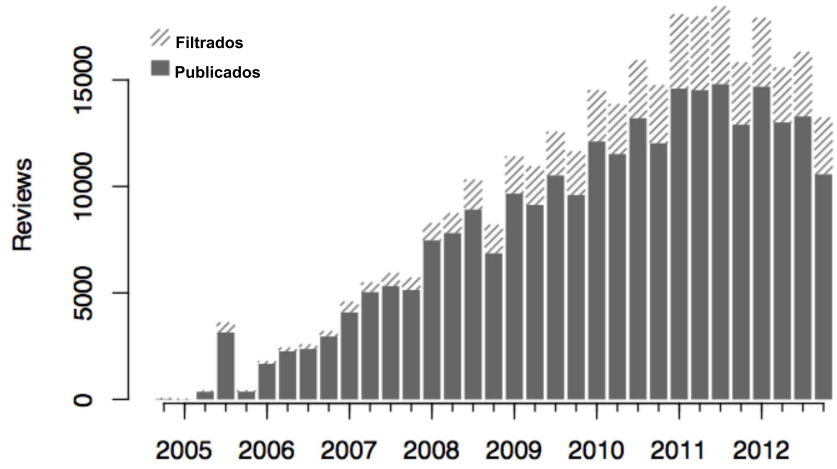
Essas opiniões falsas publicadas *online* são denominadas *spam* de opinião, ou então: *spam reviews*, *fake reviews*, *deceptive reviews*, *fraudulent reviews*. Seus autores são denominados *spammers*. *Spam reviews* causam um grande prejuízo para empresas fabricantes de produtos ou provedores de serviços, que muitas vezes são prejudicados com opiniões falsas negativas ou por *fake reviews* promovendo rapidamente seus concorrentes. Um exemplo desse tipo de fraude, que virou notícia, foi o caso de um *Chef* de um restaurante na Inglaterra que publicou opiniões difamando seus concorrentes no site TripAdvisor. Após a descoberta, o *Chef* foi demitido (TAYLOR, 2015; REPORTER, 2015).

[1] *Amazon*. Disponível em <https://www.amazon.com/>, acessado em 25/01/2017.

Em outro caso[2], a Samsung foi processada por contratar pessoas para escrever *reviews* falsos *online* atacando a empresa concorrente HTC e, ao mesmo tempo, promovendo sua própria marca.

Existem sites que possuem filtros automáticos deste tipo de *spam*. Um exemplo é o Yelp[3], site especializado em *reviews* sobre estabelecimentos comerciais, que possui um algoritmo proprietário para filtragem de avaliações falsas. Através deste algoritmo, usuários visualizam primeiro os *reviews* que são mais relevantes e que possuem maior probabilidade de serem autênticos. Consequentemente, os *reviews* indesejados e menos relevantes encontram-se na categoria "*reviews* não recomendados pelo Yelp" e são ocultados para o usuário. Porém, se o usuário decidir visualizar o conteúdo desta categoria, o Yelp permite que tais *reviews* não recomendados sejam visualizados. A Figura 2 apresenta a quantidade de *reviews* processados pelo algoritmo do Yelp ao longo dos anos, por trimestre; as barras mais claras mostram o volume de *reviews* filtrados (*spam reviews*) e as mais escuras indicam os *reviews* publicados.

Figura 2 – Número de *spam reviews* no Yelp.



Fonte: Luca e Zervas (2015) (adaptado).

Outro exemplo de mecanismo que tenta minimizar as ocorrências de opiniões falsas é uma funcionalidade da Amazon. Trata-se de um indicador de compra denominado AVP (*Amazon Verified Purchase*). O AVP indica, em um *review*, se o autor efetuou a compra daquele produto. Este recurso tenta minimizar uma possível desconfiança em *reviews online* e parte do esforço da Amazon em combater este tipo de atividade fraudulenta. De acordo com Perez (2016), a Amazon investigou e está processando vendedores de produtos

[2] ABC News. Disponível em: <http://abcnews.go.com/Technology/samsung-fined-paying-people-criticize-htcs-products/story?id=20671547>. Acessado em: 05/04/2017.

[3] *Yelp*. Disponível em <https://www.yelp.com/>, acessado em 25/01/2017.

que criaram *spam reviews* para promover seus produtos e com isso enganaram usuários do site.

De maneira geral, os *spam reviews* são um problema crescente na Internet. De acordo com Luca e Zervas (2015), 16% dos *reviews* sobre restaurantes do Yelp são *spam*. Pesquisas[4] também apontam que, até 2014, entre 10% e 15% de *reviews* em mídias sociais eram falsos.

Devido ao crescente aumento no volume de opiniões falsas, redes sociais passaram a adotar mecanismos para que usuários denunciem publicações que possivelmente sejam *spam*. Apesar disso, de acordo com Harris (2012), é difícil que humanos consigam detectar com precisão os *spam reviews*, pois os mesmos são cuidadosamente criados para parecerem autênticos.

Para ilustrar essa dificuldade, são apresentados a seguir dois exemplos de *reviews* reais, sendo que um é legítimo e o outro é *spam*:

> *I have stayed at many hotels traveling for both business and pleasure and I can honestly stay that The James is tops. The service at the hotel is first class. The rooms are modern and very comfortable. The location is perfect within walking distance to all of the great sights and restaurants. Highly recommend to both business travellers and couples.*

> *My husband and I stayed at the James Chicago Hotel for our anniversary. This place is fantastic! We knew as soon as we arrived we made the right choice! The rooms are BEAUTIFUL and the staff very attentive and wonderful!! The area of the hotel is great, since I love to shop I couldn't ask for more!! We will definatly be back to Chicago and we will for sure be back to the James Chicago.*

Neste cenário, é desejável que haja formas eficazes e robustas de detectar automaticamente este tipo de atividade fraudulenta e filtrar os *spam reviews* para que não sejam exibidos para os usuários. Na literatura, a maioria dos trabalhos que objetivam realizar a detecção de *spam reviews* utilizam técnicas de aprendizado de máquina e processamento de linguagem natural (*natural language processing* – NLP).

De acordo com Heydari et al. (2015), os trabalhos existentes na literatura podem ser agrupados em 3 categorias: (1) detecção de *spam reviews*, (2) detecção de *spammers* e (3) detecção de grupos de *spammers*. Cada uma dessas categorias de trabalhos são apresentadas a seguir.

[4] *Gartner Newsroom*. Disponível em: <http://www.gartner.com/newsroom/id/2161315>. Acessado em: 05/04/2017.

1.1 Detecção de *spam reviews*

Os estudos de Jindal e Liu (2007b) foram os pioneiros na área de detecção de *spam reviews*. Eles criaram a primeira base de dados pública, com milhões de *reviews* coletados da Amazon. Segundo os autores, os *spams* encontrados podem ser divididos em três tipos:

1. Opiniões falsas: *reviews* enganosos que promovem ou difamam instituições ou produtos;

2. Opiniões sobre marcas: *reviews* vinculados a um produto, porém que analisam somente a marca ou empresa que o fabrica;

3. Mensagens sem nenhuma opinião: *reviews* que na verdade contém propagandas ou textos não relacionados com o tema ou produto.

Segundo Jindal e Liu (2007b), os tipos 2 e 3 são mais fáceis de serem detectados através de métodos tradicionais de aprendizado de máquina, porém o tipo 1 apresenta maior dificuldade, uma vez que é mais complexo de ser detectado por humanos, dificultando a criação de uma base de dados rotulada necessária para treinar métodos de classificação.

De acordo com Jindal e Liu (2007a), técnicas empregadas na detecção de *Web spam*[5] ou *email spam*[6] não são adequadas para se detectar opiniões falsas. Os autores aplicaram um método para encontrar *reviews* duplicados ou aproximadamente duplicados (BRODER, 1997). Com isso, uma base de dados foi criada, onde os *reviews* duplicados foram rotulados como *spam* e os demais foram rotulados como não-*spam* (legítimos).

Jindal e Liu (2008) utilizaram classificação através de Regressão Logística e concluíram que novas abordagens precisariam ser criadas para se aumentar a eficácia da detecção, uma vez que os *reviews* rotulados como "não-*spam*" podem conter ruídos, como por exemplo *reviews* escritos cuidadosamente para parecerem autênticos (não sendo necessariamente duplicados de outros *reviews*).

Ott et al. (2011) estudaram detecção de opiniões falsas escritas para parecerem autênticas, das quais o método proposto por Jindal e Liu (2007a) apresentaria dificuldades (HEYDARI et al., 2015). Foram testados atributos de psicolinguística para se detectar sentenças enganosas no texto, representados por unigramas, bigramas e trigramas. A base de dados utilizada é composta de *reviews* legítimos e positivos sobre um conjunto de hotéis do TripAdvisor e os *spam reviews* foram criados manualmente, com ajuda do *site* Amazon

[5] Técnicas para enganar motores de busca e retornar *websites* desejados nas primeiras posições de buscas.

[6] Propagandas contidas em emails para atrair usuários para sites maliciosos.

Mechanical Turk[7]. Segundo os autores, a melhor acurácia obtida[8] foi de aproximadamente 89%.

Ott, Cardie e Hancock (2013) estenderam o trabalho de Ott et al. (2011), analisando bases de *reviews* somente com polaridade negativa (reclamações/difamações). Os autores publicaram a primeira base *gold-standard* de *spam reviews* com polaridade negativa. Os *spam reviews* foram escritos por pessoas contratadas no site Amazon Mechanical Turk e os *reviews* legítimos foram extraídos de diversos sites, dentre eles TripAdvisor, Hotels.com[9] e Yelp. Os autores avaliaram o desempenho do SVM utilizando como atributos unigramas e bigramas. Foi empregada validação cruzada com 5-*fold* e o melhor resultado obtido apresentou acurácia de 86%.

Harris (2012) utilizou uma base de dados com *spam reviews* criados manualmente, porém acrescentou *reviews* com avaliações negativas. Segundo o autor, seres humanos detectam *hyper spam* (*reviews* falsos promovendo algo) mais facilmente do que *defaming spam* (*reviews* falsos usados para difamar a imagem de algo). Foram utilizados atributos contendo métricas sobre a complexidade dos textos dos *reviews*, como por exemplo a medida ARI (*Automated Readability Index*). Foram realizados testes observando a taxa de acerto da detecção por humanos, primeiramente utilizando apenas o texto dos *reviews* para a classificação e depois fornecendo as métricas de complexidade dos textos. A acurácia do classificador[10] foi de aproximadamente 76%, enquanto que a acurácia dos humanos foi de aproximadamente 58% no primeiro teste e 65% no segundo. Assim, o autor concluiu que *websites* podem oferecer as métricas de complexidade dos textos para auxiliar usuários a denunciarem opiniões falsas manualmente.

Mukherjee et al. (2013) reproduziram o método de Ott et al. (2011) com uma base contendo opiniões falsas do mundo real. Os autores extraíram dados do Yelp, que possui um sistema próprio de filtro de *reviews*. A acurácia do mesmo método proposto em Ott et al. (2011) diminuiu para menos de 70% com os dados reais. Os autores compararam a eficácia de atributos linguísticos, estatísticos e comportamentais contidos nos *reviews*. Os atributos linguísticos utilizados por Ott et al. (2011) falharam em obter uma boa acurácia com os dados do Yelp. Com isso, os autores concluíram que métodos de classificação treinados utilizando *spam reviews* artificiais (escritos pelos próprios autores da pesquisa ou por ajudantes) não servem para classificar e filtrar *spam reviews* do mundo real.

KC e Mukherjee (2016) realizaram uma análise sobre os *spam reviews* de restaurantes do Yelp, com o objetivo de encontrar padrões temporais de comportamento dos *spammers*. Os autores utilizaram séries temporais sobre as postagens de *reviews* genuínos e *reviews* enganosos. Após a análise, foi descoberto que existe uma forte correlação entre

[7] Amazon Mechanical Turk. Disponível em `https://mturk.com`. Acessado em Janeiro de 2016.
[8] Foram utilizados os métodos Naïve Bayes e SVM para classificação.
[9] *Hotels.com*. Disponível em: <https://www.hotels.com>. Acessado em: 11/04/17.
[10] O autor utilizou SVM como método de classificação.

os comportamentos de postagem de *spam* e as séries temporais de postagem de *reviews* genuínos. Com isso, os autores categorizaram três padrões de postagens de *spam reviews* promovendo restaurantes:

- *Early*: restaurantes começam a realizar *spamming* em aproximadamente 5 meses após início de publicação de *reviews*;
- *Mid*: restaurantes iniciam *spamming* após 14 meses;
- *Late*: restaurantes iniciam *spamming* após 30 meses.

Além desses padrões de postagem, os autores realizaram uma análise causal dos comportamentos de *spamming* nas séries temporais. Eles descobriram uma relação de causa do *spamming* em relação ao aumento de *reviews* negativos e diminuição de *reviews* positivos. Com isso, eles dividiram os comportamentos de *spamming* em duas categorias:

- *Spamming bufferizado*: restaurantes esperam uma diminuição da popularidade (aumento de *reviews* negativos ou diminuição de *reviews* positivos) para iniciar a criação de *spam reviews* positivos;
- *Reduced spamming*: comportamentos de *spamming* que são reduzidos após os restaurantes perceberem um aumento na popularidade (aumento de *reviews* positivos e diminuição de *reviews* negativos).

Após essas análises, os autores extraíram atributos provenientes das séries temporais, para utiliza-los na tarefa de classificação de *spam reviews*. Abaixo, são listados alguns desses atributos:

- Desvio padrão da classificação (*rating*) de *reviews* genuínos na semana anterior;
- Média da classificação genuína dos *reviews* postados na semana anterior;
- Classificação genuína dos *reviews* positivos postados na semana anterior;
- Classificação genuína dos *reviews* negativos postados na semana anterior.

Para validar os atributos propostos, os autores realizaram experimentos utilizando o método SVM com validação cruzada 5-*fold*, utilizando os atributos de séries temporais, atributos *N-grams*, atributos comportamentais[11] e a combinação de todos os atributos. Por fim, de acordo com os autores, houve uma melhora significativa no desempenho quando foram utilizados todos os tipos de atributos, obtendo Medida-F aproximadamente igual a 0,9.

[11] Estes atributos são propostos por Mukherjee et al. (2013).

Najada e Zhu (2014) propuseram um método para lidar com bases de *reviews* com desbalanceamento de classes. O método proposto consiste em dividir uma base de *spam reviews* desbalanceada em múltiplas bases menores; em seguida, aplica-se *random under-sampling*[12] para transformá-las em bases balanceadas. Por fim, são gerados classificadores após treinamento com cada uma das bases e o rótulo final corresponde ao que foi retornado pela maioria deles. As bases de *reviews* utilizadas contém *spam reviews* disponibilizados por Ott et al. (2011) e *reviews* legítimos disponibilizados por Ganesan e Zhai (2012). Assim, o número de *spam reviews* é bem menor que o número de *reviews* legítimos (NAJADA; ZHU, 2014). O método proposto obteve desempenho superior ao método C4.5 (QUINLAN, 2014), sendo que o valor de TPR (*true positive rate*) variou de 75% a 80%, aumentando conforme o desbalanceamento das bases aumentava, enquanto que o TPR para o *baseline* foi decaindo de 59% para 29%. Da mesma forma, os valores de FPR (*false positive rate*) foram bem menores para o método proposto, decaindo de 22% a 18%, conforme aumento do desbalanceamento das bases utilizadas; o FPR para o *baseline* aumentou de 41% a 71%.

Li et al. (2015) analisaram *spam reviews* do mundo real, através de uma parceria com um site da China especializado em *reviews*[13]. Os autores utilizaram uma base disponibilizada pelo site, que contém aproximadamente 6 milhões de *reviews* de restaurantes de Shangai. De acordo com os autores, este é o primeiro trabalho na literatura a investigar o problema de opiniões falsas utilizando uma base de *reviews* do mundo real com um volume grande de dados. Os autores analisaram atributos espaciais e temporais e propuseram um conjunto novo de atributos. O conjunto é composto por:

- *Booleano* indicando se o usuário é registrado no site;
- *Booleano* indicando se o usuário é registrado entre terça-feira e quinta-feira;
- Distância entre a cidade do usuário e Shangai;
- ATS (*Average Travel Speed*)[14];
- Porcentagem de *reviews* escritos no final de semana;
- Porcentagem de *reviews* postados através de PC;
- Distância média entre as cidades onde o usuário posta *reviews* e Shangai;
- Desvio absoluto médio da classificação (*rating*) dos *reviews* do usuário;
- Número de *IPs* únicos utilizados pelo usuário;

[12] A técnica de *random under-sampling* consiste em remover amostras da classe majoritária.
[13] *Dianping.com*. Disponível em <http://dianping.com>. Acessado em 08/06/2017.
[14] Atributo que simula a sequência de viagem de um usuário; seu valor é calculado usando a média da velocidade de viagem para cada cidade visitada (LI et al., 2015).

- Número de *cookies* únicos utilizados pelo usuário;
- Número de cidades únicas nas quais o usuário escreve *reviews*.

Os autores utilizaram o método SVM empregando atributos linguísticos sugeridos por Ott et al. (2011), atributos comportamentais descritos por Mukherjee et al. (2013) e os novos atributos propostos, além de uma combinação de todos os atributos. Os resultados obtidos consistem na média utilizando validação cruzada 10-*fold*. Os atributos propostos geraram resultados superiores aos dos outros atributos, obtendo um valor Medida-F igual a $0,83$. O melhor resultado foi obtido utilizando a combinação de todos os atributos, no qual foi obtida Medida-F igual a $0,85$.

1.2 Detecção de *spammers*

Jindal, Liu e Lim (2010) propuseram modelar regras pré-definidas a partir de atributos dos usuários e com isso definir regras não usuais que são utilizadas para calcular a probabilidade de que um usuário esteja escrevendo opiniões falsas. Os autores realizaram um estudo de caso[15] e os resultados indicaram que o método foi capaz de reconhecer usuários com alto grau de suspeita. Por exemplo, foi detectado um usuário que escreveu 626 *reviews*, todos com avaliações positivas. Outro exemplo foi um usuário que escreveu 30 *reviews* com avaliações positivas para a mesma marca de produto. Por fim, os autores concluíram que é mais fácil detectar usuários *spammers* do que *spam reviews*.

