



Ana Júlia de Oliveira Bellini

Análise de Sentimentos em Críticas de Filmes por meio de Algoritmos Clássicos de Aprendizado de Máquina

São José dos Campos

Ana Júlia de Oliveira Bellini

Análise de Sentimentos em Críticas de Filmes por meio de Algoritmos Clássicos de Aprendizado de Máquina

Trabalho de conclusão de curso apresentado ao Instituto de Ciência e Tecnologia – UNIFESP, como parte das atividades para obtenção do título de Bacharela em Ciência da Computação.

Universidade Federal de São Paulo – UNIFESP

Instituto de Ciência e Tecnologia

Bacharelado em Ciência da Computação

Orientadora: Profa. Dr^a. Lilian Berton

São José dos Campos

2019

Na qualidade de titular dos direitos autorais, em consonância com a Lei de direitos autorais nº 9610/98, autorizo a publicação livre e gratuita desse trabalho no Repositório Institucional da UNIFESP ou em outro meio eletrônico da instituição, sem qualquer ressarcimento dos direitos autorais para leitura, impressão e/ou download em meio eletrônico para fins de divulgação intelectual, desde que citada a fonte.

Ficha catalográfica gerada automaticamente com dados fornecidos pelo(a) autor(a)

de Oliveira Bellini, Ana Júlia

Análise de sentimentos em críticas de filmes por meio de algoritmos clássicos de aprendizado de máquina/ Ana Júlia de Oliveira Bellini

Orientação: Lilian Berton-São José dos Campos, 2019.

146 p.

Sentiment analysis on movie reviews by means of classic machine learning algorithms

Trabalho de Conclusão de Curso-Bacharelado em Ciência da Computação-
Universidade Federal de São Paulo-Instituto de Ciência e Tecnologia, 2019.

1. Análise de sentimentos. 2. Críticas de filmes. 3. Aprendizado de máquina. 4.
Processamento de linguagem natural. I. Berton, Lilian

Ana Júlia de Oliveira Bellini

Análise de Sentimentos em Críticas de Filmes por meio de Algoritmos Clássicos de Aprendizado de Máquina

Trabalho de conclusão de curso apresentado ao Instituto de Ciência e Tecnologia – UNIFESP, como parte das atividades para obtenção do título de Bacharela em Ciência da Computação.

Trabalho aprovado em 05 de Dezembro de 2019:

Profa. Dr^a. Lilian Berton
Orientadora

Dr. Igor Mainenti Leal Lopes
Convidado 1

Dr. Helder Luciani Casa Grande
Convidado 2

São José dos Campos
2019

*Este trabalho é dedicado à Ana Rosa, ao Marcos e a todos aqueles que me deram apoio,
enquanto eu dava o sangue durante os últimos quatro anos.*

Agradecimentos

Primeiramente, gostaria de agradecer aos meus pais, por sempre me darem suporte e me ensinarem a importância de dar meu máximo em todo e qualquer aspecto da vida, e ao meu irmão, por sempre dar seu jeito de me botar para cima.

Também deixo meus agradecimentos a todos os meus amigos do ICT-UNIFESP, que sempre estiveram lá para tornar não somente minha jornada pelo ensino superior mais leve, mas a minha vida, também, desde o primeiro dia de curso. Saibam que sempre estarei aqui por vocês, para os altos e baixos da caminhada de cada um.

E, por fim, agradeço à minha orientadora, Profa. Dr^a. Lilian Berton, por aceitar ser a pessoa que daria o norte para meu TCC, e por ser uma orientadora tão presente ao longo deste ano de trabalho.

*“Não há nada mais enganador
do que um fato óbvio.”
(Sir Arthur Conan Doyle)*

Resumo

A análise de sentimentos envolve o tratamento computacional de opiniões, emoções e subjetividade presentes em um texto, sendo objeto de estudo da área de processamento de linguagem natural. Atualmente, diversos espectadores, quando desejam assistir a algum filme em sua casa ou no cinema, recorrem às redes sociais ou a outros *sites* especializados, em busca de críticas sobre uma determinada obra, onde estas opiniões possuem forte influência sobre a decisão do usuário de assistir, ou não, a um determinado filme. Tais opiniões vêm sendo cada vez mais utilizadas na obtenção de *feedback* diretamente do público-alvo durante os últimos anos, com a aplicação da análise de sentimentos. Portanto, o objetivo deste trabalho é realizar um estudo aprofundado de técnicas de processamento de texto e aprendizado de máquina, empregadas em análises de sentimentos vistas na literatura, de forma a analisar quais são as mais adequadas para a tarefa de extrair emoções de críticas de filmes.

Palavras-chaves: análise de sentimentos. críticas de filmes. aprendizado de máquina. processamento de linguagem natural.

Abstract

Sentiment analysis involves the computational treatment of opinions, emotions and subjectivity present in a text, being a study object of the area of natural language processing. Nowadays, several audience members, when they wish to watch a movie in their homes or at the cinema, turn to a social network or other specialized websites in search of reviews about a certain film, where these opinions have a strong influence on the user's decision to watch, or not, a particular movie. Such opinions have been increasingly used to obtain feedback directly from the target audience during the last few years, with the application of sentiment analysis. Therefore, the objective of this work is to carry out an in-depth study of techniques of text processing and machine learning, used in sentiment analysis seen in the literature, in order to assess which are the most suitable for the task of extracting emotions from film reviews.

Keywords: sentiment analysis. movie reviews. machine learning. natural language processing.

Lista de ilustrações

Figura 1 – Poema “No Meio do Caminho”, por Carlos Drummond de Andrade	31
Figura 2 – Exemplo de conversão para caixa baixa e remoção de pontuações	32
Figura 3 – Exemplo de remoção de <i>stop words</i>	33
Figura 4 – Exemplo de lematização	33
Figura 5 – Exemplo de <i>stemming</i>	34
Figura 6 – Arquitetura do modelo <i>Continuous Bag-of-Words Model</i> (CBOW)	38
Figura 7 – Arquitetura do modelo <i>Continuous Skip-gram Model</i>	40
Figura 8 – Ilustração do problema de <i>overfitting</i>	45
Figura 9 – Ilustração do problema de <i>underfitting</i>	45
Figura 10 – SVMs lineares	54
Figura 11 – SVMs não-lineares	55
Figura 12 – Ilustração de Árvores de Decisão	58
Figura 13 – Ilustração do funcionamento do <i>Random Forest</i>	60
Figura 14 – Ilustração do algoritmo <i>k-Nearest Neighbours</i> (k-NN)	61
Figura 15 – Estrutura de um neurônio natural	63
Figura 16 – Arquitetura de um neurônio artificial	64
Figura 17 – Exemplo de rede neural com várias camadas	66
Figura 18 – Arquitetura de uma RNA <i>feedforward</i> de camada única	67
Figura 19 – Arquitetura de uma RNA <i>feedforward</i> multicamadas	68
Figura 20 – Arquitetura de uma RNA recorrente	69
Figura 21 – <i>Multilayer Perceptron</i> com sinais envolvidos na retropropagação	75
Figura 22 – Fluxo de trabalho proposto	93
Figura 23 – <i>Pipelines</i> de processamento de textos	94
Figura 24 – Modelos propostos para análise de sentimentos	96
Figura 25 – Distribuição de classes do <i>dataset</i> do <i>site Rotten Tomatoes</i>	98
Figura 26 – Nuvem de palavras do conjunto de dados do <i>site Rotten Tomatoes</i>	98
Figura 27 – Frequência dos termos contidos no <i>dataset</i> do <i>site Rotten Tomatoes</i>	99
Figura 28 – Distribuição de classes do <i>dataset</i> do <i>site IMDb</i>	100
Figura 29 – Nuvem de palavras do conjunto de dados do <i>site IMDb</i>	100
Figura 30 – Frequência dos termos contidos em um subconjunto do <i>dataset</i> do <i>site IMDb</i>	101
Figura 31 – Estratégia utilizada para divisão dos <i>datasets</i> originais	103