Fei et al. (2013) estudaram picos de *reviews* postados ao longo do tempo. Segundo os autores, esses picos acontecem por dois motivos: popularidade momentânea de algum produto em lançamento ou em promoção; ou ataques de *spam reviews*. Os autores definiram atributos indicadores de *spammer* (*spammer indicators*), como por exemplo:

- Porcentagem de *reviews* encontrados em picos;
- Porcentagem de *reviews* parecidos com outros *reviews*;
- Porcentagem de *reviews* encontrados tanto em picos de produtos quanto picos do próprio autor;
- Divergência entre a avaliação do autor e média de avaliações de um produto; e
- AVP (*Amazon Verified Purchase*).

Os autores modelaram o problema como um grafo e utilizaram um algoritmo de propagação de mensagens[16] para estimar a probabilidade de um usuário ser *spammer*,

[15] A base de dados utilizada foi obtida por Jindal e Liu (2007b).
[16] *Markov Random Fields* com *Loopy Belief Propagation* (FEI et al., 2013).

legítimo ou misto[17], com base nos atributos indicadores descritos. Os autores concluíram que é mais fácil detectar *spammers* através dos atributos indicadores do que detectar opiniões falsas utilizando somente o conteúdo dos *reviews*. Apesar do método proposto por Fei et al. (2013) ser baseado em picos de *reviews*, os autores buscaram apenas detectar *spammers* de maneira individual e não investigaram a fundo uma possível colaboração entre eles (HEYDARI et al., 2015).

1.3 Detecção de grupos de *spammers*

De acordo com Heydari et al. (2015), até 2015, os trabalhos referentes a essa área representavam menos de 7% dos trabalhos existentes na literatura sobre *spam reviews*.

Mukherjee, Liu e Glance (2012) estudaram a possibilidade de existir colaboração entre *spammers*. Nesse sentido, os autores propuseram técnicas para detectar grupos de *spammers*. O método proposto inicialmente cria grupos de usuários[18] através de regras de associação. Nesses grupos, todos os usuários escreveram *reviews* para os mesmos produtos. Esses são os chamados "candidatos a grupos de *spammer*". Os autores definiram atributos para indicar o grau de *spam* para os grupos de usuários (*group spam indicators*) e também para os membros do grupo (*individual spam indicators*). Além desses atributos, os autores criaram modelos binários para representar as relações de dependência entre grupos, seus membros individuais e produtos. O algoritmo criado pelos autores é denominado GSRank (*Group Spam Rank*) e iterativamente atualiza os graus de *spam* de cada entidade (grupos, membros ou produtos[19]) com base em suas dependências em relação às entidades restantes.

Rayana e Akoglu (2015) propuseram um método não-supervisionado baseado em um grafo de *reviews*, usuários e produtos. O grafo também contém uma lista de nós representando os rótulos, que podem ser para qualquer tipo de entidade (*review*, usuário ou produto). O método então iterativamente categoriza os três tipos (*spam reviews*, autores *spammers* e produtos passíveis de ataque *spam*), através de um algoritmo baseado em propagação de mensagens. O método proposto utiliza vários tipos de metadados sobre o texto de *reviews*, data de postagem, comportamentos dos autores e estatísticas sobre os produtos, que são empregados juntamente com parâmetros de compatibilidade entre os nós do grafo. O método iterativamente propaga mensagens através da rede, atualizando o valor dos nós de rótulos. Quando os valores dos rótulos não se alteram após um número pré-definido de iterações, o algoritmo encerra e os rótulos são retornados. Os autores realizaram experimentos com três bases de dados do Yelp e o método proposto obteve AUC (*Area under the curve*) = 69%.

[17] Usuário comum, mas que publica opiniões falsas às vezes.
[18] Foram utilizados dados da Amazon publicados por Jindal e Liu (2007b).
[19] No caso dos produtos, o grau de *spam* significa o "grau de ataques de *spam*" sobre um determinado produto.

Li et al. (2017) descobriram um padrão de distribuição bimodal nas postagens de *reviews*, analisando a base utilizada em Li et al. (2015). Adicionalmente, os autores reportaram que usuários tendem a postar *reviews* em grupo, tendo como alvo um determinado conjunto de produtos: este comportamento é denominado *co-bursting*. Neste cenário, os autores propuseram um modelo baseado no *Hidden Markov Model*, com a adição dos rótulos (*Labeled Hidden Markov Model*) para a classificação de *spammers*. Os atributos utilizados consistem apenas na data de publicação dos *reviews*. Em seguida, os autores estenderam o modelo para o *Coupled Hidden Markov Model*, onde foram adicionados também os sinais de *co-bursting*. Também, foram realizados testes para avaliar se os sinais de *co-bursting* analisados são eficientes. Para isso, os autores modelaram uma rede com relações *reviewer*-produto com comportamento de *co-reviewing*[20]. Eles também modelaram uma rede de *co-bursting* utilizando parâmetros desconhecidos (*hidden states*) a partir do modelo proposto. Com essas redes, eles mediram a qualidade da *clusterização*, atribuindo rótulos (usuário *spammer* ou genuíno) para cada *cluster* para avaliar a qualidade da classificação de *spammers*. Os autores utilizaram três algoritmos de *clusterização* para suportar o grande volume de dados utilizado. De acordo com os autores, as redes utilizando parâmetros de *co-bursting* apresentaram uma qualidade superior de *clusterização* e, por sua vez, de classificação de usuários *spammers*.

1.4 Considerações finais

De acordo com Crawford et al. (2015), a abordagem mais utilizada na literatura é o aprendizado supervisionado utilizando representação espaço-vetorial gerada por *bag of words*. Além disso, a maioria dos trabalhos emprega o SVM, Regressão Logística ou Naïve Bayes, sendo o SVM o método que obteve o melhor resultado na maioria dos casos. Segundo os autores, a maioria dos trabalhos utiliza bases de dados com *spam reviews* criados manualmente devido à difícil obtenção de uma base de dados *gold-standard*.

Apesar dos trabalhos existentes e estudos realizados na literatura, ainda há uma série de questões importantes sem respostas, tais como se o desempenho dos métodos pode ser afetado pela utilização de (*i*) *reviews* com polaridades distintas, (*ii*) amostras reais de *spam reviews*, (*iii*) *reviews* sobre múltiplos tipos de produtos ou serviços, (*iv*) *reviews* que sofrem mudanças de características ao longo do tempo, (*v*) aprendizado incremental e (*vi*) desbalanceamento. Além disso, não há um consenso se existe diferença estatística entre os desempenhos dos métodos de classificação existentes.

Neste cenário, esta dissertação oferece uma comparação compreensiva dos desempenhos de diversos métodos tradicionais de aprendizado de máquina, aplicados na detecção de *spam reviews* com base no conteúdo das mensagens. Apenas o conteúdo dos *reviews*

[20] Atributos sugerido por Mukherjee, Liu e Glance (2012) para detecção de grupos de *spammers* escrevendo *spam reviews* para um produto alvo.

foi utilizado para os experimentos, pois dados adicionais sobre produtos ou usuários normalmente são de propriedade das redes sociais onde os *reviews* são compartilhados. Com esses atributos, os experimentos foram realizados em múltiplos cenários, com diferentes características de aprendizado e bases de dados, com a finalidade de responder aos questionamentos propostos e oferecer resultados que poderão ser utilizados como *baseline* em trabalhos futuros.

Para avaliar métodos de classificação de *spam reviews*, é preciso primeiro entender como representar o texto dos *reviews* de maneira apropriada, para que possam servir como parâmetro de entrada para os classificadores.

2 Representação computacional

Os métodos de aprendizado de máquina são incapazes de processar diretamente os textos em linguagem natural. Desta forma, é preciso que eles sejam convertidos para vetores numéricos, para que possam ser devidamente manipulados (SEBASTIANI, 2002).

Para esta conversão, a forma mais tradicionalmente empregada é modelo espaço-vetorial, que consiste na segmentação do texto em múltiplos termos ou *tokens*. Um *token* representa a menor granularidade do texto desejada para uma determinada aplicação e pode corresponder a uma palavra, várias palavras em sequência, ou mesmo um símbolo. Este procedimento, também conhecido como *tokenização*, utiliza um ou mais caracteres como delimitadores para separar os termos. Por exemplo, a regra mais básica para segmentação é utilizar apenas o espaço em branco como delimitador.

Em seguida, define-se o vocabulário, que consiste no conjunto de termos de todos os documentos. Para exemplificar, abaixo são listadas algumas amostras, sendo que cada uma consiste em um exemplo diferente de *review*:

1. *The hotel rooms were so great*
2. *We had a great time at this hotel great stay*
3. *The rooms service is bad*
4. *The hotel rooms were so great, were so comfort*

Após converter todas as palavras dos documentos para letras minúsculas e aplicar a segmentação utilizando quaisquer caracteres não alfa-numéricos como delimitadores, obtém-se o vocabulário $\mathcal{V} = \{$ *at, bad, comfort, great, had, hotel, is, rooms, service, so, stay, the, this, time, we, were* $\}$.

É comum variar a granularidade de *tokens*, onde neste caso o vocabulário passa a ser composto por *N-grams*. O cenário mais simples é para $N = 1$ (*1-gram* ou unigrama), onde o *token* corresponde a um único termo. Para $N = 2$ (*2-gram* ou bigrama), o *token* corresponde à junção de dois termos subsequentes de tamanho $N = 1$ e assim por diante. Por exemplo, um vocabulário *2-gram* para as amostras de *reviews* citadas anteriormente seria de $\mathcal{V}_{2-gram} = \{$ *at this, great stay, great time, great were, had great, hotel great, hotel rooms, is bad, rooms service, rooms were, service is, so comfort, so great, the hotel, the rooms, this hotel, time at, we had, were so* $\}$.

Após a segmentação e construção do vocabulário, é possível então realizar a indexação dos *tokens* e, assim, passar a representar cada amostra de texto utilizando um

vetor com atributos númericos. Para isso, o modelo *bag of words* é um dos mais utilizados em categorização de texto e processamento de linguagem natural (SEBASTIANI, 2002).

O *bag of words* permite obter uma representação espaço-vetorial das amostras de texto, onde cada mensagem corresponde a um conjunto de palavras, sem considerar características como gramática ou sequência. Neste modelo, considerando que cada documento d é composto por um conjunto de termos $\{t_1, ..., t_{|\vec{x}|}\}$, pode-se representar cada documento como um vetor de atributos $\vec{x} = \{x_1, ..., x_{|\vec{x}|}\}$, onde $x_1, ..., x_{|\vec{x}|}$ são valores relacionados aos termos $t_1, ..., t_{|\vec{x}|}$. No formato mais simples, cada atributo corresponde à ocorrência ($x_i = 1$) ou não ocorrência de um termo ($x_i = 0$) no vocabulário (peso binário). A Tabela 1 ilustra a representação vetorial *1-gram*, com pesos binários, dos *reviews* apresentados anteriormente.

Tabela 1 – Exemplo de representação vetorial com atributos binários.

	at	*bad*	*comfort*	*great*	*had*	*hotel*	*is*	*rooms*	*service*	*so*	*stay*	*the*	*this*	*time*	*we*	*were*
(1)	0	0	0	1	0	1	0	1	0	1	0	1	0	0	0	1
(2)	1	0	0	1	1	1	0	0	0	0	1	0	1	1	1	0
(3)	0	1	0	0	0	0	1	1	1	0	0	1	0	0	0	0
(4)	0	0	1	1	0	1	0	1	0	1	0	1	0	0	0	1

Outra maneira de representar os atributos é através do esquema de pesos $tf(t_i, d)$, que indica a frequência local de cada termo t_i em um documento d (*term frequency* - TF). Para normalizar os valores, frequentemente realiza-se a transformação dos mesmos em escala logarítmica (WILBUR; KIM, 2009), como mostrado na Equação 2.1. A Tabela 2 apresenta a representação vetorial utilizando a frequência TF dos termos como esquema de pesos dos atributos.

$$TF(t_i, d) = log(1 + tf(t_i, d)) \tag{2.1}$$

Tabela 2 – Exemplo de representação vetorial; os atributos indicam a frequência normalizada de ocorrência de cada termo (TF).

	at	*bad*	*comfort*	*great*	*had*	*hotel*	*is*	*rooms*	*service*	*so*	*stay*	*the*	*this*	*time*	*we*	*were*
(1)	0.00	0.00	0.00	0.41	0.00	0.41	0.00	0.41	0.00	0.41	0.00	0.41	0.00	0.00	0.00	0.41
(2)	0.30	0.00	0.00	0.60	0.30	0.30	0.00	0.00	0.00	0.00	0.30	0.00	0.30	0.30	0.30	0.00
(3)	0.00	0.45	0.00	0.00	0.00	0.00	0.45	0.45	0.45	0.00	0.00	0.45	0.00	0.00	0.00	0.00
(4)	0.00	0.00	0.28	0.28	0.00	0.28	0.00	0.28	0.00	0.55	0.00	0.28	0.00	0.00	0.00	0.55

Adicionalmente, outro esquema de pesos bastante utilizado juntamente com métodos de aprendizado de máquina é o TF-IDF (*term frequency-inverse document frequency*), que consiste na divisão do peso TF pela frequência do número de documentos em que o termo aparece. Esta métrica é similar ao TF, porém se o termo aparecer na maioria dos documentos, então sua importância diminui. Ele é calculado pela Equação 2.2 (WILBUR; KIM, 2009), onde $|\mathcal{D}|$ indica o número total de documentos e $df(t_i)$ indica o número de

documentos em que o termo aparece. A Tabela 3 apresenta a representação vetorial usando o esquema de pesos TF-IDF.

$$TFIDF(t_i, d) = log(1 + tf(t_i, d)) \times log\left(\frac{|\mathcal{D}| + 1}{df(t_i) + 1}\right) \tag{2.2}$$

Tabela 3 – Exemplo de representação vetorial; os atributos indicam o esquema de pesos TF-IDF (*term frequency - inverse of document frequency*).

	at	*bad*	*comfort*	*great*	*had*	*hotel*	*is*	*rooms*	*service*	*so*	*stay*	*the*	*this*	*time*	*we*	*were*
(1)	0.00	0.00	0.00	0.38	0.00	0.38	0.00	0.38	0.00	0.47	0.00	0.38	0.00	0.00	0.00	0.47
(2)	0.35	0.00	0.00	0.45	0.35	0.23	0.00	0.00	0.00	0.00	0.35	0.00	0.35	0.35	0.35	0.00
(3)	0.00	0.51	0.00	0.00	0.00	0.00	0.51	0.33	0.51	0.00	0.00	0.33	0.00	0.00	0.00	0.00
(4)	0.00	0.00	0.36	0.23	0.00	0.23	0.00	0.23	0.00	0.57	0.00	0.23	0.00	0.00	0.00	0.57

Como pode ser observado, o valor do atributo '*at*' na Tabela 3 é superior em relação ao valor observado na Tabela 2, pois este *token* não aparece em muitos documentos. Analogamente, valores do atributo '*great*' são maiores na Tabela 2, pois o *token* aparece na maioria das mensagens.

2.1 *Stopwords*

Uma técnica que é amplamente utilizada para diminuir o efeito de *tokens* considerados de baixa importância ou significância é a remoção de *stopwords*. Trata-se de palavras que geralmente possuem alta frequência de ocorrência e que na maioria das vezes não agregam informação que ajude na identificação do rótulo do documento. Alguns exemplos dessas palavras, em Inglês, são preposições e artigos, tais como: *a*, *an*, *be*, *but*, *the*, *so*, etc.

O principal objetivo para remoção desses *tokens* é diminuir a dimensionalidade do vocabulário e, consequentemente, acelerar o processo de treinamento dos modelos de classificação. A Tabela 4 apresenta um exemplo de representação vetorial após remover as *stopwords* do vocabulário dos *reviews* apresentados anteriormente.

Tabela 4 – Exemplo de representação vetorial; os atributos indicam o esquema de pesos TF-IDF após remoção de *stopwords*.

bad	*comfort*	*great*	*hotel*	*rooms*	*service*	*stay*	*time*
0.00	0.00	0.58	0.58	0.58	0.00	0.00	0.00
0.00	0.00	0.64	0.32	0.00	0.00	0.50	0.50
0.64	0.00	0.00	0.00	0.41	0.64	0.00	0.00
0.00	0.67	0.43	0.43	0.43	0.00	0.00	0.00

2.2 *Stemming*

Outra técnica comumente utilizada em processamento de linguagem natural é a *stemming*. Trata-se da redução e normalização de palavras para o seu radical. Por exemplo, as palavras *investing* e *investor* são reduzidas para *invest*. Esta técnica é utilizada para não atribuir frequências diferentes para palavras que, na maioria das vezes, possuem o mesmo significado. A Tabela 5 apresenta a representação vetorial após a remoção de *stopwords* e aplicação do *stemming*.

Tabela 5 – Exemplo de representação vetorial; os atributos indicam o esquema de pesos TF-IDF após remoção de *stopwords* e processo de *stemming*.

bad	*comfort*	*gre*	*hotel*	*room*	*serv*	*stay*	*tim*
0.00	0.00	0.58	0.58	0.58	0.00	0.00	0.00
0.00	0.00	0.64	0.32	0.00	0.00	0.50	0.50
0.64	0.00	0.00	0.00	0.41	0.64	0.00	0.00
0.00	0.67	0.43	0.43	0.43	0.00	0.00	0.00

3 Métodos de classificação

De acordo com Crawford et al. (2015), *spam reviews* são mais frequentemente filtrados através da classificação automática com base no conteúdo das mensagens em representação *bag of words*, através de aprendizado supervisionado. Nesta modalidade de aprendizado, o algoritmo tenta aprender através de uma base de dados de treinamento que foi previamente rotulada por um especialista no domínio dos dados em questão. Em seguida, o modelo realiza a predição dos rótulos de amostras desconhecidas, através da generalização realizada com as amostras anteriores. Para isso, utiliza-se comumente métodos tradicionais de aprendizado de máquina, juntamente com duas abordagens de aprendizado diferentes: aprendizado em *batch* (*offline*) e aprendizado incremental (*online*).

No aprendizado em *batch*, os modelos são treinados com a base de dados em sua totalidade, sendo que uma base de dados consiste em um conjunto de amostras X e seus respectivos rótulos y. Após a etapa de treinamento, os modelos não podem mais ser atualizados. Se o classificador sofrer perdas de desempenho ao longo do tempo, com mudanças nas características nas mensagens, por exemplo, será necessário realizar uma nova etapa de treinamento, utilizando uma nova base de dados rotulada e, assim, gerar um novo classificador.