Lista de tabelas

Tabela 1 – Exemplo de <i>Part-of-Speech Tagging</i>	35
Tabela 2 – Exemplo de <i>Bag-of-Words</i>	35
Tabela 3 – Grupos de algoritmos de treinamento de RNAs	74
Tabela 4 – Exemplo de matriz de confusão para um problema de 4 classes	81
Tabela 5 – Matriz de confusão para classificação binária	82
Tabela 6 – Informações sobre as publicações	88
Tabela 7 – Síntese das abordagens utilizadas em cada publicação	89
Tabela 8 – Experimentos realizados com <i>Naïve Bayes</i> , utilizando o <i>dataset</i> do site IMDb	104
Tabela 9 – Matrizes de confusão dos experimentos com <i>Naïve Bayes</i> , sobre o <i>dataset</i> do site IMDb	105
Tabela 10 – Experimentos realizados com <i>Naïve Bayes</i> , utilizando o <i>dataset</i> do site <i>Rotten Tomatoes</i>	105
Tabela 11 – Matrizes de confusão dos experimentos do algoritmo <i>Naïve Bayes</i> , sobre o <i>dataset</i> do site <i>Rotten Tomatoes</i>	105
Tabela 12 – Primeiros experimentos realizados com <i>Support Vector Machine</i> , utilizando o <i>dataset</i> do site IMDb e função <i>kernel</i> radial	106
Tabela 13 – Matrizes de confusão dos primeiros experimentos com <i>Support Vector Machine</i> , sobre o <i>dataset</i> do site IMDb, utilizando função <i>kernel</i> radial	106
Tabela 14 – Experimentos realizados com <i>Support Vector Machine</i> , utilizando o <i>dataset</i> do site IMDb	107
Tabela 15 – Matrizes de confusão dos experimentos com <i>Support Vector Machine</i> , sobre o <i>dataset</i> do site IMDb	107
Tabela 16 – Experimentos realizados com <i>Support Vector Machine</i> , utilizando o <i>dataset</i> do site <i>Rotten Tomatoes</i>	107
Tabela 17 – Matrizes de confusão dos experimentos com <i>Support Vector Machine</i> , sobre o <i>dataset</i> do site <i>Rotten Tomatoes</i>	108
Tabela 18 – Experimentos realizados com Árvore de Decisão, utilizando o <i>dataset</i> do site IMDb	108
Tabela 19 – Matrizes de confusão dos experimentos com Árvore de Decisão, sobre o <i>dataset</i> do site IMDb	109
Tabela 20 – Experimentos realizados com Árvore de Decisão, utilizando o <i>dataset</i> do site <i>Rotten Tomatoes</i>	109
Tabela 21 – Matrizes de confusão dos experimentos com Árvore de Decisão, sobre o <i>dataset</i> do site <i>Rotten Tomatoes</i>	109

Tabela 22 – Experimentos realizados com <i>Random Forest</i> , utilizando o <i>dataset</i> do site IMDb	110
Tabela 23 – Matrizes de confusão dos experimentos com <i>Random Forest</i> , sobre o <i>dataset</i> do site IMDb	110
Tabela 24 – Experimentos realizados com <i>Random Forest</i> , utilizando o <i>dataset</i> do site <i>Rotten Tomatoes</i>	110
Tabela 25 – Matrizes de confusão dos experimentos com <i>Random Forest</i> , sobre o <i>dataset</i> do site <i>Rotten Tomatoes</i>	111
Tabela 26 – Experimentos realizados com <i>k-Nearest Neighbours</i> , utilizando o <i>dataset</i> do site IMDb	111
Tabela 27 – Matrizes de confusão dos experimentos com <i>k-Nearest Neighbours</i> , sobre o <i>dataset</i> do site IMDb	112
Tabela 28 – Experimentos realizados com <i>k-Nearest Neighbours</i> , utilizando o <i>dataset</i> do site <i>Rotten Tomatoes</i>	112
Tabela 29 – Matrizes de confusão dos experimentos com <i>k-Nearest Neighbours</i> , sobre o <i>dataset</i> do site <i>Rotten Tomatoes</i>	112
Tabela 30 – Experimentos realizados com <i>Multilayer Perceptron</i> , utilizando o <i>dataset</i> do site IMDb	113
Tabela 31 – Matrizes de confusão dos experimentos com <i>Multilayer Perceptron</i> , sobre o <i>dataset</i> do site IMDb	113
Tabela 32 – Experimentos realizados com <i>Multilayer Perceptron</i> , utilizando o <i>dataset</i> do site <i>Rotten Tomatoes</i>	113
Tabela 33 – Matrizes de confusão dos experimentos com <i>Multilayer Perceptron</i> , sobre o <i>dataset</i> do site <i>Rotten Tomatoes</i>	114

Lista de abreviaturas e siglas

AM	Aprendizado de Máquina
BESAHOT	<i>Bewertung Saarländischer Hotels im Web</i>
BoW	<i>Bag-of-Words</i>
CART	<i>Classification and Regression Tree</i>
CBOW	<i>Continuous Bag-of-Words Model</i>
CSV	<i>Comma-Separated Values</i>
F_N	<i>False Negative</i>
F_P	<i>False Positive</i>
ID3	<i>Iterative Dichotomiser 3</i>
IDF	<i>Inverse Document Frequency</i>
IMDb	<i>Internet Movie Database</i>
k-NN	<i>k-Nearest Neighbours</i>
LSTM	<i>Long Short-Term Memory</i>
MAP	<i>Maximum A Posteriori</i>
MaxEnt	<i>Maximum Entropy</i>
MLP	<i>Multilayer Perceptron</i>
N/A	<i>Not Available</i>
NCE	<i>Noise Contrastive Estimation</i>
NLTK	<i>Natural Language Toolkit</i>
NNLM	<i>Neural Net Language Model</i>
PLN	Processamento de Linguagem Natural
POS Tagging	<i>Part-of-Speech Tagging</i>
RBF	<i>Radial Basis Function</i>

RNA	Rede Neural Artificial
RSLP	Removedor de Sufixos da Língua Portuguesa
SMART	<i>System for the Mechanical Analysis and Retrieval of Text</i>
SMO	<i>Sequential Minimal Optimisation</i>
SV	<i>Support Vector</i>
SVM	<i>Support Vector Machine</i>
TF	<i>Term Frequency</i>
TF-IDF	<i>Term Frequency-Inverse Document Frequency</i>
T_N	<i>True Negative</i>
T_P	<i>True Positive</i>
TSV	<i>Tab-Separated Values</i>
WEKA	<i>Waikato Environment for Knowledge Analysis</i>

Lista de símbolos

\Im	Imaginário, utilizado como espaço de características da SVM
Δ	Letra grega Delta, representando ajustes de peso em redes neurais
α	Letra grega minúscula alpha
δ	Letra grega minúscula delta, para cálculo de gradiente local da MLP, e também como um parâmetro de função <i>kernel</i> da SVM
η	Letra grega minúscula eta, indicando taxa de aprendizado de redes neurais
κ	Letra grega minúscula kappa, um parâmetro de função <i>kernel</i> da SVM
φ	Letra grega minúscula phi, denotando funções de ativação de redes neurais
ϱ ou ρ	Letra grega minúscula rho, representando uma função de uma margem de separação de uma SVM, ou um limiar para esta margem, respectivamente
σ	Letra grega minúscula sigma, representando um parâmetro de função <i>kernel</i> da SVM
θ	Letra grega minúscula theta, sendo uma constante usada em cálculos de risco em uma SVM
ξ	Letra grega minúscula xi, denotando uma folga aplicada sobre margens de uma SVM
Φ	Letra grega phi, denotando uma função <i>kernel</i> da SVM
$ A $	Módulo de A
$\ A\ $	Norma de A
\forall	Para todo
\in	Pertence
\prod_a^b	Produtório de a até b
\propto	Proporcional
\sum	Somatório
\sum_a^b	Somatório de a até b
\top	Transposta

Sumário

1	Introdução	27
2	Fundamentação teórica	29
2.1	Análise de sentimentos	29
2.1.1	Definição	29
2.1.2	Contexto histórico	29
2.1.3	Aplicações	30
2.1.4	Desafios	30
2.2	Processamento de textos	31
2.2.1	Pré-processamento	31
2.2.1.1	Conversão para caixa baixa e remoção de pontuações	32
2.2.1.2	Remoção de <i>stop words</i>	32
2.2.1.3	Lematização	33
2.2.1.4	<i>Stemming</i>	34
2.2.1.5	<i>Part-of-Speech Tagging</i>	34
2.2.2	<i>Bag-of-Words</i>	35
2.2.3	Geração do vetor de atributos	35
2.2.3.1	<i>Term Frequency (TF)</i>	35
2.2.3.2	<i>Term Frequency-Inverse Document Frequency (TF-IDF)</i>	36
2.2.4	<i>Embedding</i> via redes neurais	36
2.2.4.1	Skip-gram	37
2.2.4.2	Word2vec	40
2.3	Aprendizado de máquina	42
2.3.1	Conceitos e definições	42
2.3.1.1	Exemplo	43
2.3.1.2	Atributo	43
2.3.1.3	Classe	43
2.3.1.4	Conjunto de exemplos	43
2.3.1.5	Classificador	43
2.3.1.6	Viés	44
2.3.1.7	Distribuição e prevalência de classes	44
2.3.1.8	<i>Underfitting</i> e <i>overfitting</i>	44
2.3.1.9	Ruído	46
2.3.1.10	Poda	46
2.3.2	<i>Naïve Bayes</i>	46