No aprendizado incremental (*online*), os modelos são inicialmente treinados com poucos dados de treinamento. Em seguida, na etapa de predição, é apresentada para o modelo uma amostra por vez. Então, o modelo calcula o erro da predição $\xi = y_{true} - \hat{y}$, onde y_{true} indica o rótulo correto da amostra e $\hat{y}$ indica o rótulo gerado pelo modelo. Se o modelo errar a predição ou o valor de ξ obtido for maior que um determinado limiar, o modelo é treinado de maneira incremental com o rótulo correto da amostra. Esta abordagem de aprendizado é útil para aplicações que precisam manipular grandes quantidades de dados que variam com o tempo, pois o volume de informações tornaria inviável para o modelo ser treinado todas as vezes em que precisasse ser atualizado com dados mais recentes.

Neste capítulo, são brevemente apresentados métodos de classificação com aprendizado *offline* e também métodos com aprendizado *online*. Alguns desses métodos são tradicionais na literatura sobre recuperação de informação, como Naïve Bayes, Rocchio e Máquinas de vetores de suporte (KOTSIANTIS, 2007). A seguir, são brevemente apresentados cada um desses métodos, algumas de suas variações e também métodos estatísticos similares.

3.0.1 Naïve Bayes multinomial

O Naïve Bayes multinomial (M.NB - *multinomial Naïve Bayes*) é um algoritmo de aprendizado probabilístico. Ele calcula a probabilidade de um documento d pertencer à uma determinada classe c pela Equação 3.1 (MANNING et al., 2008), sendo que $P(t_k|c)$ é a probabilidade condicional de um termo t_k ocorrer em um documento da classe c; $P(c)$ é a probabilidade *apriori* de ocorrência de um documento na classe c.

$$P(c|d) \propto P(c) \prod_{1 \leq k \leq |d|} P(t_k|c) \tag{3.1}$$

Após calcular a probabilidade para todas as classes, o modelo escolhe aquela com maior valor (MANNING et al., 2008):

$$y = \arg\max_{c \in \mathcal{C}} P(c|d) \tag{3.2}$$

$$= P(c) \prod_{1 \leq k \leq |d|} P(t_k|c), \tag{3.3}$$

sendo que $P(c)$ e $P(t_k|c)$ são estimados a partir das amostras de treinamento. $P(c)$ corresponde à divisão do total de documentos da classe c ($|\mathcal{D}_c|$) pelo total de documentos de treinamento ($|\mathcal{D}|_{treino}$), como mostrado na Equação 3.4 (MANNING et al., 2008).

$$P(c) = \frac{|\mathcal{D}_c|}{|\mathcal{D}|_{treino}} \tag{3.4}$$

A probabilidade condicional $P(t|c)$ é calculada pela frequência do termo t em documentos da classe c (MANNING et al., 2008):

$$P(t|c) = \frac{T_{ct}}{\sum_{k \in \mathcal{V}} T_{ck}}, \tag{3.5}$$

sendo que T_{ct} representa o total de ocorrências do termo t em documentos da classe c, $\sum_{k \in \mathcal{V}} T_{ck}$ corresponde ao total de ocorrências de todos os termos do vocabulário ($\mathcal{V}$) em documentos da classe c. Para evitar divisão por zero, costuma-se aplicar suavização de Laplace (MANNING et al., 2008):

$$P(t|c) = \frac{T_{ct} + 1}{\sum_{k \in \mathcal{V}} (T_{ck} + 1)} \tag{3.6}$$

$$= \frac{T_{ct} + 1}{\sum_{k \in \mathcal{V}} T_{ck} + |\mathcal{V}|} \tag{3.7}$$

A complexidade da etapa de treinamento é assintoticamente $O(|\mathcal{C}||\mathcal{V}|)$, pois é preciso calcular $|\mathcal{C}|$ probabilidades *apriori* e $|\mathcal{C}||\mathcal{V}|$ probabilidades condicionais (MANNING et al.,

2008). A complexidade da predição de um documento d é $O(|\mathcal{C}||d|)$, onde $|d|$ representa o número de *tokens* do documento; nesta etapa, é preciso recuperar a probabilidade de valor máximo após $|\mathcal{C}||d|$ iterações (MANNING et al., 2008).

3.0.2 Naïve Bayes Bernoulli

O Naïve Bayes Bernoulli (B.NB - *Bernoulli Naïve Bayes*) é uma variante do método Naïve Bayes. Em contraste ao modelo multinomial, o Bernoulli utiliza a informação de não ocorrência de um termo para calcular $P(t|c)$ (MANNING et al., 2008):

$$P(t|c) = \begin{cases} \frac{|\mathcal{D}^t|+1}{|\mathcal{D}_c|+2}, & \text{se o termo } t \text{ ocorre} \\ 1 - \frac{|\mathcal{D}^t|+1}{|\mathcal{D}_c|+2}, & \text{caso contrário,} \end{cases} \tag{3.8}$$

sendo que $|\mathcal{D}^t|$ corresponde ao número de documentos em que o termo t aparece nas amostras de treinamento e $|\mathcal{D}_c|$ indica o número de documentos pertencentes à classe c.

As complexidades de treinamento e predição são equivalentes às do método Naïve Bayes Multinomial (MANNING et al., 2008).

3.0.3 Rocchio

Diferentemente do Naïve Bayes, que utiliza a sequência de termos como representação dos documentos, o Rocchio utiliza a representação espaço-vetorial como principal fundamento para a sua hipótese. Desta forma, a hipótese consiste em: *documentos de uma mesma classe formam uma região contígua e regiões de diferentes classes não se sobrepõem* (MANNING et al., 2008).

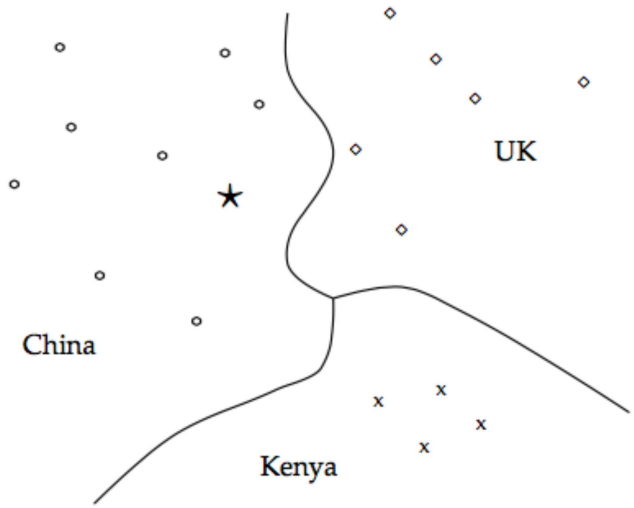
Com base nesta hipótese, para a classificação de texto considera-se que documentos e suas respectivas classes formam regiões distintas, de maneira que seja possível determinar limites e, portanto, separar documentos novos na etapa de predição. Contudo, é necessário empregar a representação adequada para os documentos de texto, pois algumas escolhas podem fazer com que eles não formem regiões contíguas; por exemplo, a remoção de *stopwords* descrita no Capítulo 2 pode acabar por remover atributos que seriam importantes para a correta separação das classes de acordo com suas respectivas regiões (MANNING et al., 2008).

Esta divisão em regiões é demonstrada pela Figura 3, onde os documentos podem ser divididos em três classes, China, Kenya e UK; as linhas representam os limites entre essas classes. A partir disso, métodos de classificação baseados em espaço vetorial objetivam encontrar esses limites entre as classes durante a etapa de treinamento, para que possam atribuir uma classe apropriada para a predição de novas amostras. Neste cenário, o Rocchio divide os documentos de treinamento em diferentes regiões através de centróides. Cada

uma dessas regiões possui como centro o respectivo centróide de uma classe, que é calculado pelo centro de massa de todos os documentos pertencentes a essa classe (MANNING et al., 2008).

Os centróides são calculados pela Equação 3.9, onde $\mathcal{D}_c$ é o conjunto de documentos de treinamento rotulados com a classe c e $\vec{v}(d)$ corresponde ao vetor normalizado para o documento d (MANNING et al., 2008).

Figura 3 – Documentos de texto divididos em regiões distintas de acordo com sua classe.



Fonte: Manning et al. (2008).

$$\vec{\mu}(c) = \frac{1}{|\mathcal{D}_c|} \sum_{d \in \mathcal{D}_c} \vec{v}(d) \tag{3.9}$$

Na etapa de predição, o modelo atribui para a amostra de teste a classe do centróide mais próximo (MANNING et al., 2008), como na Figura 3, onde a amostra de teste simbolizada pela estrela seria classificada como pertencente à classe China. A distância em relação ao centróide é calculada comumente utilizando a distância Euclideana (MANNING et al., 2008).

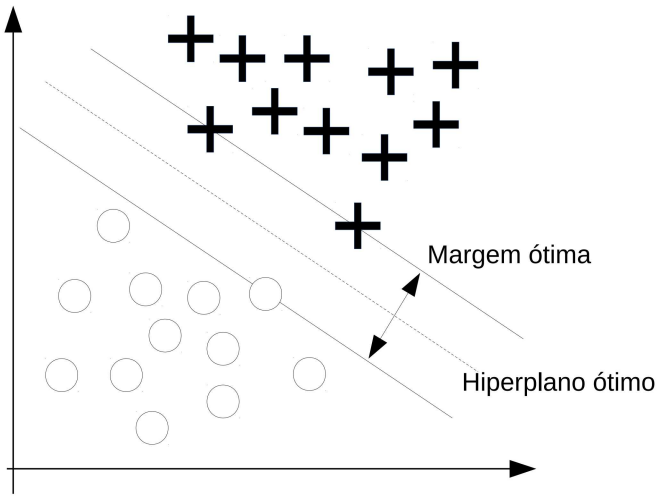
A complexidade da etapa de treinamento é $O(|\mathcal{D}||\bar{d}|+|\mathcal{C}||\mathcal{V}|)$, pois é preciso $O(|\mathcal{D}||\bar{d}|)$ iterações para calcular as somas dos atributos do vetor normalizado, sendo $|\bar{d}|$ o número de atributos médio de um documento d. Utiliza-se $|\bar{d}|$ ao invés de $|\mathcal{V}|$, pois cada documento é um vetor esparso e, portanto, não é necessário percorrer todos os atributos. Para obter os centróides de todas as classes, realizam-se $O(|\mathcal{C}||\mathcal{V}|)$ iterações, pois divide-se cada atributo dos vetores de soma calculados no passo anterior pela quantidade de documentos de cada classe. A complexidade de predição é $O(|\mathcal{C}||d|)$, pois é preciso percorrer todas as classes e

calcular a distância Euclideana entre os centróides e o vetor normalizado do documento d, que possui $|d|$ componentes (MANNING et al., 2008).

3.0.4 Máquinas de vetores de suporte

O método denominado máquinas de vetores de suporte (SVM - *support vector machines*) visa encontrar o hiperplano que, de maneira ótima, separa linearmente amostras de diferentes classes $y \in \{-1, +1\}$ (CORTES; VAPNIK, 1995). Este hiperplano é definido por $\vec{w}\vec{x} + b$, onde $\vec{w}$ corresponde ao vetor de pesos e b representa uma constante de viés (*bias*). O SVM objetiva encontrar um hiperplano que oferece a melhor separação possível entre as classes, como apresentado na Figura 4. Para isso, ele minimiza a expressão definida pela Equação 3.10 (CORTES; VAPNIK, 1995).

Figura 4 – Hiperplano ótimo de separação entre as classes.



$$min \left[\frac{1}{2}\vec{w}^2 + CF_M \left(\sum_{i=1}^{|\mathcal{D}|} \xi_i \right) \right], \tag{3.10}$$

onde $\vec{w}$ representa o vetor de pesos do hiperplano ótimo de separação, C é uma constante suficientemente grande para regularização do aprendizado, $F_M(u)$ indica uma função monotônica convexa, $|\mathcal{D}|$ é o número total de documentos de treinamento e ξ_i representa o erro de classificação para uma amostra de treinamento. De acordo com a Equação 3.10, trata-se de um problema de otimização para minimizar uma função quadrática em $\vec{w}$ (CORTES; VAPNIK, 1995).

Como demonstrado por Cortes e Vapnik (1995), o vetor $\vec{w}$ para o hiperplano ótimo de separação pode ser definido por $\vec{w}_0 = \sum_{i=1}^{|\mathcal{D}|} \hat{\alpha}_i y_i \vec{x}_i$, sendo que y_i corresponde à classe da i-ésima amostra, $\vec{x}_i$ é a i-ésima amostra de treinamento e $\hat{\alpha}$ são multiplicadores de

Lagrange empregados para resolver o problema de otimização mencionado anteriormente. Neste caso, a função a ser minimizada pode ser reescrita por (CORTES; VAPNIK, 1995):

$$min \left[\frac{1}{2} \sum_{i=1}^{|\mathcal{D}|} \sum_{j=1}^{|\mathcal{D}|} \hat{\alpha}_i \hat{\alpha}_j y_i y_j \vec{x}_i . \vec{x}_j + CF_M \left(\sum_{i=1}^{|\mathcal{D}|} \xi_i \right) \right] \tag{3.11}$$

Como pode ser observado, a Equação 3.11 depende do produto escalar $\vec{x}_i.\vec{x}_j$. Desta forma, para cenários onde as amostras são impossíveis de serem separadas linearmente, aplica-se uma transformação Φ_Z de uma amostra $\vec{x}$, para um espaço Z de maior dimensionalidade, de forma que haja uma função $K(\vec{x}_i, \vec{x}_j) = \Phi_Z(\vec{x}_i).\Phi_Z(\vec{x}_j)$, que é denominada função de *kernel* (CORTES; VAPNIK, 1995).

A função de *kernel* é utilizada para projetar os atributos em um novo espaço de maior dimensionalidade e então o SVM encontra o hiperplano ótimo de separação neste novo espaço de atributos expandido (CORTES; VAPNIK, 1995). Por fim, a expressão que demonstra a etapa de treinamento pode ser definida por:

$$min \left[\frac{1}{2} \sum_{i=1}^{|\mathcal{D}|} \sum_{j=1}^{|\mathcal{D}|} \hat{\alpha}_i \hat{\alpha}_j y_i y_j K(\vec{x}_i, \vec{x}_j) + CF_M (\sum_{i=1}^{|\mathcal{D}|} \xi_i) \right] \tag{3.12}$$

A etapa de classificação é dada por (CORTES; VAPNIK, 1995):

$$y(\vec{x}) = sign \left[\sum_{i=1}^{|\vec{V}_{sup}|} \hat{\alpha}_i y_i K(\vec{x}_i, \vec{x}) + b_0 \right], \tag{3.13}$$

onde y corresponde à classe retornada pelo método, sendo que $y \in \{-1, +1\}$; $\vec{x}$ indica a amostra de teste, $|\vec{V}_{sup}|$ corresponde ao número de vetores de suporte utilizados, $\hat{\alpha}$ são multiplicadores de Lagrange, y_i corresponde à classe da amostra que consiste no i-ésimo vetor de suporte, K é a função de *kernel* empregada e b_0 indica a constante de viés do hiperplano ótimo encontrado (CORTES; VAPNIK, 1995).

A complexidade da etapa de treinamento pode variar conforme a técnica utilizada para resolver o problema de otimização. Em geral, ela depende (i) do produto escalar $\vec{x}_i.\vec{x}_j$ ($O(|\mathcal{D}|^2|\vec{x}|)$, sendo $|\mathcal{D}|$ o número de amostras e $|\vec{x}|$ o número de atributos) e (ii) da função de *kernel* empregada. A complexidade da etapa de predição também depende da função de *kernel* empregada e do número de vetores de suporte obtidos ($O(|\vec{V}_{sup}||\vec{x}|)$) (CORTES; VAPNIK, 1995).

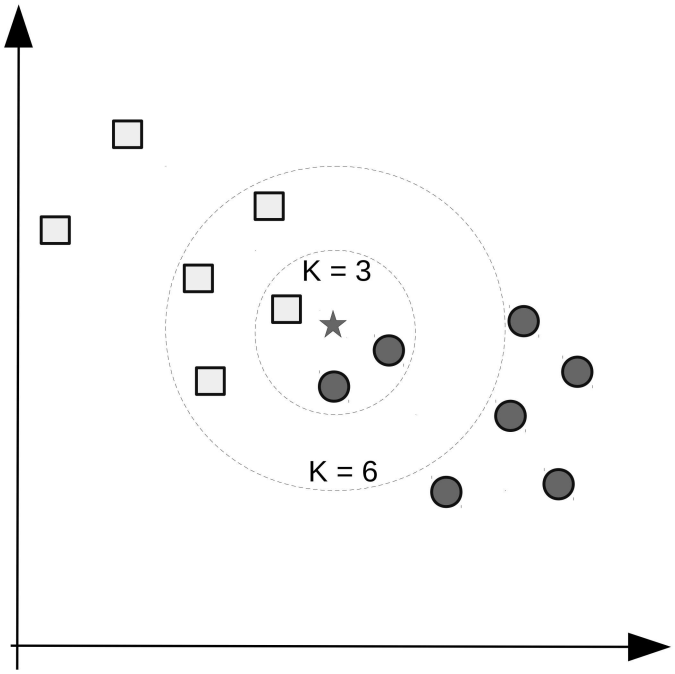
3.0.5 *k*-vizinhos mais próximos

O método k-vizinhos mais próximos (KNN - *k-Nearest Neighbors*) (COVER; HART, 1967; MANNING et al., 2008) determina a classe de um documento de teste através dos

rótulos de documentos de treinamento mais próximos. Por exemplo, para $k = 1$, o método prediz o rótulo de um documento novo através do primeiro documento vizinho mais próximo. Para $k > 1$, o método calcula o rótulo que predomina na maioria dos k vizinhos mais próximos do documento de teste (COVER; HART, 1967).

O cenário apresentado na Figura 5 possui documentos divididos em duas classes (quadrados e círculos) e um documento de teste (estrela). Neste cenário, se a etapa de predição for feita utilizando $k = 3$, o rótulo atribuído para o documento será o de círculo. Por outro lado, se a predição for feita com $k = 6$ o rótulo predominante nos vizinhos mais próximos será o quadrado, que será atribuído ao documento de teste.

Figura 5 – Predição de um documento de teste de acordo com os k vizinhos mais próximos; dependendo da escolha de k, o rótulo escolhido pode variar.



Adicionalmente, é desejável diminuir a probabilidade de ocorrer empates em relação ao número de classes predominante nos k vizinhos mais próximos. Desta forma, é comum utilizar números ímpares para a escolha de k (MANNING et al., 2008).

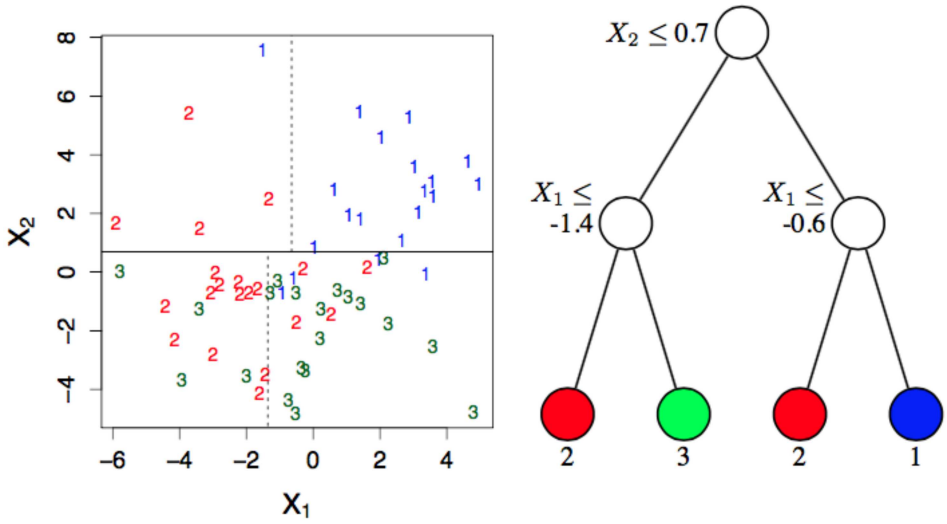
O KNN posterga o treinamento para a etapa de predição e, neste caso, é preciso percorrer todos os documentos e calcular a distância deles em relação ao documento de teste. Assim, a complexidade para a etapa de testes é $O(|\mathcal{D}||\bar{d}|_{treino}|\bar{d}|_{teste})$, onde $|\bar{d}|_{treino}$ é o número médio de termos de cada documento de treinamento e $|\bar{d}|_{teste}$ é o número de termos do documento de teste (COVER; HART, 1967; MANNING et al., 2008).

3.0.6 Árvore de Decisão

Árvore de decisão (DT - *Decision Tree*) é um algoritmo de aprendizado que infere regras simples a partir dos dados de treinamento. Nesta árvore, os nós folhas representam possíveis classes para a predição e os nós restantes representam, individualmente, pontos de particionamento dos dados. Este particionamento é feito com base nas amostras de treinamento e, durante a predição, cada nó intermediário da árvore representa uma "pergunta" sobre a amostra de teste, sendo que cada resposta levará a uma sub-partição específica. Quando não houver mais sub-partições, o nó atual da árvore será um nó folha, que conterá a classe a ser atribuída na predição (BREIMAN et al., 1984).

Um exemplo deste modelo pode ser observado na Figura 6, onde há três classes distintas e todas as amostras foram divididas em quatro partições. Estas partições são delimitadas por valores dos atributos das amostras; por exemplo, há uma partição para amostras da classe "1", que é delimitado por $x_1 = -0.6$ e $x_2 = 0.7$. Após o particionamento, a árvore da direita contém nós com perguntas sobre a amostra de teste, sendo que o nó raiz corresponde à pergunta "$x_2 \leq 0.7$?" e os nós intermediários contém as perguntas "$x_1 \leq -1.4$?" e "$x_1 \leq 0.6$?". Por fim, as folhas da árvore levam a uma das quatro partições, sendo que cada uma se refere, respectivamente, à classe "2", "3", "2" e "1".

Figura 6 – Particionamento de amostras durante a etapa de treinamento (esquerda); árvore de decisão de acordo com o particionamento (direita).



Fonte: Loh (2011).

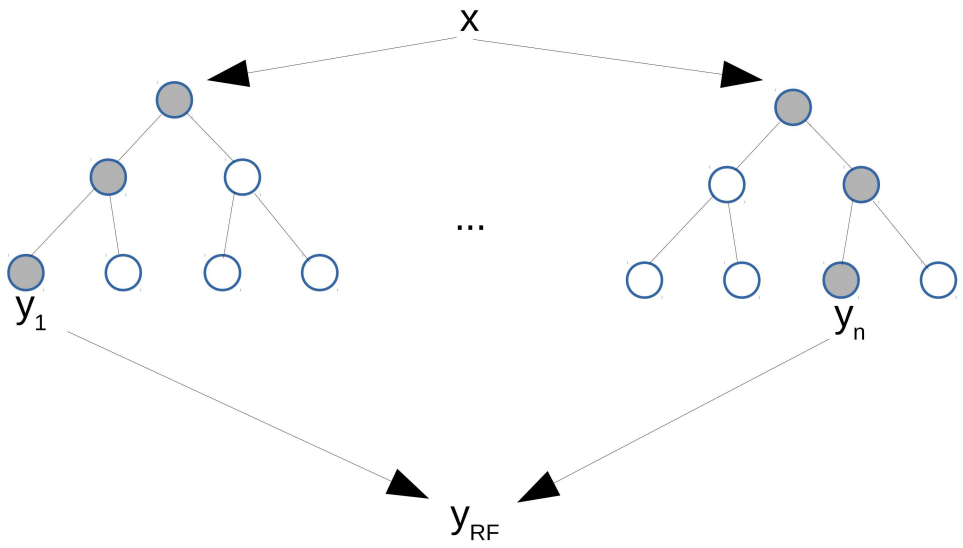
Com base nestes procedimentos, a complexidade para a construção da árvore na etapa de treinamento é de $O(|\bar{d}|_{treino} \times |\mathcal{D}| \times log(|\mathcal{D}|))$, enquanto que a complexidade para a predição de uma amostra de teste é logarítmica em relação ao número de nós da árvore, sendo de $O(log(|\mathcal{D}|))$ (BREIMAN et al., 1984).

3.0.7 Floresta Aleatória

O método Floresta Aleatória (RF - *Random Forest*) consiste em um *ensemble* de árvores de decisão onde é atribuído, para cada árvore, um parâmetro de crescimento aleatório Θ_N. Este parâmetro demonstra a diferença entre as árvores deste método e uma árvore de decisão convencional, já que uma árvore convencional tenta encontrar o particionamento ótimo a partir de amostras de treinamento; as árvores da Floresta Aleatória, porém, realizam os particionamentos até um certo ponto, que é definido pelo parâmetro Θ_N (BREIMAN, 2001).

Como mostrado no exemplo da Figura 7, múltiplas árvores podem constituir a floresta, sendo que cada árvore poderá retornar um resultado diferente para a predição de uma amostra x. Desta forma, todos os resultados são combinados e o rótulo que for predominante na maioria das predições das árvores será o rótulo final retornado pelo método (BREIMAN, 2001).

Figura 7 – Árvores de decisão pertencentes a uma Floresta Aleatória; o rótulo resultante é computado através do rótulo predominante na maioria das árvores.



As complexidades para as etapas de treinamento e predição estão intrinsecamente ligadas ao número $|A|$ de árvores de decisão utilizadas e também à complexidade de cada árvore de maneira individual, que poderá ter diferentes níveis de particionamento, com mais ou menos nós intermediários (BREIMAN, 2001).

3.0.8 MDLText

O MDLText é um método de categorização de texto que suporta múltiplas classes e aprendizado incremental (*online*) (SILVA; ALMEIDA; YAMAKAMI, 2017). Ele é baseado

no princípio MDL, cuja ideia principal é que em um problema onde é necessário escolher entre dois ou mais modelos para se ajustar a um determinado conjunto de dados, deve-se escolher aquele que possui o menor tamanho de descrição (RISSANEN, 1978; RISSANEN, 1983). Isso significa que modelos menos complexos são preferíveis (BARRON; RISSANEN; YU, 1998; GRÜNWALD; MYUNG; PITT, 2005).

Os possíveis modelos avaliados pelo MDLText representam as possíveis classes do problema. Quando um documento de texto d não rotulado é apresentado, ele atribui ao documento a classe que possui o menor tamanho de descrição. Para isso, ele utiliza a seguinte equação:

$$c(d) = argmin_{\forall c} L(d|c_j). \tag{3.14}$$

$L(d|c_j)$ corresponde ao tamanho de descrição do documento d que está sendo classificado, quando é representado pelo modelo relativo à classe c_j. $L(d|c_j)$ é calculado por:

$$L(d|c_j) = \left[\sum_{i=1}^{|d|} L(t_i|c_j) \times \hat{K}(t_i)\right] \times \hat{S}(d, c_j), \tag{3.15}$$

sendo que $L(t_i|c_j)$ é o tamanho de descrição de cada termo quando representado pelo modelo relativo a cada classe; $\hat{K}(t_i)$ é uma penalidade para cada termo do documento baseado em quanto ele pode ser relevante para a identificação da classe e $\hat{S}(d, c_j)$ corresponde a uma penalidade baseada na similaridade entre o documento que está sendo classificado e um protótipo da classe (SILVA; ALMEIDA; YAMAKAMI, 2017).

O tamanho da descrição de cada termo ($L(t_i|c_j)$) é calculado pela Equação 3.16 e é baseado na codificação de Shannon-Fano, conforme explicado em Silva, Almeida e Yamakami (2017).

$$L(t_i|c_j) = \lceil -log_2 \beta(t_i|c_j) \rceil . \tag{3.16}$$

$\beta(t_i|c_j)$ é um peso condicional associado a cada termo, relativo a todos os outros termos que já apareceram em documentos da classe. Ele é calculado por:

$$\beta(t_i|c_j) = \frac{W(t_i|c_j) + \frac{1}{|\Omega|}}{W(c_j) + 1}. \tag{3.17}$$

A função $W(t_i|c_j)$ representa a soma de pesos TF-IDF normalizados do termo t_i, para os documentos da classe c_j. A função $W(c_j)$ representa a soma dos pesos TF-IDF normalizados para todos os termos de todos os documentos da classe c_j. O parâmetro Ω é utilizado para preservar uma porção do tamanho de descrição para termos desconhecidos, não observados durante o treinamento. Segundo Silva, Almeida e Yamakami (2017), o

melhor valor para esse parâmetro pode variar para cada problema e, portanto, uma busca em grade pode ser utilizada para encontrar o valor mais adequado.

Na Equação 3.15, a função $\hat{K}(t_i)$ representa uma penalidade para um determinado termo e seu objetivo é aumentar a capacidade de separação de classes. Segundo Silva, Almeida e Yamakami (2017), para calcular esse valor pode ser usada a técnica de fatores de confidência proposta por Assis et al. (2006).

Na Equação 3.15, a função $\hat{S}(d, c_j)$ representa uma penalidade aplicada de acordo com a dissimilaridade existente entre um documento d e um vetor protótipo da classe c_j. Quanto menor a similaridade entre os dois, maior será essa penalidade. Segundo Silva, Almeida e Yamakami (2017), seu cálculo pode ser realizado utilizando a similaridade de cosseno.

A Equação 3.14, usada para definir a classe de um documento não-rotulado, e todas as demais equações apresentadas são baseadas apenas na frequência dos termos e nos pesos TF-IDF. Portanto, o estágio de treinamento do MDLText consiste apenas na coleta dessas informações a partir dos documentos de treinamento. Consequentemente, esse processo pode ser feito de maneira incremental, o que torna esse método escalável e apto a ser empregado em cenários de classificação *online*, onde novos exemplos podem ser usados para atualizar o modelo de predição previamente treinado. Assim, a complexidade da etapa de treinamento é linear na ordem de $O(|\mathcal{D}||\bar{d}|_{treino})$, onde $|\mathcal{D}|$ indica o número de documentos de treinamento e $|\bar{d}|_{treino}$ indica o número médio de termos de um documento de treinamento. A complexidade da etapa de predição é linear na ordem de $O(|\mathcal{C}||\bar{d}|_{teste})$, onde $|\mathcal{C}|$ corresponde ao número de classes e $|\bar{d}|_{teste}$ indica o número médio de termos do documento de teste (SILVA; ALMEIDA; YAMAKAMI, 2017).

3.0.9 Perceptron

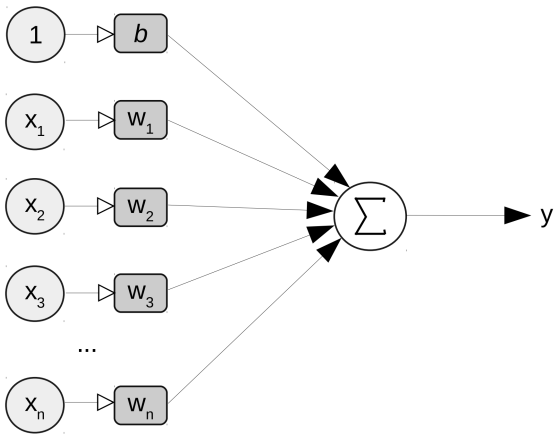
O Perceptron (ROSENBLATT, 1958) é um modelo de classificação binária que procura imitar de maneira bastante simplista o comportamento de um neurônio biológico. Para isso, o modelo recebe um vetor $\vec{x}$ de atributos de entrada ponderados por um vetor $\vec{w}$ de pesos de aprendizado e aplica uma função de ativação, segundo a Equação 3.18:

$$y(\vec{x}) = \vec{w}.\vec{x} + b >= 0, \tag{3.18}$$

onde b é a constante de viés (*bias*); se $y(\vec{x}) \geq 0$ a classe retornada é 1, caso contrário 0. Este modelo também pode ser observado na Figura 8, onde os pesos w precisam ser ajustados na etapa de treinamento.

A etapa de treinamento consiste em apresentar uma amostra por vez. Assim, em cada iteração, se o modelo errar a predição, ele atualiza os pesos $\vec{w}$ incrementalmente utilizando uma taxa de aprendizado l e o erro ξ, que é calculado subtraindo o rótulo

Figura 8 – Funcionamento de um Perceptron.



correto do rótulo esperado. Este cálculo é mostrado na Equação 3.19 (ROSENBLATT, 1958), onde $\vec{w}_j$ corresponde ao j-ésimo componente do vetor de pesos a ser ajustado.

$$\vec{w}_j = \vec{w}_j + l \times \xi \times \vec{x}_j \tag{3.19}$$

Além disso, pode-se atribuir um valor de número de épocas E, que é utilizado para realizar a etapa de treinamento múltiplas vezes até atingir um grau de erro baixo ou nulo, de acordo com os rótulos das amostras (ROSENBLATT, 1958).

Com isso, a etapa de treinamento do Perceptron possui complexidade $O(E|\mathcal{D}||\vec{x}|)$, onde $|\mathcal{D}|$ é o número de amostras e $|\vec{x}|$ é o número de atributos de cada amostra. A complexidade da etapa de predição é $O(|\vec{x}|)$, pois precisa percorrer todos os atributos da amostra de teste (ROSENBLATT, 1958).

3.0.10 Gradiente Descendente Estocástico

O método Gradiente Descendente Estocástico (SGD - *Stochastic Gradient Descent*) (ZHANG, 2004) procura minimizar a função de perda (*loss function*) de um classificador linear no formato $p(\vec{x}) = \vec{w}^T\vec{x}$. O método assume que a qualidade Q do classificador $p(\vec{x})$ é dada por:

$$Q(p(.)) = \mathcal{E}_{X,Y}\phi(p(X), Y), \tag{3.20}$$

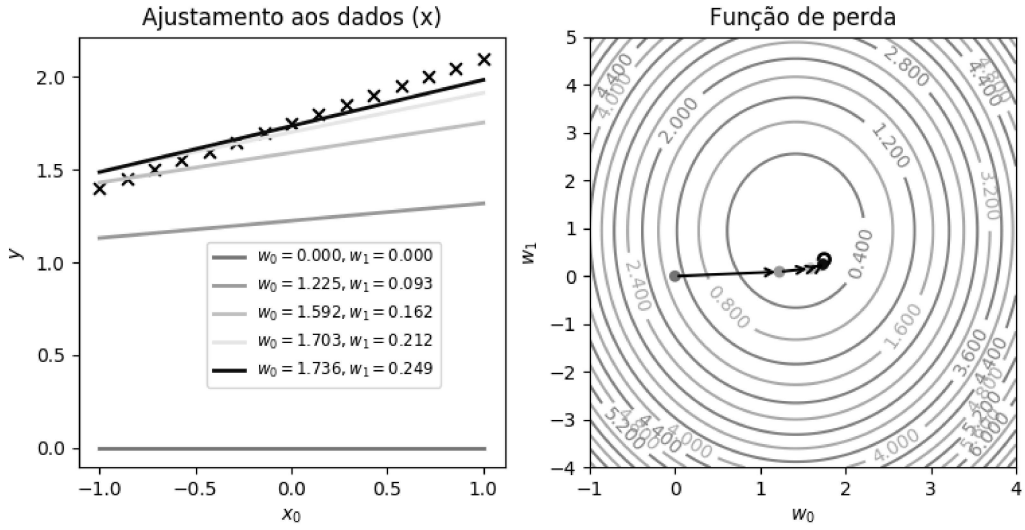
onde $\mathcal{E}_{X,Y}$ corresponde a um valor esperado com relação à matriz de amostras X e à matriz de seus rótulos Y e ϕ corresponde a função de perda de um classificador linear (ZHANG, 2004). Esta equação pode gerar resultados múltiplos ou então não possuir resultados

(ZHANG, 2004). Para isso, utiliza-se um fator de regularização λ, como mostrado na Equação 3.21.

$$Q_\lambda(\vec{w}) = \mathcal{E}_{X,Y}\phi(\vec{w}^T X, Y) + \frac{\lambda}{2}||\vec{w}||_2^2 \tag{3.21}$$

Com base nisso, o método minimiza os erros de um classificador linear utilizando a técnica do gradiente descendente. Um exemplo deste procedimento pode ser observado na Figura 9, onde à esquerda é mostrado um processo de ajustamento de hipóteses de um classificador (com seus pesos w), de acordo com dados de treinamento; à direita é mostrada a sequência de passos do gradiente que, a cada iteração, vai em direção ao ponto mínimo da função de perda.

Figura 9 – Ajustamento da hipótese do classificador (esquerda); gradiente descendente minimizando a função de perda (direita).