2.3.3	<i>Support Vector Machine (SVM)</i>	47
2.3.3.1	SVMs lineares	49
2.3.3.2	SVMs não-lineares	54
2.3.3.3	Vantagens e desvantagens	57
2.3.4	Árvores de Decisão e <i>Random Forest</i>	57
2.3.4.1	Árvores de Decisão	57
2.3.4.2	Métricas para regras de divisão	58
2.3.4.3	<i>Random Forest</i>	60
2.3.4.4	Vantagens e desvantagens	61
2.3.5	<i>k-Nearest Neighbours (k-NN)</i>	61
2.4	Redes neurais	62
2.4.1	<i>Multilayer Perceptron (MLP)</i>	74
2.4.2	Vantagens e desvantagens das RNAs	80
2.5	Avaliação dos algoritmos de classificação	80
2.5.1	<i>k-Fold cross validation</i>	80
2.5.2	Matriz de confusão	81
2.5.3	Acurácia	82
2.5.4	Erro	83
2.5.5	<i>Precision</i>	83
2.5.6	<i>Recall</i>	83
2.5.7	<i>F-Score</i>	83
3	Revisão da literatura	85
3.1	Críticas de filmes	85
3.2	Outros domínios	86
3.3	Síntese da revisão	87
3.4	Considerações	90
4	Proposta do trabalho	91
4.1	Objetivos	91
4.2	Metodologia	91
4.2.1	Bibliotecas utilizadas	91
4.2.2	<i>Datasets</i>	92
4.2.3	Fluxo de trabalho	92
4.2.3.1	Processamento de texto	93
4.2.3.2	Classificação	95
4.2.3.3	Validação	95
4.2.4	Tipos de modelos para classificação	96
5	Resultados	97

5.1	Análise exploratória dos conjuntos de dados	97
5.2	Configuração dos experimentos realizados	101
5.3	Resultados dos experimentos	103
5.3.1	<i>Naïve Bayes</i>	104
5.3.2	<i>Support Vector Machine</i>	105
5.3.3	Árvore de Decisão	108
5.3.4	<i>Random Forest</i>	109
5.3.5	<i>k-Nearest Neighbours</i>	111
5.3.6	<i>Multilayer Perceptron</i>	112
5.4	Discussão dos resultados obtidos	114
6	Considerações finais	117
	Referências	119
	Apêndices	131
	APÊNDICE A Experimentos completos com IMDb	133
A.1	<i>Naïve Bayes</i>	133
A.2	<i>Support Vector Machine</i>	134
A.2.1	Classificador linear	134
A.2.2	Classificador não linear, com função <i>kernel</i> radial	135
A.3	Árvore de Decisão	136
A.4	<i>Random Forest</i>	137
A.5	<i>k-Nearest Neighbours</i>	138
A.6	<i>Multilayer Perceptron</i>	139
	APÊNDICE B Experimentos completos com <i>Rotten Tomatoes</i>	141
B.1	<i>Naïve Bayes</i>	141
B.2	<i>Support Vector Machine</i>	142
B.3	Árvore de Decisão	143
B.4	<i>Random Forest</i>	144
B.5	<i>k-Nearest Neighbours</i>	145
B.6	<i>Multilayer Perceptron</i>	146

1 Introdução

Ao longo dos últimos anos, houve um grande crescimento no acesso à Internet e uso de *smartphones* ao redor do mundo. Em uma pesquisa feita pela *Pew Research Center*¹, é revelado que no ano de 2015, uma média de 54% entre 21 países emergentes (incluindo o Brasil) relataram que seus cidadãos faziam uso de Internet ao menos uma vez ao dia, ou possuíam um *smartphone*. Já entre 11 países considerados desenvolvidos, a porcentagem era de 87% (POUSHTER; STEWART, 2016).

A Internet mudou profundamente o modo como as pessoas se comunicam e expressam suas visões sobre os mais diversos assuntos, o que inclui experiências com produtos e serviços consumidos pelas mesmas. Os usuários da *Web* cada vez mais buscam opiniões de outros consumidores, antes de realizar alguma compra ou visitar algum local (PANG; LEE, 2008). Pesquisas feitas pela companhia *Comscore*² e pela *Pew Research Center*, no ano de 2007, indicam que dentre mais de 2000 pessoas entrevistadas nos Estados Unidos, cerca de 87% relataram que opiniões sobre hotéis, restaurantes e outros serviços tinham grande influência em suas decisões de compra. Além disso, também indicam que os consumidores estavam dispostos a pagar entre 20% e 99% a mais por um produto com classificação “5 estrelas” do que por outro que recebeu classificação “4 estrelas”, dependendo do tipo de produto comercializado (COMSCORE; The Kelsey Group, 2007; HARRIGAN, 2008).

Outro tópico onde a busca por outras opiniões é significativa para o público, é a política. De acordo com pesquisa divulgada também pela *Pew Research Center*, feita com mais de 2500 pessoas durante as eleições americanas de 2006, cerca de 29% dos entrevistados acompanhavam *sites* com opiniões políticas divergentes das suas, 34% buscavam relatos de pessoas de fora de suas comunidades, e 27% pesquisavam por endossos de organizações externas, mostrando a grande importância destas informações para a definição de posições políticas dos eleitores (RAINEE; HARRIGAN, 2007).

Além de impactar em decisões de compra e posicionamentos políticos, a influência de opiniões de terceiros não é diferente para o consumo de produtos da indústria cinematográfica, como observado nos estudos de Pentheny (2015) e Basuroy, Chatterjee e Ravid (2003). Muitos espectadores recorrem a *sites* especializados em cinema, como os conhecidos *Internet Movie Database*³ (IMDb) e *Rotten Tomatoes*⁴, em busca de críticas de especialistas e de outros usuários para decidir se assistem, ou não, a algum filme, além de deixarem suas próprias opiniões sobre diferentes obras.

¹ <https://www.pewinternet.org/>

² <https://www.comscore.com/>

³ <https://www.imdb.com/>

⁴ <https://www.rottentomatoes.com/>

Toda esta troca de informações entre a audiência gera uma quantidade enorme de dados, que vem sendo frequentemente utilizada na obtenção de *feedback* do público sobre novos lançamentos, por meio do processamento de linguagem natural (PLN) e do aprendizado de máquina, duas subáreas da inteligência artificial com destaque crescente, abordadas em diversas obras, como [Mitchell \(1997\)](#), [Russell e Norvig \(2009\)](#) e [Manning e Schütze \(1999\)](#). PLN é a subárea responsável pela aplicação de técnicas computacionais com a finalidade de aprender e produzir conteúdos em linguagem humana ([HIRSCHBERG; MANNING, 2015](#)). Aprendizado de máquina, por sua vez, pode ser descrito como o estudo de algoritmos computacionais que melhoram automaticamente por meio da experiência, com os quais pode-se executar tarefas de natureza preditiva e descritiva, como classificação, regressão, agrupamento, entre outras ([MITCHELL, 1997](#); [FACELI et al., 2011](#)).

Tanto o processamento de linguagem natural como o aprendizado de máquina constituem a base para a tarefa de análise de sentimentos, em que documentos de texto são examinados com o objetivo de determinar a polaridade do mesmo, classificando as sentenças analisadas como positivas ou negativas. A análise de sentimentos em críticas de filmes já vem sendo abordada na literatura nos últimos anos, fazendo uso de diversas técnicas, com o trabalho de [Pang, Lee e Vaithyanathan \(2002\)](#) sendo um de grande destaque, aplicando diversos algoritmos de aprendizado de máquina para tal tarefa. Outros trabalhos utilizam bibliotecas dedicadas à análise de sentimentos, como por exemplo a SentiWordNet, visto em [Singh et al. \(2013\)](#), ou redes neurais artificiais, utilizadas por [Zhang et al. \(2018\)](#).

Com a crescente importância da avaliação do público sobre os mais diversos filmes, tanto para outros espectadores como para produtores, este trabalho objetiva desenvolver um estudo de técnicas de PLN e aprendizado de máquina para análise de sentimentos de críticas de filmes, buscando avaliar quais são os métodos mais adequados para esta tarefa, já que haverá uma utilização cada vez maior de tais técnicas na avaliação da receptividade do público aos produtos da indústria cinematográfica, sendo uma ferramenta de extrema importância para o desenvolvimento de filmes futuros.

Neste trabalho, serão abordadas as técnicas mais conhecidas para processamento do texto contido nas críticas, como lematização, *stemming*, remoção de *stop words* e representação por *Bag-of-Words*. Também serão abordados os principais algoritmos de classificação conhecidos, sendo eles o *Naïve Bayes*, *Support Vector Machine* (SVM), *Random Forest*, *Multilayer Perceptron* (MLP), Árvore de Decisão e *k-Nearest Neighbours* (k-NN). Por fim, o desempenho de cada modelo será analisado com métricas variadas, como acurácia, *precision* e *recall*.

O trabalho está organizado do seguinte modo: no Capítulo 2, são descritos os principais conceitos teóricos envolvidos no trabalho. No Capítulo 3, são apresentados os trabalhos relacionados. No Capítulo 4, são descritos os objetivos do trabalho e métodos utilizados. No Capítulo 5, são apresentados os resultados obtidos. Por fim, no Capítulo 6, são apresentadas as considerações finais.

2 Fundamentação teórica

Neste capítulo, os principais conceitos relacionados a análise de sentimentos, processamento de textos, aprendizado de máquina e avaliação de desempenho dos algoritmos são descritos.

2.1 Análise de sentimentos

2.1.1 Definição

A análise de sentimentos é uma área de pesquisa dentro do processamento de linguagem natural (PLN), dada pelo estudo computacional das opiniões, avaliações e emoções das pessoas, em relação a outros indivíduos, eventos e produtos (LIU; ZHANG, 2012).

Pang e Lee (2008) define o termo “análise de sentimentos” como o tratamento computacional de opiniões, sentimentos e subjetividade presentes em um texto. Enquanto isso, outros artigos na literatura o definem como uma tarefa mais específica, dada pela classificação das *reviews* quanto à sua polaridade, como positiva, negativa, neutra, entre outras nuances, sendo esta a definição mais compatível com o objetivo deste trabalho.

2.1.2 Contexto histórico

Alguns dos trabalhos precursores da área datam do fim dos anos 1970 e início dos anos 1980, aproximadamente. Em Wilks e Bien (1983), por exemplo, é proposto um modelo de crenças que permite a compreensão computacional da linguagem natural. Em outras palavras, em uma conversa entre duas pessoas, o ouvinte possui ideias a respeito das crenças emitidas pelo locutor da conversa. O autor do artigo, por sua vez, argumenta que é possível desenvolver um algoritmo que constrói suas próprias crenças a partir daquelas do locutor e do ouvinte. Posteriormente, o foco dos artigos estava na interpretação do texto e das narrativas analisadas, como em Hearst (1992) e Wiebe e Rapaport (1988).

As pesquisas sobre análise de sentimentos, mais especificamente, começaram a se multiplicar no início dos anos 2000, com o uso cada vez maior de algoritmos de aprendizado de máquina em tarefas de processamento de linguagem natural e recuperação de informações, maior disponibilidade de conjuntos de dados para treino destes algoritmos, além do crescente interesse por parte da indústria, por conta das diversas aplicações possíveis da análise de opiniões, nas áreas comerciais e de inteligência de negócios, por exemplo.

A seguir, serão descritas algumas aplicações da análise de sentimentos na indústria.

2.1.3 Aplicações

A análise de sentimentos pode ser utilizada de diversas formas, como a já citada aplicação em inteligência de negócios, utilizando sites especializados em *reviews* de clientes ou redes sociais (como Twitter ou Facebook), buscando um *feedback* do público geral a respeito de algum produto, como será feito neste trabalho. Esta abordagem é usada em diversos setores da indústria, como na hotelaria (KASPER; VELA, 2011; SHI; LI, 2011), transporte aéreo (WAN; GAO, 2015; LIAU; TAN, 2014), alimentação (GAN et al., 2017; KANG; YOO; HAN, 2012) e tecnologia (ARORA; LI; NEVILLE, 2015; GUZMAN; MAALEJ, 2014).

Outros usos incluem a melhoria de sistemas de recomendação, buscando trazer sempre itens aprovados pelo público, como visto em Tatemura (2000), além de aplicação em sistemas de respostas, levando em conta o sentimento dos consumidores ao realizar suas perguntas (LITA et al., 2005).

Por fim, pode-se destacar a presença da análise de sentimentos em assuntos governamentais e relacionados à política, como na mineração de opiniões de eleitores (LAVIER; BENOIT; GARRY, 2003; MALOUF; MULLEN, 2006) e dos cidadãos em geral sobre propostas de regulamentação do governo, por exemplo (KWON; SHULMAN; HOVY, 2006).

2.1.4 Desafios

Dentre os diversos desafios presentes nesta área de estudo, Pang e Lee (2008) ressaltam que na análise de sentimentos, existem diferentes graus de positividade, força de sentimento, entre outros indicadores de subjetividade, possuindo grande influência na classificação do texto em poucas classes, que fazem uma generalização entre diversos domínios.

Outra das dificuldades se encontra em determinar um conjunto de palavras que servirá como base para o classificador realizar sua tarefa, como abordado por Pang, Lee e Vaithyanathan (2002). Neste estudo, foi pedido a duas pessoas que elaborassem uma lista de palavras consideradas positivas ou negativas pelos mesmos, e notou-se uma grande diferença na escolha dos termos, em comparação entre si e com um conjunto de palavras selecionadas por técnicas de estatística, demonstrando o grande peso da subjetividade na análise de sentimentos por algoritmos.

Opiniões sobre filmes advindas do site *IMDb*, por exemplo, também podem ser classificadas como “fortemente negativa”, “levemente negativa”, “neutra”, e assim por diante. Isto ocorre por conta da mistura que pode haver de sentimentos em uma única opinião, onde o espectador enfatiza pontos fortes e fracos do filme, ao mesmo tempo. Além disso, um fator que impacta no grau de positividade de uma *review* é a presença de sentenças com opiniões do usuário junto de fatos da vida real trazidos pelo mesmo, que podem ser prejudiciais à classificação (TABOADA et al., 2011).

Por fim, outro desafio enfrentado se dá pelo fato de que muitas vezes, as opiniões dadas

pelos espectadores possuem um sentimento mais sutil, com uma escolha de palavras que para um humano, pode ser trivial determinar se é uma opinião positiva ou negativa (como em uma situação de uso de ironia ou sarcasmo), mas para um dado classificador, não fica tão claro qual é a polaridade da *review* analisada.

Com isso, a seguir, inicia-se a descrição do processamento de textos para extração de emoções.

2.2 Processamento de textos

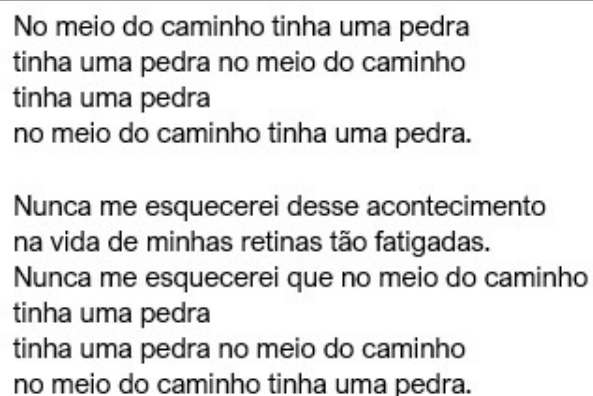
O processamento de textos é parte essencial da análise de sentimentos em opiniões. Para que os algoritmos de classificação possam realizar esta tarefa, é preciso que os textos das críticas estejam em um formato interpretável pelos modelos gerados, uma vez que se tratam de modelos matemáticos. Todo o processamento envolve diversas técnicas e etapas, detalhadas a seguir.

2.2.1 Pré-processamento

O pré-processamento dos textos é crucial não somente para convertê-los em estruturas interpretáveis pelos algoritmos de aprendizado de máquina, mas para melhorar a performance destes, removendo informações irrelevantes para a tarefa de classificação textual (UYSAL; GUNAL, 2014). É comum aplicar diversas técnicas de pré-processamento em um único conjunto. Algumas das mais utilizadas são descritas nas próximas seções.

Para visualização de cada técnica, será utilizado como exemplo o poema “No Meio do Caminho”, conforme visto abaixo, na Figura 1.

Figura 1: Poema “No Meio do Caminho”, por Carlos Drummond de Andrade



No meio do caminho tinha uma pedra
tinha uma pedra no meio do caminho
tinha uma pedra
no meio do caminho tinha uma pedra.

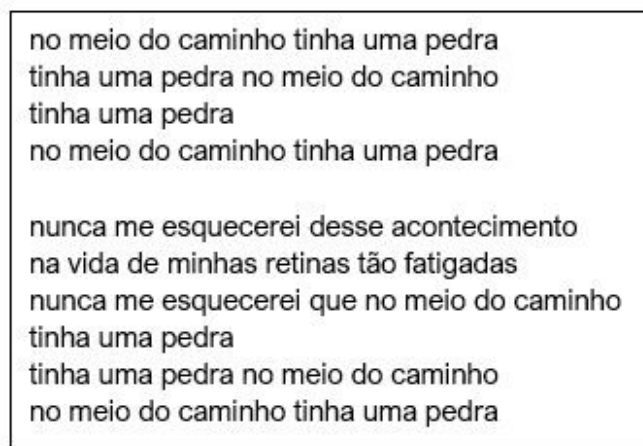
Nunca me esquecerei desse acontecimento
na vida de minhas retinas tão fatigadas.
Nunca me esquecerei que no meio do caminho
tinha uma pedra
tinha uma pedra no meio do caminho
no meio do caminho tinha uma pedra.