De maneira análoga ao Perceptron, o SGD atualiza o vetor de pesos $\vec{w}$ do modelo em cada iteração t de acordo com a Equação 3.22 (ZHANG, 2004):

$$\vec{w}_t = \vec{w}_{t-1} - lS_t^{-1}\frac{\partial}{\partial \vec{w}}L_\phi(\vec{w}_{t-1}, X_t, Y_t), \tag{3.22}$$

sendo que $\vec{w}_t$ corresponde ao vetor de pesos a ser atualizado na iteração atual, $\vec{w}_{t-1}$ é o vetor de pesos da iteração anterior, l indica a taxa de aprendizado (*learning rate*), S corresponde a uma matriz constante para acelerar o processo de convergência e $L_\phi(\vec{w}, \vec{x}, y) = \phi(\vec{w}^T\vec{x}, y) + \frac{\lambda}{2}||\vec{w}||_2^2$. Desta forma, após um número de iterações (ou épocas, de maneira similar ao Perceptron), o modelo normalmente atinge um grau de perda que converge para o mínimo, e o classificador aprende de acordo com as amostras de treinamento (ZHANG, 2004). A

complexidade de treinamento do SGD, portanto, é similar a do Perceptron, pois depende do número de amostras, atributos e épocas (ZHANG, 2004).

3.1 Considerações finais

Neste capítulo, foram brevemente apresentados os principais métodos de classificação disponíveis na literatura e empregados nos experimentos realizados neste trabalho. A Tabela 6 sumariza os tipos de aprendizado suportados por cada método e suas respectivas complexidades para treinamento e predição.

Tabela 6 – Informações sobre os métodos de classificação, tipos de aprendizado e respectivas complexidades para as etapas de treinamento e predição.

Método	Aprendizado	Treinamento	Predição
Rocchio	*offline*	$O(\|\mathcal{D}\|\|\bar{d}\| + \|\mathcal{C}\|\|\mathcal{V}\|)$	$O(\|\mathcal{C}\|\|d\|)$
SVM	*offline*	$O(\|\mathcal{D}\|^2\|\vec{x}\|)$	$O(\|\vec{V}_{sup}\|\|\vec{x}\|)$
KNN	*offline*	$O(1)$	$O(\|\mathcal{D}\|\|\bar{d}\|_{treino}\|\bar{d}\|_{teste})$
DT	*offline*	$O(\|\bar{d}\|_{treino} \times \|\mathcal{D}\| \times log(\|\mathcal{D}\|))$	$O(log(\|\mathcal{D}\|))$
RF	*offline*	$O(\|A\|)$	$O(\|A\|\|)$
M.NB	*online* / *offline*	$O(\|\mathcal{C}\|\|\mathcal{V}\|)$	$O(\|\mathcal{C}\|\|d\|)$
B.NB	*online* / *offline*	$O(\|\mathcal{C}\|\|\mathcal{V}\|)$	$O(\|\mathcal{C}\|\|d\|)$
MDLText	*online* / *offline*	$O(\|\mathcal{D}\|\|\bar{d}\|_{treino})$	$O(\|\mathcal{C}\|\|\bar{d}\|_{teste})$
Perceptron	*online* / *offline*	$O(E\|\mathcal{D}\|\|\vec{x}\|)$	$O(\|\vec{x}\|)$
SGD	*online* / *offline*	$O(E\|\mathcal{D}\|\|\vec{x}\|)$	$O(\|\vec{x}\|)$

4 Avaliação experimental

Objetivando responder as questões de pesquisa que ainda permeiam o tema abordado neste trabalho, este capítulo descreve as etapas seguidas para avaliar o desempenho dos métodos de aprendizado de máquina, quando aplicados na classificação de *reviews* com base no conteúdo das mensagens. Para oferecer uma análise compreensiva, foram realizados experimentos em quatro cenários distintos, usando métodos que oferecem aprendizado *offline* e *online* e coleções de dados reais, públicas e de grande porte, conforme descritos a seguir:

1. *Cenário 1* – métodos com aprendizado *offline* usando conjunto de bases de dados compostas por amostras não ordenadas cronologicamente (atemporais);
2. *Cenário 2* – métodos com aprendizado *offline* usando conjunto de bases de dados compostas por amostras ordenadas cronologicamente (temporais);
3. *Cenário 3* – métodos com aprendizado *online* usando conjunto de bases de dados compostas por amostras não ordenadas cronologicamente (atemporais);
4. *Cenário 4* – métodos com aprendizado *online* usando conjunto de bases de dados compostas por amostras ordenadas cronologicamente (temporais);

4.1 Bases de dados

Para dar credibilidade aos resultados, foram utilizadas duas grandes coleções de dados: *reviews* extraídos do TripAdvisor (OTT et al., 2011) e *reviews* obtidos no Yelp (MUKHERJEE et al., 2013).

A coleção do TripAdvisor (OTT et al., 2011) possui *reviews* legítimos de hotéis de Chicago; os *spam reviews* não são reais e foram escritos por pessoas contratadas *online* pelos autores, através do site *Amazon Mechanical Turk*. A única informação disponibilizada é o conteúdo textual dos *reviews* e sua respectiva classe (*spam* ou legítima), não há outro tipo de informação disponível, como data de criação ou dados do autor.

A coleção do Yelp (MUKHERJEE et al., 2013) possui *reviews* reais de hotéis e restaurantes dos Estados Unidos. Os *reviews* considerados *spam* foram aqueles detectados pelo filtro de *spam* proprietário do Yelp e o restante foi considerado legítimo. Nesta base, há outras informações além do texto dos *reviews*, tais como as datas de publicação.

A partir destas coleções de dados, foram utilizadas apenas as amostras de *reviews* escritos na língua inglesa. Os textos foram inicialmente convertidos para letras minúsculas

e segmentados em múltiplos *tokens*, utilizando como delimitador quaisquer caracteres não alfa-numéricos. A representação utilizada para os *reviews* foi o modelo espaço-vetorial criado por *bag of words*, com N-*grams* como atributos, sendo que o valor de N foi ajustado através de uma busca em grade. Não foram aplicadas as técnicas de *stemming* e remoção de *stopwords*, pois pequenos detalhes na forma de escrita podem ser importantes para a classificação correta das opiniões falsas. O esquema de pesos para representar os *tokens* é detalhado nas seções subsequentes, onde os experimentos são apresentados.

Em seguida, essas coleções foram divididas em bases de dados com (*i*) polaridades positivas (*reviews* positivos ou elogios), (*ii*) negativas (*reviews* negativos ou difamatórios) e (*iii*) com ambas as polaridades. Então, as bases foram agrupadas em dois conjuntos distintos: (*i*) bases sem ordem cronológica de postagem (atemporais) e (*ii*) bases ordenadas por data de postagem (temporais). Essa separação é justificada pela hipótese de que tanto as polaridades quanto a ordem das mensagens podem influenciar no desempenho dos métodos de classificação.

O conjunto de bases atemporais de *reviews* e seus dados são mostrados na Tabela 7. Da mesma forma, informações sobre as bases temporais são mostradas na Tabela 8. Nessas tabelas, a coluna ID corresponde a sigla da base utilizada na descrição dos experimentos e análises subsequentes, $|\mathcal{D}|$ corresponde ao total de amostras, $|\mathcal{V}|$ é o tamanho do vocabulário, $\#Spam$ representa o total de amostras rotuladas como *spam*, $\#Ham$ indica o total de amostras legítimas, $\mathcal{M}$ é a mediana do número de termos por amostra e IQR corresponde a amplitude interquartil (*Interquartile Range*) do número de termos por amostra. Destaca-se que os *reviews* extraídos do TripAdvisor não possuem data de postagem, e portanto, não foram incluídos no conjunto de bases temporais.

Tabela 7 – Informações básicas sobre as bases de dados atemporais. Não foi utilizado nenhum critério de ordenação para as amostras dessas bases.

Base de dados	Polaridade	ID	$\lvert\mathcal{D}\rvert$	$\lvert\mathcal{V}\rvert$	$\#Spam$	$\#Ham$	$\mathcal{M}$	IQR
TripAdvisor - Hotéis	Pos	T-H-P	800	5.548	400	400	71	40
TripAdvisor - Hotéis	Neg	T-H-N	800	7.596	400	400	99	53
TripAdvisor - Hotéis	Pos + Neg	T-H-PN	1.600	9.571	800	800	84	51
Yelp - Hotéis	Pos	Y-H-P	916	7.204	500	416	71	70
Yelp - Hotéis	Neg	Y-H-N	684	7.708	400	284	98	82
Yelp - Hotéis	Pos + Neg	Y-H-PN	1.600	10.673	900	700	80	78
Yelp - Restaurantes	Pos	Y-R-P	1.200	10.358	600	600	69	86
Yelp - Restaurantes	Neg	Y-R-N	1.200	10.146	600	600	83	73
Yelp - Restaurantes	Pos + Neg	Y-R-PN	2.400	14.923	1.200	1.200	77	80
Yelp - Hotéis + Rest.	Pos	Y-HR-P	2.116	13.401	1.100	1.016	70	78
Yelp - Hotéis + Rest.	Neg	Y-HR-N	1.884	13.380	1.000	884	87	76
Yelp - Hotéis + Rest.	Pos + Neg	Y-HR-PN	4.000	19.043	2.100	1.900	78	80

É possível observar que, em geral, o número médio de termos que compõe cada

Tabela 8 – Informações básicas sobre as bases de dados temporais. As amostras estão ordenadas por data de postagem.

Base de dados	Polaridade	ID	$\|\mathcal{D}\|$	$\|\mathcal{V}\|$	$\#Spam$	$\#Ham$	$\mathcal{M}$	IQR
Yelp - Hotéis	Pos	Y-H-P-Ord	500	5.308	250	250	72	75
Yelp - Hotéis	Neg	Y-H-N-Ord	500	6.631	250	250	96	84
Yelp - Hotéis	Pos + Neg	Y-H-PN-Ord	1.000	8.676	500	500	82	79
Yelp - Restaurantes	Pos	Y-R-P-Ord	1.000	9.351	500	500	67	71
Yelp - Restaurantes	Neg	Y-R-N-Ord	1.000	9.588	500	500	79	70
Yelp - Restaurantes	Pos + Neg	Y-R-PN-Ord	2.000	13.773	1.000	1.000	74	71
Yelp - Hotéis + Rest.	Pos	Y-HR-P-Ord	1.500	11.384	750	750	69	73
Yelp - Hotéis + Rest.	Neg	Y-HR-N-Ord	1.500	12.254	750	750	85	75
Yelp - Hotéis + Rest.	Pos + Neg	Y-HR-PN-Ord	3.000	16.926	1.500	1.500	77	76

amostra é muito menor que o número total de termos (tamanho do vocabulário). Isso significa que as amostras são representadas por vetores de alta dimensionalidade, porém muito esparsos, o que certamente pode afetar o desempenho dos métodos de classificação. Outro destaque é para o desbalanceamento no número de amostras por classe que há nas bases atemporais de *reviews* de hotéis do Yelp e também nas bases atemporais de *reviews* de hotéis + restaurantes do Yelp. Essa característica também pode prejudicar o desempenho de métodos, tornando-os tendenciosos à classe majoritária. Além disso, é interessante observar que a quantidade de termos em *reviews* com polaridade negativa é maior que os *reviews* com polaridade positiva. Isso indica que, em média, os usuários tendem a escrever mais quando fazem reclamações do que quando expressam elogios.

4.2 Medidas de desempenho

Para comparar os resultados obtidos pelos métodos, foi adotada como medida de desempenho a tradicional Medida-F (também conhecida como F_1*score*, F-*score* ou F-*measure*). Basicamente, ela calcula a média harmônica entre a precisão e a revocação de um classificador, e seus valores variam entre 0 (pior desempenho) e 1 (melhor desempenho). Os valores da precisão e revocação são baseados nos resultados das predições, que podem ser positivos ou negativos, verdadeiros ou falsos, como mostra a Tabela 9.

Tabela 9 – Tabela de confusão.

	Classe verdadeira	
Resultado da predição	1 (*spam*)	0 (*ham*)
1 (*spam*)	t_p *(true positive)*	f_p *(false positive)*
0 (*ham*)	f_n *(false negative)*	t_n *(true negative)*

O cálculo da precisão (P) é baseado no número de falsos positivos que o modelo de

classificação gerou:

$$P = \frac{t_p}{t_p + f_p}$$

O cálculo da revocação (R) é baseado no total de amostras que o classificador acertou:

$$R = \frac{t_p}{t_p + f_n}$$

A partir dessas definições, a Medida-F (FM) é calculada como mostra a Equação 4.1:

$$FM = 2 \times \frac{P \times R}{P + R} \tag{4.1}$$

As seções a seguir apresentam detalhadamente todos os experimentos realizados considerando os quatro cenários distintos, bem como os desempenhos obtidos por cada método de classificação.

4.3 Experimentos com métodos de aprendizado *offline*

Foram avaliados os seguintes métodos de classificação que oferecem aprendizado *offline* (*batch learning*):

1. Naïve Bayes multinomial (M.NB);
2. Naïve Bayes Bernoulli (B.NB);
3. k-Nearest Neighbours (KNN);
4. Decision Trees (DT);
5. Random Forest (RF);
6. Rocchio;
7. Support Vector Machines (SVM); e
8. MDLText.

Tanto os métodos quanto os experimentos foram implementados usando a linguagem de programação `Python`[1], através de funções disponibilizadas pela biblioteca

[1] *Python*. Disponível em <http://www.python.org>, acessado em 20/04/2017.

`scikit-learn`[2]. Apenas o método MDLText[3] foi avaliado na sua linguagem nativa, a C++.

Para avaliar o desempenho dos métodos, inicialmente foram realizados experimentos com o conjunto de bases de *reviews* atemporais. Posteriormente, foram realizados experimentos com as bases temporais. Uma vez que o desempenho dos métodos SVM, KNN, RF e MDLText pode ser drasticamente afetado pelas escolhas de seus meta-parâmetros, foi realizada uma busca em grade com validação cruzada *k-fold*[4] estratificada, com $k = 5$, para encontrar o melhor valor para os seguintes parâmetros:

- SVM: fator de regularização C;
- KNN: número de vizinhos mais próximos k;
- RF: número de árvores de decisão $|A|$; e
- MDLText: tamanho do vocabulário Ω.

A Tabela 10 apresenta o intervalo de valores percorridos na busca em grade.

Tabela 10 – Intervalo de valores analisados na busca em grade.

Método	Parâmetro	Intervalo
SVM	C	(de 2^{-15} até 2^{15})
KNN	k	(de 5 até 50)
RF	$\|A\|$	(de 10 até 150)
MDLText	Ω	(de 2 até 2^{25})

A função de *kernel* empregada no SVM foi a linear, pois a dimensionalidade do número de atributos para categorização de texto normalmente é muito alta e, portanto, inviabiliza a aplicação prática de funções de *kernels* não-lineares. O esquema de pesos utilizado para representação dos documentos foi o TF-IDF, exceto para o método B.NB, onde o esquema binário de pesos foi escolhido, pois é intrínseco ao método.

Nas subseções a seguir, os resultados de ambos experimentos, com bases de dados atemporais e temporais, são apresentados e analisados.

4.3.1 Cenário 1 – Resultados obtidos com bases de dados atemporais

Neste cenário, foi utilizado o conjunto de bases de dados atemporais (Tabela 7) e os métodos *offline* foram treinados e avaliados usando validação cruzada *5-fold* estratificada,

[2] *Scikit learn*. Disponível em <http://scikit-learn.org/stable/>, acessado em 20/04/2017.
[3] *MDLText*. Disponível em <https://github.com/renatoms88/MDLText>, acessado em 20/04/2017.
[4] Foi utilizado $k = 5$ para separar 20% das bases nas iterações, pois isso também é feito em outras etapas dos experimentos.

ou seja, utilizando 80% das amostras para o treinamento e 20% para o teste, mantendo-se a proporção de classes em cada base.

A Tabela 11 apresenta os resultados obtidos pelos métodos com aprendizado *offline*. Cada valor da tabela representa a média das Medidas-F obtidas e os valores em negrito indicam o melhor resultado para cada base.

Tabela 11 – Resultados obtidos com métodos de aprendizado *offline* e bases de dados atemporais.

ID	B.NB	DT	KNN	MDLText	M.NB	RF	Rocchio	SVM
T-H-P	0,870	0,686	0,849	0,870	0,889	0,861	0,891	**0,899**
T-H-N	0,835	0,662	0,684	0,848	0,760	0,848	0,785	**0,873**
T-H-PN	0,846	0,681	0,761	0,866	0,846	0,848	0,843	**0,878**
Y-H-P	0,626	0,564	0,614	**0,764**	0,568	0,650	0,638	0,648
Y-H-N	0,613	0,529	0,599	**0,701**	0,588	0,604	0,610	0,607
Y-H-PN	0,616	0,557	0,609	**0,750**	0,566	0,637	0,615	0,633
Y-R-P	0,778	0,792	0,802	**0,894**	0,762	0,843	0,828	0,869
Y-R-N	0,595	0,556	0,542	**0,722**	0,551	0,600	0,569	0,583
Y-R-PN	0,680	0,632	0,623	**0,735**	0,566	0,703	0,677	0,687
Y-HR-P	0,681	0,649	0,692	0,719	0,637	0,721	0,719	**0,750**
Y-HR-N	0,564	0,535	0,523	**0,708**	0,535	0,605	0,569	0,577
Y-HR-PN	0,633	0,588	0,583	**0,717**	0,548	0,651	0,632	0,639

Individualmente, o SVM obteve o melhor desempenho nas bases do TripAdvisor, enquanto que o MDLText obteve os melhores resultados para as bases do Yelp. O resultado obtido pelo SVM está em linha com o esperado, pois conforme reportado por Mukherjee et al. (2013), o mesmo apresentou uma significativa piora ao classificar *spam reviews* do mundo real (*reviews* extraídos do Yelp). O MDLText, assim como os demais métodos, também apresentou desempenho inferior quando foram utilizados *reviews* do Yelp. Contudo, essa queda de desempenho não foi tão acentuada se comparada aos outros métodos. Um possível motivo para esse comportamento pode estar relacionado com a característica inerente do MDLText de evitar *overfitting*, pois o modelo mais simples é privilegiado.

O melhor resultado geral foi obtido pelo SVM usando a base de *reviews* de hotéis do TripAdvisor com avaliações de polaridade positiva, sendo $FM = 0,899$. Já, o pior resultado geral foi obtido pelo KNN, para a base com *reviews* de hotéis e restaurantes do Yelp, com polaridade negativa, sendo $FM = 0,523$.

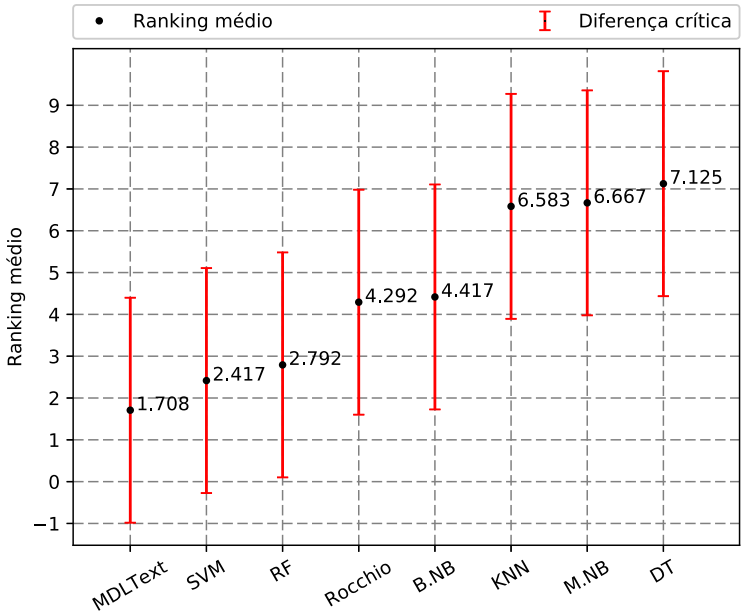
É possível observar que os resultados da detecção de *spam reviews* com avaliações de polaridade positiva foram superiores em relação aos de avaliações negativas; este fato indica que o problema pode ser melhor explorado separadamente, com diferentes modelos para cada polaridade. Como esperado, a junção de *reviews* positivos com negativos resultou

em um desempenho intermediário entre os resultados individuais das polaridades. Em geral, conclui-se também que os métodos obtiveram resultados superiores para as bases de *spam reviews* criados artificialmente (OTT et al., 2011).

Outra característica que pode ser observada é que os resultados com *reviews* de restaurantes do Yelp foram, em geral, superiores aos de *reviews* de hotéis. Como esperado, a união dos *reviews* de hotéis e de restaurantes ocasionou uma leve piora em relação aos resultados obtidos usando apenas *reviews* de restaurantes. Isso indica que, provavelmente, induzir um modelo de predição para cada tipo de estabelecimento/serviço poderá oferecer melhores resultados.

Para certificar que os resultados reportados na Tabela 11 não foram obtidos ao acaso, foi realizada uma análise estatística usando o teste não paramétrico de Friedman, seguindo a metodologia descrita em Demšar (2006). A Figura 10 apresenta o *ranking* médio de cada método avaliado.

Figura 10 – *Rankings* médios usados na análise estatística e diferenças críticas calculadas usando o teste *post-hoc* de Bonferroni–Dunn para o cenário *offline* atemporal.



O teste de Friedman verifica se a hipótese nula, que afirma que os resultados obtidos são estatisticamente equivalentes, pode ser descartada. Para aplicar o teste, foi calculado o valor de χ_F^2 demonstrado pela Equação 4.2, onde N indica o número de bases de dados utilizadas e $\hat{k}$ indica o número de métodos de classificação; $\sum_j \bar{R}_j^2$ indica a soma dos *rankings* médios de cada método.

$$\chi_F^2 = \frac{12N}{\hat{k}(\hat{k}+1)} \left[\sum_j \bar{R}_j^2 - \frac{\hat{k}(\hat{k}+1)^2}{4} \right] \tag{4.2}$$

O valor de χ_F^2 foi então utilizado para calcular o valor de F_F, segundo a Equação 4.3. Em seguida, foi utilizado o valor crítico na distribuição F, para os graus de liberdade $g_1 = (\hat{k} - 1)$ e $g_2 = (\hat{k} - 1)(N - 1)$ (nesta etapa, 7 e 77, respectivamente). Foi obtido um valor de $F_F = 31,106$, enquanto que o valor crítico na distribuição F para um grau de confiança $\alpha = 5\%$ foi de $F_{critico} = 2,13$. Como $F_F > F_{critico}$, a hipótese nula foi descartada, o que indica que houve diferença entre os desempenhos obtidos pelos métodos.

$$F_F = \frac{(N-1)\chi_F^2}{N(\hat{k}-1) - \chi_F^2} \tag{4.3}$$

Na sequência, foi realizado o teste *post-hoc* de Bonferroni-Dunn (DEMŠAR, 2006). Neste caso, todos os métodos são comparados com o que obteve o melhor *ranking* médio, ou seja, com o MDLText. Se a diferença entre os *rankings* for maior do que uma diferença crítica CD, então conclui-se que há diferença estatística entre os resultados. A Equação 4.4 demonstra o cálculo do valor de CD, onde o valor q_α deve ser obtido de tabelas estatísticas para o teste de Bonferroni-Dunn, baseado no número de métodos de classificação utilizados (DEMŠAR, 2006).

$$CD = q_\alpha \sqrt{\frac{\hat{k}(\hat{k}+1)}{6N}} \tag{4.4}$$

Foi obtido um valor calculado de $CD = 2,69$. Com base nos *rankings* médios de cada método e no valor de CD, há evidência estatística de que o método MDLText foi superior aos métodos DT, KNN, B.NB e M.NB. Apesar do *ranking* médio do MDLText ter sido superior ao dos métodos SVM, RF e Rocchio, a diferença entre eles ficou abaixo da diferença crítica CD estipulada pelo teste de Bonferroni-Dunn, sendo eles, portanto, estatisticamente equivalentes.

4.3.2 Cenário 2 – Resultados obtidos com bases de dados temporais

Neste cenário, foi utilizado o conjunto de bases temporais (Tabela 8) e os métodos de classificação foram treinados com os primeiros 80% dos *reviews* postados. Os 20% restantes foram usados para testar o modelo de predição obtido.

O ajuste dos meta-parâmetros também foi realizado usando a busca em grade apresentada na Tabela 10 com as amostras da partição de treinamento. Em seguida, os métodos foram treinados com a mesma partição e testados com a partição de teste.

A Tabela 12 apresenta os resultados obtidos pelos métodos de aprendizado *offline*, avaliados com bases de dados temporais. Os valores em negrito indicam o melhor resultado obtido para cada base de dados.

Tabela 12 – Resultados obtidos com métodos de aprendizado *offline* e bases de dados temporais.

ID	B.NB	DT	KNN	MDLText	M.NB	RF	Rocchio	SVM
Y-H-P-Ord	0,674	0,551	0,692	0,660	0,721	0,702	**0,743**	0,667
Y-H-N-Ord	0,593	0,505	0,495	0,611	**0,679**	0,632	0,673	0,557
Y-H-PN-Ord	0,649	0,599	0,644	0,691	**0,721**	0,636	0,686	0,622
Y-R-P-Ord	**0,786**	0,678	0,700	0,661	0,680	0,728	0,717	0,688
Y-R-N-Ord	0,515	0,537	0,696	0,693	**0,711**	0,639	0,611	0,667
Y-R-PN-Ord	0,689	0,611	0,677	**0,707**	0,704	0,686	0,674	0,697
Y-HR-P-Ord	**0,697**	0,620	0,693	0,657	0,688	0,681	0,694	0,693
Y-HR-N-Ord	0,549	0,592	0,614	0,671	**0,720**	0,572	0,647	0,673
Y-HR-PN-Ord	0,641	0,571	0,621	0,672	**0,724**	0,650	0,672	0,707

De maneira geral, todos os métodos avaliados apresentaram maior dificuldade em classificar dados que variam ao longo do tempo (apresentados em uma ordem cronológica). Devido ao desbalanceamento existente nas bases atemporais de hotéis do Yelp, observou-se uma leve melhora nos resultados com as bases temporais, pois estas não apresentam desbalanceamento entre as classes. Em relação as bases de restaurantes do Yelp, que são balanceadas em ambos os cenários, houve uma piora significativa no desempenho para a polaridade positiva de *reviews*; para polaridades negativas, os desempenhos ficaram levemente melhores em alguns casos e nos outros ficaram com valores próximos aos do cenário atemporal.

Individualmente, o método M.NB obteve os melhores resultados para a maioria das bases de dados. O MDLText obteve uma perda significativa de desempenho neste cenário, possivelmente pelo fator temporal das bases, que indica uma mudança nas características dos *reviews* ao longo do tempo, sem que o modelo de predição pudesse acompanhar essa evolução. O melhor resultado geral foi obtido pelo método B.NB, com *reviews* positivos sobre restaurantes do Yelp, sendo de $FM = 0,786$. O pior resultado geral foi o do método DT, com *reviews* negativos sobre hotéis do Yelp, sendo de $FM = 0,505$.

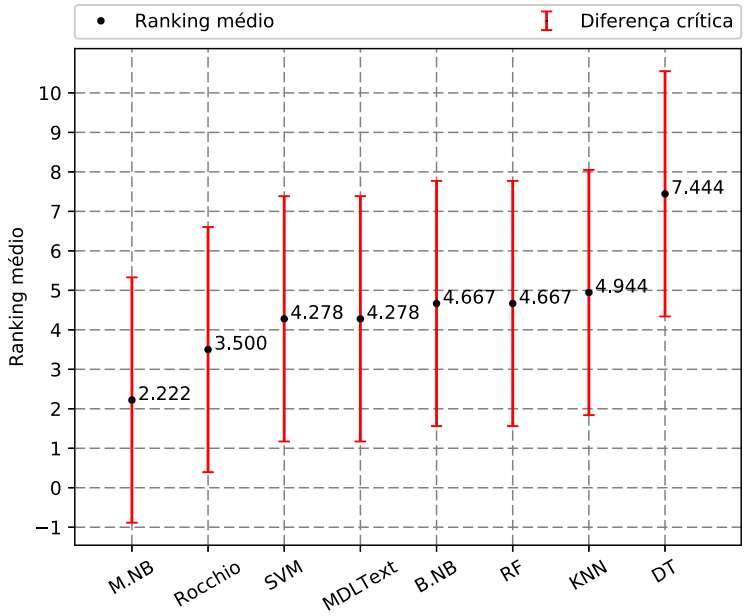
De acordo com os resultados, é possível também constatar as mesmas características encontradas no experimento realizado com bases atemporais, onde os resultados foram inferiores para *reviews* com polaridade negativa; resultados de *reviews* sobre restaurantes também foram, em geral, superiores aos resultados de *reviews* sobre hotéis.

Analisando os resultados dos métodos M.NB e MDLText, observa-se que a junção de *reviews* positivos com negativos ocasionou uma melhora nos resultados dos métodos para as bases temporais. Este fato não ocorreu nos resultados dos outros métodos e também não ocorreu nos experimentos com as bases atemporais.

Para avaliar se há superioridade estatística entre os resultados, também foi realizado

o teste de Friedman (DEMŠAR, 2006). A Figura 11 indica os valores de *rankings* médios obtidos pelos métodos. Dado os graus de liberdade $g_1 = 7$ e $g_2 = 56$, o valor de F_F foi de $4,5419$ e o valor crítico na distribuição F foi de $F_{critico} = 2,1781$, o que indica que a hipótese nula pôde ser descartada.

Figura 11 – *Rankings* médios usados na análise estatística e diferenças críticas calculadas usando o teste *post-hoc* de Bonferroni–Dunn para o cenário *offline* temporal.



Em seguida, foi realizado o teste *post-hoc* de Bonferroni-Dunn e foi obtido um valor de diferença crítica $CD = 3,10$. De acordo com os *rankings* médios obtidos, apenas o método M.NB foi superior ao método DT e não houve evidência estatística o suficiente para concluir que o M.NB foi superior aos demais métodos.

4.4 Experimentos com métodos de aprendizado *online*

Foram avaliados os seguintes métodos tradicionais que oferecem aprendizado *online* (ou incremental):

1. Naïve Bayes multinomial (M.NB);
2. Naïve Bayes Bernoulli (B.NB);
3. Stochastic Gradient Descent (SGD);
4. Perceptron; e
5. MDLText.

Todos os métodos e experimentos foram implementados usando `Python` e funções da biblioteca `scikit-learn`, com exceção do método MDLText que está nativamente implementado na linguagem C++.

Foram realizados experimentos com conjuntos de bases de *reviews* atemporais e temporais. A busca em grade somente foi utilizada para encontrar o melhor meta-parâmetro Ω para o MDLText, pois os demais métodos não possuem meta-parâmetros. O intervalo de valores percorridos na busca foi o mesmo utilizado nos experimentos *offline*, conforme apresentado na Tabela 10. Para a maioria dos métodos, o esquema de pesos utilizado para a representação das amostras de texto foi o TF-IDF; o único método em que foi empregado outro esquema de pesos foi o B.NB, para o qual foi usado o esquema binário, já que este é intrínseco ao método.

Nas subseções a seguir, os resultados de ambos experimentos, com bases de dados atemporais e temporais, são apresentados e analisados.

4.4.1 Cenário 3 – Resultados obtidos com bases de dados atemporais

Neste cenário, os métodos de aprendizado *online* foram avaliados com o conjunto de bases atemporais. Cada base foi dividida de maneira aleatória em *5-fold*, mantendo-se, em cada *fold*, a proporção de classes. Este procedimento é similar a uma validação cruzada *k-fold* estratificada. Porém, neste caso cada *fold* representa 20% da base de dados e foi usado para o treinamento inicial dos métodos, sendo que os 80% restantes foram usados para teste. Na fase de teste, uma amostra foi apresentada por vez ao classificador que, após emitir a predição, recebeu *feedback* caso o resultado tenha sido incorreto. Se a predição fosse incorreta, o classificador atualizaria seu modelo, baseado na classe correta da amostra. Para simular um cenário real, o número de *feedbacks* foi limitado em 40% da base de testes, para caracterizar os casos em que os usuários eventualmente parariam de dar *feedback* para um modelo que frequentemente errasse as predições. Por fim, como demonstra o Algoritmo 1, o resultado consiste na média das Medidas-F de todas as iterações do experimento.

A Tabela 13 apresenta os resultados obtidos pelos métodos com aprendizado *online*, avaliados com o conjunto de bases de dados atemporais. Cada valor da tabela representa a média das Medidas-F obtidas nas cinco rodadas variando a partição de treinamento. Os valores em negrito indicam o melhor resultado para cada base.

Os desempenhos dos métodos B.NB, M.NB e MDLText sofreram uma leve piora se comparados com os resultados obtidos na classificação de *reviews* do TripAdvisor usando aprendizado *offline*. Essa leve piora era esperada, devido ao fator incremental no aprendizado e a expressiva redução no tamanho inicial da base de treinamento. Para as bases do Yelp, o MDLText obteve uma leve piora no desempenho; os métodos B.NB e M.NB, porém, obtiveram melhoras em relação aos do cenário *offline*. Os resultados do

Algoritmo 1 Avaliação de métodos *online* com conjunto de bases atemporais.

Entrada: D (Base de dados atemporal), M (Método de classificação *online*)

$F \leftarrow$ divisão da base de dados D em *5-fold*

para cada *fold* f em F **faça**
  $D_{treino} \leftarrow f$ {20% das amostras de D}
  $D_{teste} \leftarrow F - f$ {80% restante das amostras de D}

  Realiza busca em grade com base de dados D_{treino}
  $p \leftarrow$ melhores parâmetros obtidos pela busca em grade

  Configura método M com parâmetros p
  $modelo \leftarrow$ Treina método M com base D_{treino}

  $\psi_{pred} \leftarrow \phi$ {Conjunto de rótulos das predições}
  $\psi_{true} \leftarrow \phi$ {Conjunto de rótulos dos *reviews*}
  $nf \leftarrow 0$ {Número de *feedbacks*}

  para cada amostra $\vec{x} \in D_{teste}$ **faça**
    $\hat{y} \leftarrow$ rótulo da predição de $\vec{x}$ emitido pelo *modelo*
    $y_{true} \leftarrow$ rótulo correto da amostra $\vec{x}$
    $\psi_{pred} \leftarrow \psi_{pred} \cup \hat{y}$
    $\psi_{true} \leftarrow \psi_{true} \cup y_{true}$

    se $y_{true} \neq \hat{y}$ e $nf < 0.4 \times |D_{teste}|$ **então**
      $nf \leftarrow nf + 1$
      Atualiza *modelo* com $\vec{x}$ e y_{true}
    fim se
  fim para
  Aplica medida de avaliação utilizando ψ_{pred} e ψ_{true}
fim para
Retorna média dos resultados de todos os *fold*

MDLText e M.NB apresentaram valores bem próximos. Porém, os resultados individuais do M.NB foram levemente superiores para a maioria das bases. A capacidade *online* do método M.NB parece ter favorecido a classificação de *spam reviews* do mundo real, sem levar em conta a ordem cronológica das amostras.

Neste experimento, o M.NB obteve o melhor *ranking* médio, como ilustra a Figura 12. O melhor resultado geral foi obtido pelo M.NB com *reviews* positivos de hotéis do TripAdvisor, sendo $FM = 0,856$. O pior resultado geral foi obtido pelo Perceptron em bases de *reviews* negativos de hotéis e restaurantes do Yelp, sendo $FM = 0,570$.

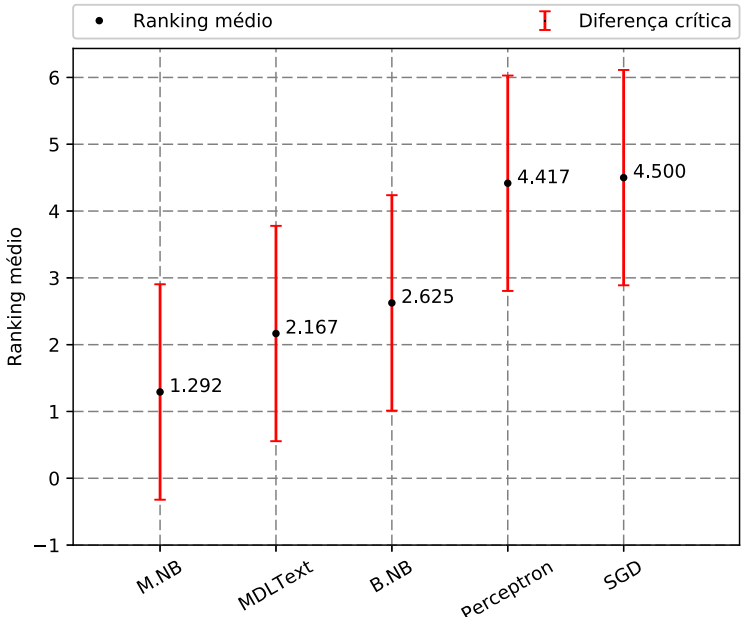
Comparando os resultados deste experimento com os anteriores, é possível observar características semelhantes nos desempenhos dos métodos. Por exemplo, a capacidade de predição em *reviews* positivos foi superior aos de *reviews* negativos. De maneira análoga ao que ocorreu nos experimentos *offline*, os métodos também obtiveram, em geral, resultados

Tabela 13 – Resultados obtidos com métodos de aprendizado *online* e bases de dados atemporais.