Fonte: a autora.

2.2.1.1 Conversão para caixa baixa e remoção de pontuações

Estas duas etapas, mesmo sendo triviais, são importantes para o pré-processamento do texto. Especialmente em textos com repetições de uma mesma palavra, a conversão para caixa baixa evita que diferentes formas dela sejam interpretadas como palavras distintas. As pontuações, por sua vez, não possuem significância para a análise de sentimentos, devendo, portanto, ser removidas.

Figura 2: Exemplo de conversão para caixa baixa e remoção de pontuações

A figura mostra um exemplo de conversão de texto para caixa baixa e remoção de pontuações. O texto original, em caixa alta, é: "NO MEIO DO CAMINHO TINHA UMA PEDRA. TINHA UMA PEDRA NO MEIO DO CAMINHO. TINHA UMA PEDRA. NO MEIO DO CAMINHO TINHA UMA PEDRA. NUNCA ME ESQUECEREI DESSE ACONTECIMENTO NA VIDA DE MINHAS RETINAS TÃO FATIGADAS. NUNCA ME ESQUECEREI QUE NO MEIO DO CAMINHO TINHA UMA PEDRA. TINHA UMA PEDRA NO MEIO DO CAMINHO. NO MEIO DO CAMINHO TINHA UMA PEDRA." O texto convertido para caixa baixa e com as pontuações removidas é: "no meio do caminho tinha uma pedra tinha uma pedra no meio do caminho tinha uma pedra no meio do caminho tinha uma pedra nunca me esquecerei desse acontecimento na vida de minhas retinas tão fatigadas nunca me esquecerei que no meio do caminho tinha uma pedra tinha uma pedra no meio do caminho no meio do caminho tinha uma pedra".

no meio do caminho tinha uma pedra
tinha uma pedra no meio do caminho
tinha uma pedra
no meio do caminho tinha uma pedra

nunca me esquecerei desse acontecimento
na vida de minhas retinas tão fatigadas
nunca me esquecerei que no meio do caminho
tinha uma pedra
tinha uma pedra no meio do caminho
no meio do caminho tinha uma pedra

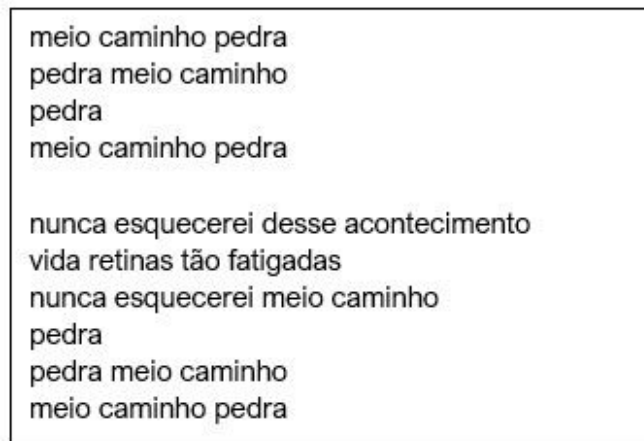
Fonte: a autora.

2.2.1.2 Remoção de *stop words*

As palavras conhecidas como *stop words* são aquelas que apesar de largamente utilizadas em um certo idioma, não dependem de um tópico específico, ou seja, que não carregam informação, como pronomes, artigos e conjunções (UYSAL; GUNAL, 2014; SRIVIDHYA; ANITHA, 2010).

Para a língua inglesa, existem listas de *stop words* disponíveis para uso na etapa de remoção, como a contida no sistema de recuperação de informações SMART, desenvolvido na Universidade Cornell (BUCKLEY; SALTON; ALLAN, 1993). Também existem tais listas para português brasileiro, como na biblioteca NLTK, em linguagem Python (vista na Seção 4.2).

O processo de remoção de *stop words* pode ser visto na Figura 3.

Figura 3: Exemplo de remoção de *stop words*A caixa contém o texto original com as palavras 'meio', 'caminho' e 'pedra' em negrito, indicando que são as stop words a serem removidas.

meio caminho pedra
pedra meio caminho
pedra
meio caminho pedra

nunca esquecerei desse acontecimento
vida retinas tão fatigadas
nunca esquecerei meio caminho
pedra
pedra meio caminho
meio caminho pedra

Fonte: a autora.

Com as *stop words* removidas, o texto pode ser submetido à técnica de lematização ou de *stemming*, ambas vistas a seguir.

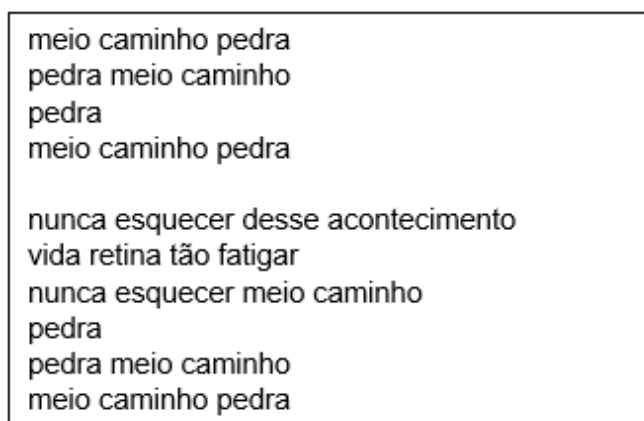
2.2.1.3 Lematização

Na aplicação desta técnica, as palavras passam por uma análise em busca de termos flexionados, como conjugações de verbos e plurais, por exemplo. Após a detecção destas palavras flexionadas, estas são convertidas em seu lema, ou seja, sua forma básica, conforme [Korenius et al. \(2004\)](#).

Uma desvantagem da lematização está nas palavras homógrafas (com mesma grafia, mas significados distintos), que podem causar problemas de ambiguidade.

Ao aplicar esta técnica, obtém-se o resultado visto na Figura 4.

Figura 4: Exemplo de lematização

A caixa contém o texto lematizado, onde as palavras 'meio', 'caminho' e 'pedra' foram convertidas para sua forma básica (lemas).

meio caminho pedra
pedra meio caminho
pedra
meio caminho pedra

nunca esquecer desse acontecimento
vida retina tão fatigar
nunca esquecer meio caminho
pedra
pedra meio caminho
meio caminho pedra

Fonte: a autora.

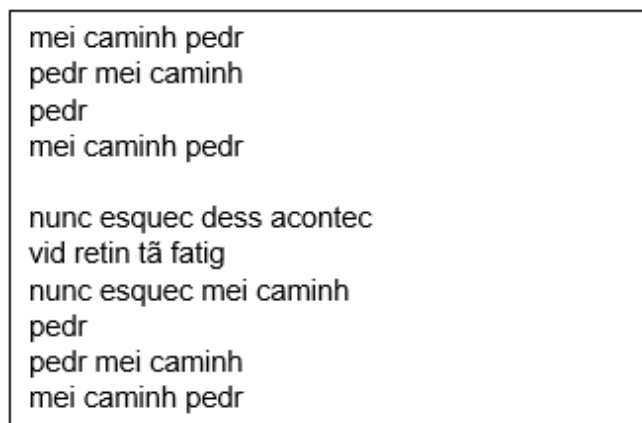
2.2.1.4 Stemming

Como alternativa à lematização, a técnica de *stemming*, conhecida em Português como *stemização*, é uma das mais utilizadas para recuperação de informações não somente na língua inglesa, mas também em textos de diversos outros idiomas, como esloveno, francês e grego, como visto em [Korenius et al. \(2004\)](#).

Esta técnica envolve, essencialmente, a remoção dos sufixos das palavras, reduzindo-as ao seu radical ([PORTER, 1980](#)). Atualmente, existem diversos algoritmos para a automatização desta tarefa, sendo o algoritmo de Porter um dos mais conhecidos e aplicados em textos em inglês, além do algoritmo de Lancaster.

Na língua portuguesa, existem alguns algoritmos apresentados na literatura, como o Removedor de Sufixos da Língua Portuguesa (RSLP), proposto por [Orengo e Huyck \(2005\)](#), exemplificado na Figura 5, aplicado após a remoção de *stop words*.

Figura 5: Exemplo de *stemming*



mei caminh pedr
pedr mei caminh
pedr
mei caminh pedr

nunc esquec dess acontec
vid retin tã fatig
nunc esquec mei caminh
pedr
pedr mei caminh
mei caminh pedr

Fonte: a autora.

2.2.1.5 Part-of-Speech Tagging

Por fim, a técnica de *POS Tagging* consiste em atribuir uma *tag* morfosintática adequada a cada palavra do texto, de acordo com o contexto em que está inserida, para desambiguação do texto analisado ([MÀRQUEZ; RODRÍGUEZ, 1998](#)).

Apesar de todas as classes de palavras terem importância dentro da análise de sentimentos, os adjetivos possuem um papel especial nesta tarefa, pois muitas vezes são bons indicadores do sentimento presente na opinião, como visto em [Pang e Lee \(2008\)](#).

Um exemplo de aplicação do *Part-of-Speech Tagging*, após o uso de *stemming*, pode ser visto na Tabela 1.

Tabela 1: Exemplo de *Part-of-Speech Tagging*

mei	caminh	pedr	nunc	esquec	dess	acontec	vid	retin	tão	fatig
numeral	subst.	subst.	adv. (tempo)	verbo	adv. (lugar)	subst.	subst.	subst.	adv. (intens.)	adjetivo

Fonte: a autora.

2.2.2 *Bag-of-Words*

Bag-of-Words é um método bastante usado para representação de documentos em texto ou de imagens. No caso de texto, este é representado como um vetor, onde cada posição possui o número de aparições de uma palavra ao longo do conteúdo de tal documento, ou seja, sua frequência absoluta (BOULIS; OSTENDORF, 2005).

Um exemplo de representação por *Bag-of-Words* pode ser visto na Tabela 2.

Tabela 2: Exemplo de *Bag-of-Words*

mei	caminh	pedr	nunc	esquec	dess	acontec	vid	retin	tão	fatig
6	6	7	1	1	1	1	1	1	1	1

Fonte: a autora.

Na literatura, existem diversos trabalhos que propõem modificações da técnica de *Bag-of-Words*, como em Sahlgren e Cöster (2004), onde é apresentado o *Bag-of-Concepts*, em que se realiza seleção de atributos para representação de um texto de acordo com os conceitos contidos nele, agrupando palavras com significado similar.

2.2.3 Geração do vetor de atributos

Com o texto já processado e representado no formato de *Bag-of-Words*, a próxima etapa é definir o vetor de atributos que será utilizado na tarefa de classificação.

Como visto anteriormente, no *Bag-of-Words*, os textos são representados com base na medida de *Term Frequency* (TF), mas o vetor resultante pode ser normalizado ou dimensionado de diversas formas, como por exemplo, com *Term Frequency-Inverse Document Frequency* (TF-IDF). Estas duas medidas serão descritas a seguir.

2.2.3.1 *Term Frequency (TF)*

A frequência do termo, conhecida em inglês como *Term Frequency*, trata-se da simples razão entre o número de ocorrências de um termo e o total de palavras contidas em um dado

texto avaliado (SALTON; BUCKLEY, 1988), como visto na Equação abaixo.

$$tf(\text{termo}) = \frac{\text{número de ocorrências do termo}}{\text{total de palavras no texto}} \quad (2.1)$$

2.2.3.2 Term Frequency-Inverse Document Frequency (TF-IDF)

Com a medida TF-IDF, por sua vez, cada termo possui uma frequência relativa, calculada por meio de uma proporção inversa entre sua frequência absoluta em um dado texto e a porcentagem de documentos em que este termo aparece (RAMOS, 2003). Conforme Silge e Robinson (2017), a medida de *Inverse Document Frequency* (IDF) diminui o peso de palavras comuns (como *stop words*, vistas anteriormente) e aumenta o de palavras não tão usadas dentro de um conjunto de textos, e seu cálculo se dá pela Equação a seguir:

$$idf(\text{termo}) = \ln \left(\frac{N}{n} \right) \quad (2.2)$$

onde:

- N = total de textos contidos no conjunto de dados; e
- n = número de textos no *dataset* que contêm o termo analisado.

Com o IDF calculado, a medida TF-IDF, por fim, se dá pela seguinte Equação:

$$tf-idf(\text{termo}) = tf(\text{termo}) \times idf(\text{termo}) \quad (2.3)$$

Trata-se de uma medida eficiente e simples de ser implementada. Porém, uma grande desvantagem está no fato de não conseguir relacionar sinônimos, considerando-os como palavras distintas.

2.2.4 Embedding via redes neurais

Além dos métodos para representação de texto selecionados para experimentos neste trabalho, e que foram descritos acima, uma alternativa conhecida na literatura envolve *word embedding* por meio do uso de redes neurais.

Word embedding é o processo de representação de palavras como vetores, capaz de captar relações sintáticas e semânticas entre elas, extraíndo estas representações de uma grande quantidade de dados não estruturados (MIKOLOV et al., 2013a; MIKOLOV et al., 2013b). Dentre os modelos mais conhecidos de *word embedding* via redes neurais, há os modelos Skip-gram, propostos em Mikolov et al. (2013a), e Word2vec, proposto em Mikolov et al. (2013b).

2.2.4.1 Skip-gram

A primeira arquitetura do modelo Skip-gram, *Continuous Bag-of-Words Model* (CBOW), baseia-se na técnica de *Bag-of-Words*, vista na Seção 2.2.2, onde a palavra atual é prevista de acordo com o contexto em que está inserida. CBOW é baseado no *Feedforward Neural Net Language Model* (NNLM), proposto por [Bengio et al. \(2003\)](#), que consiste em uma rede neural com 4 tipos de camadas: de entrada, de projeção, escondida e de saída.

No modelo *Feedforward NNLM*, as palavras são codificadas na camada de entrada com codificação “1 de V ” (onde V representa o número de palavras no vocabulário analisado). Em seguida, esta camada é projetada na camada de projeção, por meio de uma matriz de projeção compartilhada, e a camada escondida calcula a Distribuição de Probabilidade para cada palavra do vocabulário, com base na probabilidade de uma certa palavra pertencer a um texto analisado em um dado contexto, sendo este resultado passado para a camada de saída, de tamanho V .

O modelo CBOW, por sua vez, não possui a camada escondida, e sua camada de projeção é compartilhada entre todas as palavras do vocabulário, sendo todas projetadas na mesma posição. Além disso, uma vez que a palavra atual é prevista com base no contexto - representado de modo distribuído e contínuo -, utilizam-se palavras do futuro e do passado para a predição.

Primeiramente, gera-se um vetor de contexto para cada palavra, considerando as palavras ao redor, usando a Equação:

$$\mathbf{c} = \frac{1}{2b} \sum_{i \in [-b, b] - \{0\}} \mathbf{w}_i \quad (2.4)$$

onde:

- b = hiperparâmetro para definir intervalo de palavras dentro de um contexto; e
- \mathbf{w}_i = i -ésima palavra ao redor.

A probabilidade calculada pela camada escondida (e passada para a camada de saída) é calculada por meio de uma função de ativação Softmax, e se dá pela Equação a seguir, vista em [Ling et al. \(2015\)](#):

$$p(\mathbf{v}_0 | \mathbf{w}_{[-b, b] - \{0\}}) = \frac{\exp \mathbf{v}_0^\top \mathbf{O}_{\mathbf{c}}}{\sum_{v \in V} \exp \mathbf{v}^\top \mathbf{O}_{\mathbf{c}}} \quad (2.5)$$

sendo:

- \mathbf{v}_0 = predição da palavra atual;
- $\mathbf{w}_{[-b, b] - \{0\}}$ = todas as palavras em torno da palavra atual \mathbf{w}_0 ; e

- O_c = projeção do vetor de contexto c no vocabulário V .

A complexidade do treinamento deste modelo é dada por:

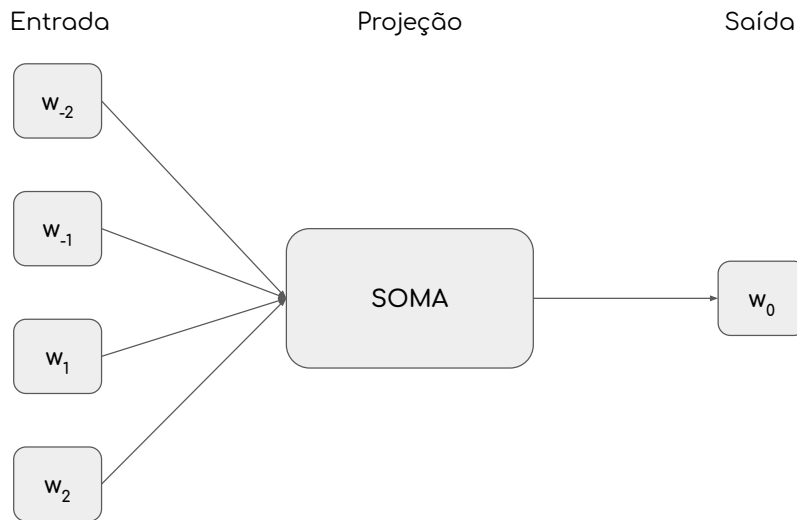
$$Q = N \times D + D \times \log_2 V \quad (2.6)$$

onde:

- Q = complexidade do treinamento;
- N = número de palavras do passado, codificadas na camada de entrada;
- $N \times D$ = dimensão da camada de projeção; e
- V = número de palavras em todo o vocabulário.