ID	B.NB	MDLText	M.NB	Perceptron	SGD
T-H-P	0,848	0,835	**0,856**	0,733	0,740
T-H-N	**0,823**	0,806	0,819	0,659	0,709
T-H-PN	**0,852**	0,825	**0,852**	0,724	0,742
Y-H-P	0,699	**0,718**	0,703	0,619	0,601
Y-H-N	0,644	0,652	**0,662**	0,608	0,607
Y-H-PN	0,674	**0,692**	0,690	0,620	0,616
Y-R-P	0,799	0,836	**0,870**	0,801	0,797
Y-R-N	0,669	0,657	**0,676**	0,576	0,571
Y-R-PN	0,712	0,719	**0,739**	0,665	0,664
Y-HR-P	0,748	0,762	**0,785**	0,695	0,711
Y-HR-N	0,655	0,659	**0,668**	0,570	0,600
Y-HR-PN	0,680	0,706	**0,716**	0,637	0,638

superiores com o conjunto de bases compostas por *spam reviews* criados artificialmente do que com as bases compostas por amostras reais. Observa-se também que *reviews* de restaurantes do Yelp foram mais fáceis de classificar e, portanto, conduziram a resultados melhores do que os *reviews* de hotéis do Yelp.

Figura 12 – *Rankings* médios usados na análise estatística e diferenças críticas calculadas usando o teste *post-hoc* de Bonferroni–Dunn para o cenário *online* atemporal.



Para certificar que os resultados não foram obtidos ao acaso, foi aplicado o teste não paramétrico de Friedman (DEMŠAR, 2006). Nesta etapa, o número de métodos foi

$\hat{k} = 5$ e o número de bases de dados utilizadas foi $N = 12$. Assim, foi obtido um valor de $F_F = 44,2879$, enquanto que o valor crítico na distribuição F, para um grau de confiança $\alpha = 5\%$, grau de liberdade $g_1 = 4$ e $g_2 = 44$, foi de $F_{critico} = 2,13$. Como $F_F > F_{critico}$, a hipótese nula, que afirma que os resultados são estatisticamente equivalentes, foi descartada.

Para a análise *post-hoc*, os desempenhos de todos os métodos foram comparados com o do M.NB, utilizando o teste de Bonferroni-Dunn (DEMŠAR, 2006). Para $\alpha = 0,05$ e $q_\alpha = 2,4980$, foi obtido um valor de $CD = 1,6124$. Com base em CD e nos *rankings* médios de cada método, conclui-se que há evidência estatística de que o M.NB foi estatisticamente superior aos métodos Perceptron e SGD, e equivalente aos demais.

4.4.2 Cenário 4 – Resultados obtidos com bases de dados temporais

Neste cenário, mais desafiador e realístico, os métodos de classificação com aprendizado *online* foram avaliados com o conjunto de bases temporais. Com essas bases, não se pode variar as amostras aleatoriamente em múltiplos *fold*, pois os *reviews* estão ordenados por data de postagem. Desta forma, neste experimento, os primeiros 20% *reviews* postados foram usados no treinamento inicial dos métodos e os 80% restantes para avaliação, com possibilidade de *feedback*. Assim, a base de treinamento é composta por *reviews* mais antigos, enquanto que a base de teste possui amostras mais recentes.

Foi realizada uma busca em grade usando a base de treinamento e os melhores meta-parâmetros encontrados foram utilizados para configurar o método de classificação. Em seguida, o método foi treinado com as amostras da base de treino. Na fase de teste, cada amostra da base de testes foi apresentada ao classificador por vez e o modelo pôde receber *feedback* se a predição foi incorreta. Conforme ilustra o Algoritmo 2, o *feedback* é usado para atualizar o modelo de maneira incremental. O número de *feedbacks* também foi limitado a 40% da base de teste. Por fim, através dos rótulos das predições do modelo e dos rótulos corretos, foi calculada a Medida-F.

A Tabela 14 apresenta os resultados obtidos, sendo que cada valor representa a Medida-F obtida nos experimentos descritos no Algoritmo 2. Os valores em negrito indicam o melhor resultado obtido para cada base de dados.

Comparando os desempenhos obtidos com os do Cenário 2, em que foram usadas as bases atemporais, observa-se que, em geral, o métodos obtiveram uma leve piora para bases positivas de *reviews*; para alguns resultados com bases negativas, houve uma leve melhora. Os métodos B.NB e M.NB apresentaram queda de desempenho em todas as bases de dados. Os métodos MDLText, Perceptron e SGD obtiveram uma leve melhora para as bases que contém hotéis do Yelp, provavelmente devido ao desbalanceamento existente nas bases atemporais de hotéis. De maneira geral, estes resultados comprovam a dificuldade de se classificar *spam reviews* que "evoluem" com o tempo, pois em geral houve

Algoritmo 2 Avaliação de métodos *online* com conjunto de bases temporais.

Entrada: D (Base de dados ordenada por data de postagem), M (Método de classificação *online*)

$D_{treino} \leftarrow$ primeiras 20% amostras de D {*Reviews* mais antigos}
$D_{teste} \leftarrow$ 80% amostras restantes de D {*Reviews* mais recentes}

Realiza busca em grade com base D_{treino} e método M
$p \leftarrow$ melhores parâmetros obtidos pela busca em grade

Configura método M com parâmetros p
$modelo \leftarrow$ Treina método M com base D_{treino}

$\psi_{pred} \leftarrow \phi$ {Conjunto de rótulos das predições}
$\psi_{true} \leftarrow \phi$ {Conjunto de rótulos dos *reviews*}
$nf \leftarrow 0$ {Número de *feedbacks* do classificador}

para cada amostra $\vec{x} \in D_{teste}$ **faça**
    $\hat{y} \leftarrow$ rótulo da predição de $\vec{x}$ emitido pelo *modelo*
    $y_{true} \leftarrow$ rótulo correto da amostra $\vec{x}$
    $\psi_{pred} \leftarrow \psi_{pred} \cup \hat{y}$
    $\psi_{true} \leftarrow \psi_{true} \cup y_{true}$

    se $y_{true} \neq \hat{y}$ e $nf < 0.4 \times |D_{teste}|$ **então**
        $nf \leftarrow nf + 1$
        Atualiza *modelo* com rótulo correto y_{true}
    fim se
fim para
Aplica medida de avaliação utilizando ψ_{pred} e ψ_{true}

Tabela 14 – Resultados obtidos com métodos de aprendizado *online* e bases de dados temporais.

ID	B.NB	MDLText	M.NB	Perceptron	SGD
Y-H-P-Ord	0,673	**0,772**	0,641	0,655	0,689
Y-H-N-Ord	0,578	**0,654**	0,577	**0,654**	0,647
Y-H-PN-Ord	0,628	**0,744**	0,555	0,658	0,665
Y-R-P-Ord	0,784	**0,788**	0,760	0,765	0,767
Y-R-N-Ord	0,634	**0,723**	0,618	0,677	0,653
Y-R-PN-Ord	0,697	**0,752**	0,629	0,698	0,710
Y-HR-P-Ord	0,742	**0,757**	0,721	0,701	0,686
Y-HR-N-Ord	0,595	**0,653**	0,600	0,578	0,614
Y-HR-PN-Ord	0,654	**0,665**	0,662	0,643	0,645

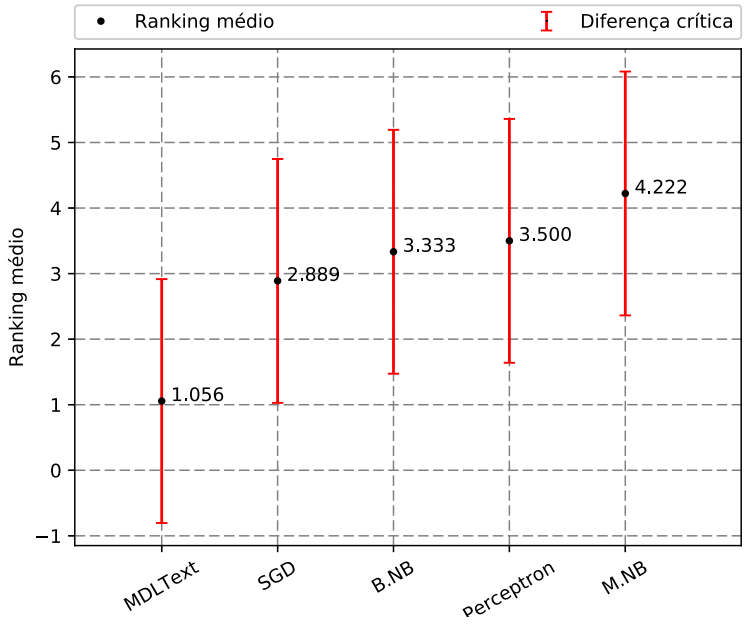
piora no desempenho da maioria dos métodos. O MDLText, apesar de também sofrer com essas quedas de desempenho, apresentou resultados consistentes, não obtendo uma variação muito drástica entre um cenário e outro; possivelmente isso se deve ao fato de que

o MDLText foi criado para possuir uma capacidade grande de generalização dos dados, naturalmente evitando o *overfitting* e minimizando perdas com relação a desbalanceamento de classes.

Neste cenário, o MDLText obteve os melhores resultados em todas as bases de dados e, consequentemente, o melhor *ranking* médio, como ilustrado na Figura 13. O melhor resultado geral obtido foi de $FM = 0,788$, para *reviews* positivos de restaurantes do Yelp. O pior resultado geral foi obtido pelo M.NB, para a base com *reviews* positivos e negativos de hotéis do Yelp, sendo de $FM = 0,555$.

Como observado em etapas anteriores, *reviews* positivos conduziram a resultados superiores aos de *reviews* negativos e a junção das duas polaridades resultou em desempenho intermediário. Adicionalmente, *reviews* sobre restaurantes do Yelp favorecerem a obtenção de resultados superiores aos de hotéis e a junção dos *reviews* desses dois tipos de estabelecimento também gerou resultados intermediários.

Figura 13 – *Rankings* médios usados na análise estatística e diferenças críticas calculadas usando o teste *post-hoc* de Bonferroni–Dunn para o cenário *online* temporal.



Para garantir a consistência estatística dos resultados, foi realizado o teste de Friedman (DEMŠAR, 2006). Para esta etapa, foi obtido um valor de $F_F = 10,3829$, enquanto que o valor crítico na distribuição F, para um grau de confiança $\alpha = 5\%$, grau de liberdade $g_1 = 4$ e $g_2 = 32$, foi de $F_{critico} = 2,6684$. Como $F_F > F_{critico}$, a hipótese nula foi descartada, o que indica que há diferença estatística entre os resultados obtidos.

Em seguida, foi aplicado o teste *post-hoc* de Bonferroni-Dunn (DEMŠAR, 2006), onde todos os métodos foram comparados com o MDLText, que obteve o melhor *ranking* médio. Para $q_\alpha = 2,4980$, foi obtido $CD = 1,8618$. Com base em CD e nos *rankings*

médios de cada método, conclui-se que o método MDLText foi estatisticamente superior aos métodos Perceptron, M.NB e B.NB e equivalente ao SGD.

4.5 Considerações finais

Neste capítulo, foram descritos e apresentados os experimentos realizados para avaliar os métodos de classificação de *spam reviews* com o objetivo de encontrar respostas adequadas para as questões de pesquisa em aberto. Também, foram realizadas análises estatísticas dos resultados para verificar se houve diferença estatística entre os desempenhos dos métodos.

No Cenário 1, utilizando bases atemporais e métodos com aprendizado *offline*, todos os métodos foram melhores com as bases do TripAdvisor e apresentaram quedas de desempenho para *spam reviews* do mundo real. O SVM apresentou os melhores resultados para os *reviews* do TripAdvisor, enquanto que o MDLText obteve os melhores resultados para os *reviews* do Yelp.

No Cenário 2, com bases temporais e aprendizado *offline*, foi observada uma queda geral de desempenho em todos os métodos, devido ao fator cronológico na ordem dos *reviews*. O MDLText apresentou uma queda significativa de desempenho em relação ao Cenário 1, possivelmente por não utilizar aprendizado incremental neste cenário e, consequentemente, não receber atualizações incrementais com *reviews* cujas características sofrem alterações ao longo do tempo. O M.NB obteve o melhor resultado para a maioria das bases de dados.

No Cenário 3, com bases atemporais e aprendizado *online*, os métodos M.NB, MDLText e B.NB sofreram leve piora no desempenho em relação ao cenário *offline* anterior, provavelmente devido ao uso de aprendizado incremental com bases sem ordenação cronológica nesta etapa, onde os métodos foram treinados inicialmente com uma base pequena de *reviews*. Sendo assim, o M.NB apresentou os melhores resultados para a maioria das bases, sendo que o MDLText ficou em segundo lugar, obtendo resultados bem próximos.

No Cenário 4, foram simuladas aplicações do mundo real, com bases temporais e métodos com aprendizado *online*. Neste caso, foi observada uma queda geral de desempenho na maioria dos métodos em relação aos resultados com métodos *offline*. O MDLText apresentou quedas de desempenho menores que as dos demais métodos, obtendo assim os melhores resultados em todas as bases de dados. Isso possivelmente deve-se ao fato de que o método MDLText foi especificamente criado para a tarefa de categorização de texto, sendo que seu funcionamento é baseado no princípio da descrição mais simples, o que garante que modelos mais simples são privilegiados na classificação. Esta característica inerentemente ajuda a evitar o problema de *overfitting*, sem depender exclusivamente de

parâmetros de penalidade de modelos.

Observando todos os cenários, os métodos apresentaram, em geral, quedas de desempenho sob as seguintes condições:

- Bases de dados desbalanceadas;
- Cenários *online*;
- Polaridade negativa de *reviews*;
- Bases com *spam reviews* do mundo real; e
- *Reviews* que evoluem com o tempo.

Com base nos resultados e análises reportados, conclui-se que é necessário investigar o problema no cenário temporal e aprendizado incremental, pois este é o que mais se aproxima de aplicações reais e também o que apresenta maior dificuldade. Além disso, recomenda-se separar o problema por polaridades, utilizando diferentes combinações de atributos para cada polaridade. Da mesma forma, os resultados mostram que o desbalanceamento pode afetar o desempenho dos métodos e, portanto, é necessário utilizar bases reais de *reviews* e, se necessário, aplicar técnicas para balanceá-las por classe.

5 Conclusão

É comum que as pessoas, ao gostarem de um produto ou serviço, os recomendem para outros. Da mesma forma, se um cliente for mal atendido em um hotel ou restaurante, ele provavelmente irá disseminar a reclamação sobre o estabelecimento para seus amigos ou familiares, de modo que ninguém tenha que passar pela mesma experiência ruim. Somando-se a isto, é natural do ser humano procurar por opiniões alheias, sejam recomendações ou reclamações, antes de comprar algum produto, pagar por um serviço ou mesmo escolher um destino de viagem. Assim, opiniões de terceiros são frequentemente importantes e podem afetar a decisão final de consumidores em potencial.

Com a Internet, este hábito de opinar e buscar por opiniões cresceu, sendo que atualmente existem redes sociais específicas para este tipo de atividade. Além disso, os *smartphones* permitem que as pessoas postem opiniões *online* (*reviews*) em tempo real através de sites como TripAdvisor e Yelp. De maneira similar, sites de *e-commerce* como a Amazon oferecem, juntamente com os produtos do catálogo, seus respectivos *reviews* escritos por usuários que os compraram através do site. Devido a essas facilidades, hoje em dia a maioria das pessoas confia tanto em recomendações pessoais quanto em *reviews online*.

Além de beneficiar diretamente consumidores em potencial, os *reviews online* também servem como *feedback* rápido para empresas fabricantes de produtos ou então para proprietários de estabelecimentos como restaurantes, bares ou hotéis. Desta forma, eles podem ajustar seus serviços ou aperfeiçoar seus produtos caso seja necessário, conquistando assim novos clientes. No entanto, reclamações recorrentes sobre um produto ou serviço podem causar um prejuízo enorme para as empresas ou proprietários, que rapidamente poderão perder seus clientes, uma vez que os *reviews* negativos estarão visíveis para qualquer um na Web.

Rapidamente, o ambiente *online* tornou-se um cenário novo de competição para fabricantes de produtos ou serviços, que precisam garantir que sua imagem permaneça positiva nas redes sociais. Além disso, na maioria dos sites, não há um controle rigoroso sobre a autenticidade das informações, sendo que qualquer um pode facilmente publicar informações escondendo a sua identidade ou então criando perfis falsos de usuário. Diante deste cenário, surgiram os primeiros casos de *spam reviews*, nos quais empresas contrataram pessoas para postar *reviews* promovendo seus produtos ou difamando serviços de concorrentes. Este tipo de *spam* é criado manualmente e objetiva controlar a opinião geral sobre um produto ou serviço. Com isso, usuários podem ser facilmente enganados por estas opiniões falsas, assim como empresas podem perder seus clientes rapidamente, devido ao

impacto deste tipo de atividade na Web.

Os *spam reviews*, apesar de relativamente recentes, ganharam a atenção de diversos trabalhos na literatura, sendo que a maioria deles propôs abordagens envolvendo aprendizado de máquina e processamento de linguagem natural para a detecção automática dessas opiniões falsas. Nestas abordagens, os *reviews* são comumente representados por vetores numéricos através de *bag of words*, sendo que as bases de dados utilizadas contém, na maioria dos casos, *spam reviews* não reais e frequentemente criados pelos próprios autores para análise. Isto ocorre devido a dificuldade de se obter bases rotuladas de *spam reviews*, sendo que alguns autores demonstraram que eles são de difícil detecção por humanos, uma vez que são criados manualmente para controlar a opinião geral sobre um objeto específico. Neste cenário, diversos trabalhos reportaram resultados promissores e oferecem como contribuição conjuntos de atributos que visam aumentam a eficácia da classificação.