Sua arquitetura pode ser vista na Figura 6.

Figura 6: Arquitetura do modelo *Continuous Bag-of-Words Model (CBOW)*



Fonte: a autora. Adaptado de Mikolov et al. (2013a).

Enquanto isso, a segunda arquitetura, *Continuous Skip-gram Model*, maximiza a classificação de uma dada palavra com base em outra na mesma sentença (MIKOLOV et al., 2013a). Neste modelo, considera-se um intervalo de palavras antes e depois da palavra sendo atualmente predita. Durante o treinamento, dá-se um peso menor às palavras mais distantes e um

peso maior às mais próximas, que normalmente possuem uma relação maior com a palavra atual.

O treino deste modelo é feito com a maximização da função objetivo, vide Equação a seguir:

$$\frac{1}{V} \sum_{v=1}^V \sum_{-b \leq j \leq b, b \neq 0} \log p(w_{v+j}|w_v) \quad (2.7)$$

onde:

- V = número de palavras no conjunto de treino;
- b = hiperparâmetro para definir intervalo de palavras dentro de um contexto; e
- $p(w_{v+j}|w_v)$ representa uma probabilidade da palavra w_{v+j} ser vizinha de w_v no texto analisado, calculada por Softmax.

A função de Softmax usada na função objetivo é dada por:

$$p(w_O|w_I) = \frac{\exp(v'_{w_O}{}^\top v_{w_I})}{\sum_{w=1}^W \exp(v'_w{}^\top v_{w_I})} \quad (2.8)$$

onde:

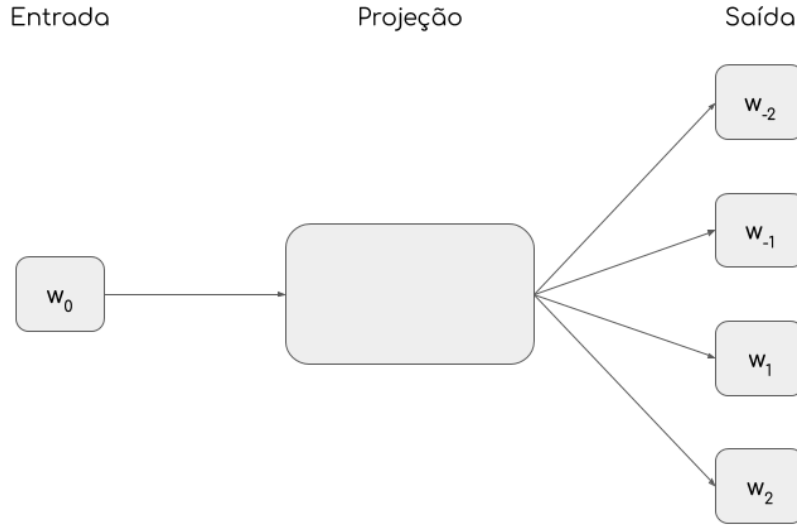
- v_w = representação da palavra w em vetores de entrada;
- v'_w = representação da palavra w em vetores de saída; e
- W = número de palavras no vocabulário.

A complexidade de treinamento, neste caso, é dada pela Equação:

$$Q = C \times (D + D \times \log_2 V) \quad (2.9)$$

Sendo que C é a medida da distância máxima entre as palavras a serem analisadas.

A arquitetura do *Continuous Skip-gram Model* é ilustrada na Figura 7.

Figura 7: Arquitetura do modelo *Continuous Skip-gram Model*

Fonte: a autora. Adaptado de Mikolov et al. (2013a).

2.2.4.2 Word2vec

Word2vec, por sua vez, é um conjunto de 3 modelos, dados como extensão do modelo Skip-gram. São muito utilizados na literatura, compostos por uma rede neural de 2 camadas, cada um tendo uma modificação.

A primeira modificação feita está na função Softmax, onde se usa uma versão hierárquica dela, apresentada por Morin e Bengio (2005). Nesta versão, utiliza-se apenas $\log_2 W$ de todos os vetores de saída, para o cálculo das Distribuições de Probabilidade.

A camada de saída é representada por uma árvore binária, utilizada pela Softmax hierárquica para os cálculos, como visto em Mikolov et al. (2013b). As folhas das árvores contêm todas as W palavras, e cada nó representa as probabilidades relativas de seus filhos.

A função Softmax hierárquica pode ser definida como:

$$p(w|w_I) = \prod_{j=1}^{L(w)-1} \sigma \left(\mathbb{I}[n(w, j+1) = ch(n(w, j))] \cdot v'_{n(w, j)}{}^\top v_{w_I} \right) \quad (2.10)$$

onde:

- w = palavra analisada;
- $L(w)$ = comprimento do caminho da raiz da árvore até a palavra w ;

- $n(w, j)$ = j -ésimo nó no caminho da raiz até w , onde $n(w, 1)$ representa a raiz;
- $ch(n)$ representa um filho fixo arbitrário de n ;
- $\llbracket x \rrbracket = 1$, se x é verdadeiro, ou -1 , caso contrário; e
- $\sigma(x) = 1/(1 + \exp(-x))$.

Outra modificação realizada utiliza o conceito de Amostragem Negativa (do inglês *Negative Sampling*), que é uma simplificação do *Noise Contrastive Estimation* (NCE).

O modelo NCE, apresentado por [Gutmann e Hyvärinen \(2012\)](#), faz uma diferenciação entre dados e ruídos utilizando regressão logística, maximizando o logaritmo da probabilidade do Softmax (vide Equação (2.8)). Porém, o modelo Skip-gram foca na representação em vetores de cada palavra. Portanto, simplifica-se a função objetivo, definindo a Amostragem Negativa conforme a Equação a seguir:

$$\log \sigma(v'_{w_O}{}^\top v_{w_I}) + \sum_{i=1}^k \mathbb{E}_{w_i \sim P_{n(w)}} \left[\log \sigma(-v'_{w_i}{}^\top v_{w_I}) \right] \quad (2.11)$$

sendo que a tarefa é distinguir a palavra alvo w_O da distribuição de ruídos $P_n(w)$ por meio de regressão logística, com k *samples* negativos para cada amostra de dados ([MIKOLOV et al., 2013a](#)). Esta equação substitui cada termo $P(w_O|w_I)$ presente na função objetivo do modelo Skip-gram original.

Por fim, a última modificação apresentada é a subamostragem de palavras frequentes, como as *stop words*, vistas na Seção 2.2.1.2. No modelo Skip-gram, a representação das palavras frequentes em vetores não passa por uma mudança significativa, mesmo depois do treino com diversos exemplos.

A subamostragem é feita de acordo com a Equação:

$$p(w_i) = 1 - \sqrt{\frac{t}{f(w_i)}} \quad (2.12)$$

onde:

- $p(w_i)$ = probabilidade da palavra w_i ;
- $f(w_i)$ = frequência da palavra w_i ; e
- t representa o limite de subamostragem.

De acordo com [Mikolov et al. \(2013a\)](#), o modelo com subamostragem de palavras frequentes possui as vantagens de acelerar o aprendizado e aumentar a acurácia de vetores aprendidos de palavras mais raras.

Em comparação com outros métodos de *word embedding*, a realização desse processo em redes neurais faz com que várias regularidades e padrões linguísticos sejam codificados explicitamente, além de que estas palavras podem ser representadas como transformações lineares. Mais detalhes sobre o funcionamento de redes neurais podem ser vistos na Seção 2.4.

2.3 Aprendizado de máquina

Aprendizado de máquina (AM) é uma área de pesquisa da computação que envolve a indução de hipóteses ou funções, a partir de experiências passadas, para resolver um determinado problema de natureza preditiva ou descritiva (FACELI et al., 2011). Este processo também é conhecido como *aprendizado indutivo*, e pode ser dividido em duas categorias principais:

- **Aprendizado supervisionado:** paradigma de aprendizado em que os algoritmos de AM recebem conjuntos de dados rotulados e definem um modelo a partir destes, sendo este usado posteriormente na rotulação de exemplos novos em problemas de natureza preditiva, ou seja, em tarefas de *classificação* e *regressão* (MONARD; BARANAUSKAS, 2003; FACELI et al., 2011);
- **Aprendizado não supervisionado:** paradigma usado em problemas de natureza descritiva, onde o algoritmo faz uma análise dos dados recebidos, explorando-os e descrevendo-os, determinando se alguns deles podem ser agrupados de alguma forma, como em tarefas de *agrupamento*, *associação* e *sumarização*.