Apesar do progresso realizado, ainda há uma série de questões relevantes sobre este tema que permanecem em aberto e demandam uma análise criteriosa para serem devidamente respondidas. Essas questões se referem ao desempenho dos métodos de classificação, sendo que não há um consenso na literatura se eles podem ser afetados (*i*) pela polaridade de *reviews*; (*ii*) por *spam reviews* do mundo real; (*iii*) pela utilização de *reviews* sobre múltiplos tipos de serviços/produtos; (*iv*) por mudanças nas características dos *reviews* ao longo do tempo; (*v*) por cenários que demandam aprendizado incremental e (*vi*) pela utilização de bases de dados desbalanceadas. Da mesma forma, também não há um consenso se existe diferença estatística entre os desempenhos dos métodos tradicionais de classificação de *spam reviews* baseados no conteúdo das mensagens.

Diante deste cenário, esta dissertação ofereceu uma análise comparativa entre diversos métodos de aprendizado de máquina, aplicados na classificação de *spam reviews* com base no conteúdo das mensagens. Esta análise foi realizada com o objetivo de oferecer respostas adequadas para os questionamentos levantados. Desta forma, foram realizados experimentos utilizando diferentes bases de dados públicas e disponibilizadas por outros autores, sendo que algumas delas contém *spam reviews* artificiais e outras contém *spam reviews* reais filtrados por algoritmos proprietários. A partir destas bases, os *reviews* foram divididos em duas categorias: (*i*) *reviews* ordenados por data de postagem e (*ii*) *reviews* sem nenhum critério de ordenação. Estes experimentos empregaram também diferentes formas de aprendizado, buscando avaliar os métodos em cenários de aprendizado em *batch* (*offline*) e incremental (*online*). Assim, todos os experimentos realizados foram divididos em quatro cenários distintos:

- *Cenário 1* – Aprendizado *offline* com *reviews* não ordenados (bases atemporais);
- *Cenário 2* – Aprendizado *offline* com *reviews* ordenados por data de postagem (bases temporais);

- *Cenário 3* – Aprendizado *online* com bases atemporais; e
- *Cenário 4* – Aprendizado *online* com bases temporais.

De acordo com os resultados obtidos no Cenário 1, o SVM obteve os melhores resultados para as bases do TripAdvisor, enquanto que o MDLText obteve os melhores resultados para os *reviews* do Yelp. Além disso, observou-se que todos métodos obtiveram resultados superiores com bases de *spam reviews* criados artificialmente por Ott et al. (2011). Essa queda de desempenho com bases que possuem *spam reviews* do mundo real demonstra a dificuldade de classificar *spam reviews* reais, conforme reportado em Mukherjee et al. (2013).

Em seguida, no Cenário 2, os resultados apresentaram uma queda geral de desempenho de todos os métodos, devido ao fator cronológico causado pela ordenação dos *reviews* por data de postagem. Além disso, o método M.NB obteve o melhor resultado para a maioria das bases de dados, enquanto que o método MDLText, que havia obtido resultados melhores no Cenário 1, obteve uma queda significativa de desempenho nesta etapa, provavelmente pela necessidade de utilizar aprendizado incremental em cenários deste tipo, onde os dados sofrem mudanças em seu conteúdo devido ao fator cronológico.

Na sequência, iniciando-se a etapa de aprendizado *online* com o Cenário 3, os resultados gerais mostraram uma leve piora de desempenho em relação aos cenários 1 e 2, com aprendizado *offline*. Esta queda de desempenho era esperada neste experimento devido ao fator incremental no aprendizado, no qual os métodos foram treinados inicialmente com uma base pequena de *reviews*. Neste cenário, o método M.NB apresentou os melhores resultados para a maioria das bases, sendo que o MDLText obteve resultados bem próximos, ficando com o segundo melhor *ranking* médio.

Finalmente, no Cenário 4, os resultados apresentaram queda significativa de desempenho em relação aos cenários com aprendizado *offline*. Porém, o MDLText, apesar de também apresentar quedas de desempenho, apresentou os melhores resultados em todas as bases de dados. Isto possivelmente ocorreu devido à característica inerente do MDLText de evitar o problema de sobreajustamento aos dados, uma vez que o modelo mais simples sempre é privilegiado pelo método na etapa de treinamento.

Além de todos os resultados apresentados em todos os cenários, faz-se necessário revisitar as perguntas propostas no Prefácio desta dissertação, onde as respostas encontradas através dos experimentos descritos foram as seguintes.

Q1: *O desempenho dos métodos pode ser afetado pela polaridade das opiniões (reclamações × elogios)?*

Em todos os cenários, foram observadas quedas de desempenho devido à polaridade negativa de *reviews*. Portanto, recomenda-se o treinamento de diferentes modelos por polaridade de *reviews*.

Q2: *Uma vez que grande parte dos trabalhos da literatura utilizaram bases artificiais de spam, o desempenho dos métodos é preservado na classificação de amostras reais?*

Como observado nos Cenários 1 e 3, todos métodos obtiveram resultados superiores com bases de *spam reviews* criados artificialmente (OTT et al., 2011). Esta queda de desempenho ao usar *spam reviews* do mundo real era esperada, pois um comportamento semelhante já tinha sido previamente observado com o SVM e apresentado por Mukherjee et al. (2013).

Q3: *O desempenho dos métodos pode ser afetado pela diversidade de objetos de análise (treinamento com opiniões de múltiplos tipos de serviço/mercadoria × um único tipo de serviço/mercadoria)?*

Em todos os cenários, foi observado que *reviews* de hotéis do Yelp são mais difíceis de classificar do que os *reviews* de restaurantes do Yelp. Por causa disso, recomenda-se utilizar modelos de classificação e conjuntos de atributos diferentes para cada tipo de estabelecimento ou serviço. Além disso, não houve ganhos de desempenho ao juntar os dois tipos de *reviews* na mesma base de dados, como foi observado em todos os cenários.

Q4: *O desempenho dos métodos pode ser afetado pelas mudanças de características dos reviews por causa do fator cronológico?*

Nos Cenários 2 e 4, foram observadas quedas gerais de desempenho nos métodos devido ao fator cronológico na ordenação das bases de *reviews* por data de postagem.

Q5: *O desempenho dos métodos pode ser afetado em cenários que demandam aprendizado incremental?*

Nos Cenários 3 e 4, os métodos sofreram uma leve piora de desempenho em relação aos cenários com aprendizado *offline*, pois os modelos são inicialmente treinados com uma base pequena de *reviews* e, em seguida, realizam a predição de amostras de teste, podendo receber *feedbacks* e atualizações incrementais caso errem a predição.

Q6: *O desempenho dos métodos pode ser afetado ao se utilizar bases de dados desbalanceadas?*

Ao comparar os resultados do Cenário 2 com os do Cenário 1, foram observadas melhoras sutis de desempenho para *reviews* de hotéis do Yelp, pois estas bases estavam levemente desbalanceadas no Cenário 1. Essa característica também foi observada comparando-se os resultados do Cenário 4 com os do Cenário 3. Portanto,

recomenda-se a aplicação de técnicas para balanceamento de classes em bases de dados, quando necessário.

Q7: *Existe diferença estatística entre os desempenhos dos métodos baseados em conteúdo?*

De maneira geral, não houve um grande vencedor ou método que tenha obtido o melhor resultado consistentemente em todos os cenários avaliados. Contudo, no experimento mais realístico (Cenário 4), os desempenhos dos métodos foram avaliados com bases de *spam reviews* reais, aprendizado incremental, *feedback* limitado e amostras apresentadas cronologicamente. Neste cenário, o MDLText apresentou os melhores resultados em todas as bases de dados, sendo estatisticamente superior aos métodos M.NB, B.NB e Perceptron, e estatisticamente equivalente ao SGD.

Com base em todos os experimentos previamente descritos, não foram realizados testes variando-se o conjunto de atributos das amostras. Assim, para trabalhos futuros é necessário investigar o impacto de diferentes grupos de atributos e técnicas de pré-processamento, como por exemplo *part-of-speech* (OTT et al., 2011), métricas sobre o texto dos *reviews* ou então atributos referentes às características de autores dos *reviews*, tais como métricas sobre a frequência de postagem do usuário. Adicionalmente, é desejável também aumentar o número de bases de dados utilizadas, para garantir maior evidência estatística nas análises *post-hoc*. Alternativamente, uma abordagem envolvendo aprendizado semi-supervisionado pode ser investigada, uma vez que há a dificuldade inerente em obter bases de dados reais e rotuladas para o problema.

Referências

ASSIS, F. et al. Exponential differential document count – a feature selection factor for improving Bayesian filters accuracy. In: *Proceedings of the 2006 MIT Spam Conference (SP'06)*. Cambridge, MA, USA: MIT Press, 2006. p. 1–6. Citado na página 57.

BARRON, A.; RISSANEN, J.; YU, B. The minimum description length principle in coding and modeling. *IEEE Transactions on Information Theory*, IEEE Press, v. 44, n. 6, p. 2743–2760, 1998. Citado na página 56.

BREIMAN, L. Random forests. *Machine learning*, Springer, v. 45, n. 1, p. 5–32, 2001. Citado na página 55.

BREIMAN, L. et al. *Classification and regression trees*. Boca Raton, FL, USA: CRC Press, 1984. Citado na página 54.

BRODER, A. Z. On the resemblance and containment of documents. In: *Proceedings of the 1997 Compression and Complexity of Sequences (SEQUENCES'97)*. Washington, DC, USA: IEEE, 1997. p. 21–29. Citado na página 34.

CORTES, C.; VAPNIK, V. Support-vector networks. *Machine learning*, Springer, v. 20, n. 3, p. 273–297, 1995. Citado 2 vezes nas páginas 51 e 52.

COVER, T.; HART, P. Nearest neighbor pattern classification. *IEEE transactions on information theory*, IEEE, v. 13, n. 1, p. 21–27, 1967. Citado 2 vezes nas páginas 52 e 53.

CRAWFORD, M. et al. Survey of review spam detection using machine learning techniques. *Journal of Big Data*, Springer, v. 42, n. 7, p. 3634–3642, 2015. Citado 2 vezes nas páginas 40 e 47.

DEMŠAR, J. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, JMLR.org, v. 7, n. 7, p. 3634–3642, 2006. Citado 6 vezes nas páginas 67, 68, 70, 73, 74 e 76.

FEI, G. et al. Exploiting burstiness in reviews for review spammer detection. In: *Proceedings of the 7th International Conference on Weblogs and Social Media (ICWSM'13)*. Cambridge, MA, USA: AAAI Press, 2013. p. 175–184. Citado 2 vezes nas páginas 38 e 39.

GANESAN, K.; ZHAI, C. Opinion-based entity ranking. *Information retrieval*, Springer, v. 15, n. 2, p. 116–150, 2012. Citado na página 37.

GRÜNWALD, P. D.; MYUNG, I. J.; PITT, M. A. *Advances in Minimum Description Length: Theory and Applications*. Cambridge, MA, USA: The MIT Press, 2005. Citado na página 56.

HARRIS, C. G. Detecting deceptive opinion spam using human computation. In: *Workshops at the Twenty-Sixth AAAI Conference on Artificial Intelligence (AAAI'12)*. Toronto, Canada: AAAI Press, 2012. p. 87–93. Citado 3 vezes nas páginas 27, 33 e 35.

HEYDARI, A. et al. Detection of review spam: A survey. *Expert Systems with Applications*, Elsevier, v. 42, n. 7, p. 3634–3642, 2015. Citado 3 vezes nas páginas 33, 34 e 39.

JINDAL, N.; LIU, B. Analyzing and detecting review spam. In: *Proceedings of 7th IEEE International Conference on Data Mining (ICDM'07)*. Omaha, NE, USA: IEEE Computer Society, 2007. p. 547–552. Citado na página 34.

JINDAL, N.; LIU, B. Review spam detection. In: *Proceedings of the 16th international conference on World Wide Web (poster paper) (WWW'07)*. Banff, Canada: ACM New York, 2007. p. 1189–1190. Citado 3 vezes nas páginas 34, 38 e 39.

JINDAL, N.; LIU, B. Opinion spam and analysis. In: *Proceedings of the 2008 International Conference on Web Search and Data Mining (WSDM'08)*. Palo Alto, CA, USA: ACM New York, 2008. p. 219–229. Citado na página 34.

JINDAL, N.; LIU, B.; LIM, E.-P. Finding unusual review patterns using unexpected rules. In: *Proceedings of the 19th ACM international conference on Information and knowledge management (CIKM'10)*. Toronto, Canada: ACM New York, 2010. p. 1549–1552. Citado na página 38.

KC, S.; MUKHERJEE, A. On the temporal dynamics of opinion spamming: Case studies on yelp. In: *Proceedings of the 25th International onference on World Wide Web (WWW '16)*. Montréal, Québec, Canada: International World Wide Web Conferences Steering Committee, 2016. p. 369–379. Citado na página 35.

KOTSIANTIS, S. B. Supervised machine learning: A review of classification techniques. In: *Proceedings of the 2007 Conference on Emerging Artificial Intelligence Applications in Computer Engineering: Real Word AI Systems with Applications in eHealth, HCI, Information Retrieval and Pervasive Technologies*. Amsterdam, The Netherlands, The Netherlands: IOS Press Amsterdam, 2007. Citado na página 47.

LI, H. et al. Analyzing and detecting opinion spam on a large-scale dataset via temporal and spatial patterns. In: *Proceedings of the 9th International AAAI Conference on Web and Social Media (ICWSM'15)*. Oxford, England: AAAI Press, 2015. p. 634–637. Citado 2 vezes nas páginas 37 e 40.

LI, H. et al. Bimodal distribution and co-bursting in review spam detection. In: *Proceedings of the 26th International Conference on World Wide Web (WWW'17)*. Perth, Australia: International World Wide Web Conferences Steering Committee, 2017. p. 1063–1072. Citado na página 40.

LOH, W. Classification and regression trees. *Wiley Interdisc. Rew.: Data Mining and Knowledge Discovery*, v. 1, n. 1, p. 14–23, 2011. Citado na página 54.

LUCA, M.; ZERVAS, G. Fake it till you make it: Reputation, competition, and yelp review fraud. *Management Science*, INFORMS, v. 62, n. 12, p. 3412–3427, 2015. Citado 2 vezes nas páginas 32 e 33.

MANNING, C. D. et al. *Introduction to information retrieval*. New York, NY, USA: Cambridge University Press, 2008. Citado 6 vezes nas páginas 48, 49, 50, 51, 52 e 53.

MUKHERJEE, A.; LIU, B.; GLANCE, N. Spotting fake reviewer groups in consumer reviews. In: *Proceedings of the 21st international conference on World Wide Web (WWW'12)*. Lyon, France: ACM New York, 2012. p. 191–200. Citado 3 vezes nas páginas 31, 39 e 40.

MUKHERJEE, A. et al. What yelp fake review filter might be doing? In: *Proceedings of the 7th International Conference on Weblogs and Social Media (ICWSM'13)*. Cambridge, MA, USA: AAAI Press, 2013. p. 409–418. Citado 7 vezes nas páginas 35, 36, 38, 61, 66, 81 e 82.

NAJADA, H. A.; ZHU, X. iSRD: Spam review detection with imbalanced data distributions. In: *Proceedings of the 15th IEEE International Conference on Information Reuse and Integration (IRI'14)*. Redwood City, CA, USA: IEEE Computer Society, 2014. p. 553–560. Citado na página 37.

OTT, M.; CARDIE, C.; HANCOCK, J. T. Negative deceptive opinion spam. In: *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (HLT-NAACL'13)*. Atlanta, Georgia, USA: The Association for Computational Linguistics, 2013. p. 497–501. Citado na página 35.

OTT, M. et al. Finding deceptive opinion spam by any stretch of the imagination. In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (HLT'11)*. Portland, OR, USA: Association for Computational Linguistics, 2011. v. 1, p. 309–319. Citado 9 vezes nas páginas 34, 35, 37, 38, 61, 67, 81, 82 e 83.

PEREZ, S. *Amazon sues more sellers for buying fake reviews*. 2016. <https://techcrunch.com/2016/10/27/amazon-sues-more-sellers-for-buying-fake-reviews/>. Acessado em Dezembro de 2016. Citado na página 32.

QUINLAN, J. R. *C4.5: programs for machine learning*. San Francisco, CA, USA: Elsevier, 2014. Citado na página 37.

RAYANA, S.; AKOGLU, L. Collective opinion spam detection: Bridging review networks and metadata. In: *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'15)*. Sydney, NSW, Australia: ACM New York, 2015. p. 985–994. Citado na página 39.

REPORTER, D. M. *Fired, chef who posted fake TripAdvisor reviews of rival restaurants claiming they were dirty and had poor service*. 2015. <http://www.dailymail.co.uk/news/article-3017290/>. Acessado em Janeiro de 2016. Citado na página 31.

RISSANEN, J. Modeling by shortest data description. *Automatica*, Elsevier, v. 14, n. 5, p. 465–471, 1978. Citado na página 56.

RISSANEN, J. A universal prior for integers and estimation by minimum description length. *The Annals of Statistics*, Institute of Mathematical Statistics, v. 11, n. 2, p. 416–431, jun. 1983. Citado na página 56.

ROSENBLATT, F. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological review*, American Psychological Association, v. 65, n. 6, p. 386, 1958. Citado 2 vezes nas páginas 57 e 58.

SEBASTIANI, F. Machine learning in automated text categorization. *ACM computing surveys (CSUR)*, ACM, v. 34, n. 1, p. 1–47, 2002. Citado 2 vezes nas páginas 43 e 44.

SILVA, R. M.; ALMEIDA, T. A.; YAMAKAMI, A. MDLText: An efficient and lightweight text classifier. *Knowledge-Based Systems*, Elsevier, v. 118, p. 152–164, 2017. Citado 3 vezes nas páginas 55, 56 e 57.

TAYLOR, C.-A. *Chef sacked after putting negative reviews about rivals on TripAdvisor*. 2015. <http://metro.co.uk/2015/03/29/chef-sacked-after-putting-negative-reviews-about-rivals-on-tripadvisor-5125901/>. Acessado em Janeiro de 2016. Citado na página 31.

WILBUR, W. J.; KIM, W. The ineffectiveness of within-document term frequency in text classification. *Information retrieval*, Springer, v. 12, n. 5, p. 509–525, 2009. Citado na página 44.

ZHANG, T. Solving large scale linear prediction problems using stochastic gradient descent algorithms. In: *Proceedings of the twenty-first international conference on Machine learning (ICML'04)*. Banff, Alberta, Canada: ACM New York, 2004. p. 116. Citado 3 vezes nas páginas 58, 59 e 60.