Além destas duas categorias, existem também o *aprendizado semissupervisionado*, em que se utiliza uma grande quantidade de dados não rotulados juntamente com dados rotulados, para desenvolver classificadores com maior acurácia (ZHU, 2005), e o *aprendizado por reforço*, onde um comportamento é assimilado com base em tentativa e erro (KAELBLING; LITTMAN; MOORE, 1996), em tarefas cuja meta é recompensar uma ação positiva e punir por uma ação negativa.

Neste trabalho, a tarefa a ser realizada sobre as críticas de filmes é de classificação, uma vez que o problema envolve determinar se uma crítica possui sentimento positivo ou negativo. A seguir, são descritos os principais conceitos e algoritmos utilizados para tal tarefa, dentro do campo do aprendizado de máquina.

2.3.1 Conceitos e definições

Existem diversas definições e conceitos básicos de AM aplicados e utilizados na literatura, como visto em Monard e Baranauskas (2003), sendo eles definidos a seguir, para melhor entendimento do trabalho a ser desenvolvido.

2.3.1.1 Exemplo

Também chamado de *dado*, um exemplo é o objeto de interesse para utilização em algoritmos de AM. É um vetor de valores de atributos, essenciais para realização de análises desse dado.

No contexto do presente trabalho, cada uma das opiniões contidas nos *datasets* utilizados é um exemplo, e será feito um processamento de cada opinião, para converter seus textos em um vetor de atributos, preparando o exemplo para a análise de sentimentos de fato.

2.3.1.2 Atributo

Um atributo é a descrição de alguma característica do exemplo analisado, podendo ser nominal ou contínuo, e ainda podendo não possuir um valor, como quando o atributo não se aplica a um exemplo. Para melhor rotulação de exemplos, é importante considerar a relevância de cada atributo para o modelo usado. Conforme [Michalski e Kaufman \(1998\)](#), um atributo é considerado irrelevante se há uma descrição completa e consistente das classes a aprender sem ele, ou seja, se todos os exemplos podem ser classificados corretamente sem o atributo em questão. Por conta disso, para aumentar o desempenho dos classificadores a serem usados, faz-se necessário selecionar os atributos com boa capacidade preditiva, de acordo com o problema a ser resolvido.

2.3.1.3 Classe

Também conhecida como *rótulo*, uma classe é um tipo especial de atributo de um dado exemplo, que se deseja prever e/ou aprender, como visto em [Monard e Baranauskas \(2003\)](#). Em tarefas de classificação, como a que será realizada neste trabalho, estas classes possuem valor nominal, uma vez que as críticas serão classificadas como positivas, negativas, neutras ou outras.

2.3.1.4 Conjunto de exemplos

Referenciado neste trabalho como *conjunto de dados* ou *dataset*, trata-se de uma coleção de exemplos, com seus respectivos valores de atributos e classe associada. Estes conjuntos são utilizados para treino e avaliação dos modelos gerados pelos algoritmos de AM, sendo divididos em dois subconjuntos disjuntos, onde um deles é o *conjunto de treino*, e o outro é o *conjunto de teste*. Pode-se, ainda, criar um terceiro subconjunto, destinado à validação destes modelos.

2.3.1.5 Classificador

Um classificador, também conhecido como *hipótese*, é o modelo gerado por um algoritmo, dado um conjunto de exemplos de treinamento. Este classificador, posteriormente, é utilizado para prever a classe de novos exemplos recebidos por ele.

2.3.1.6 Viés

O viés de aprendizado, conhecido em inglês como *bias*, é a preferência de uma hipótese sobre outra, ou conforme definição anterior, de um classificador sobre outro. É impossível ter um aprendizado sem viés, pois este é a base para o salto indutivo, permitindo escolher uma entre as diversas hipóteses consistentes para um problema, como visto em [Mitchell \(1980\)](#).

2.3.1.7 Distribuição e prevalência de classes

No caso de tarefas de classificação, pode-se calcular a distribuição das classes de um dado conjunto de exemplos, com base na Equação:

$$\text{distr}(C_j) = \frac{1}{n} \sum_{i=1}^n \|y_i = C_j\| \quad (2.13)$$

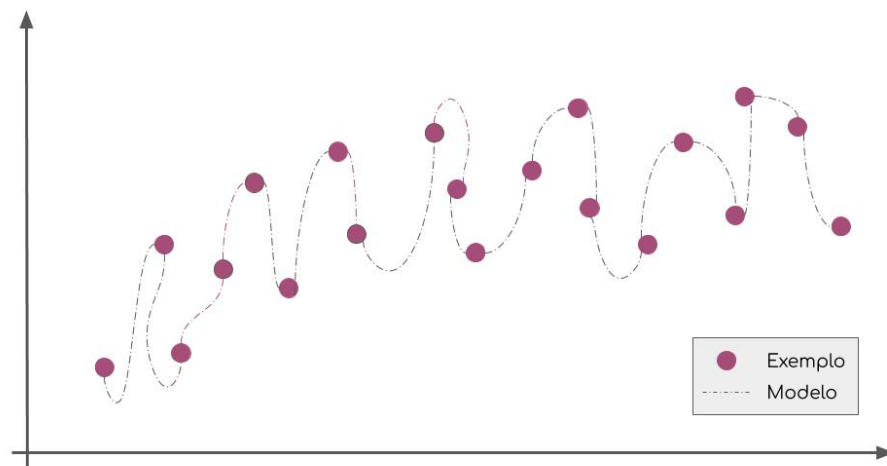
onde a distribuição da classe C_j é dada pela proporção de exemplos y_i pertencentes a ela, em relação ao total de exemplos no conjunto. Neste caso, $\|y_i = C_j\|$ denota a quantidade de exemplos y_i na classe C_j .

Esta medida de distribuição é importante para a análise do balanceamento de classes dentro de um *dataset*, pois quando há prevalência muito grande de uma delas, um classificador pode sempre classificar novos exemplos como pertencendo a ela, e dependendo do problema abordado, isso pode ser prejudicial. Um exemplo da importância da distribuição pode ser visto em [Mazurowski et al. \(2008\)](#), onde mostra-se o impacto do desbalanceamento de classes no treino de classificadores por redes neurais, destinados a diagnósticos médicos assistidos por computador.

2.3.1.8 *Underfitting* e *overfitting*

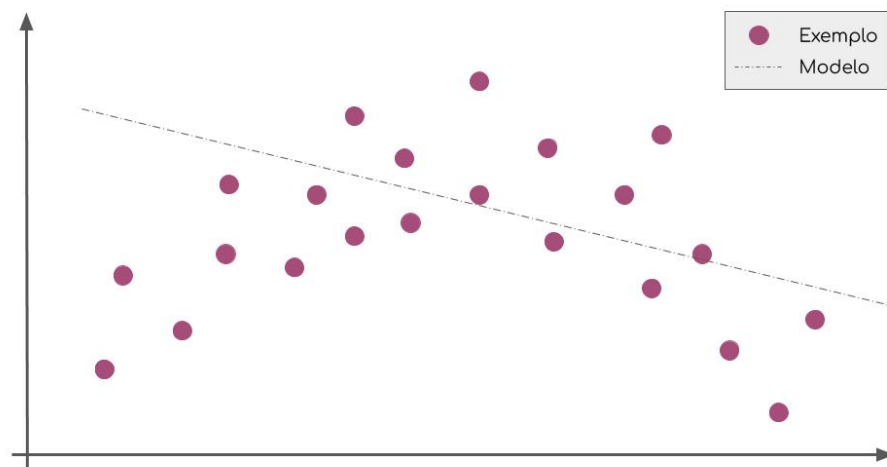
Ao treinar um modelo em cima de um dado conjunto de exemplos, podem ocorrer situações onde o aprendizado é deficiente, seja porque o classificador aprendeu demais ou não aprendeu o suficiente. Estes dois problemas são conhecidos como *overfitting* e *underfitting*.

No caso de *overfitting*, a hipótese se ajusta demais ao conjunto de treinamento fornecido ao algoritmo e não consegue generalizar o aprendizado, ou em outras palavras, apenas “decora” um padrão existente neste conjunto, não sendo capaz de prever outros exemplos novos. Um exemplo ilustrativo do problema de *overfitting* pode ser visto na Figura 8.

Figura 8: Ilustração do problema de *overfitting*

Fonte: a autora.

Já o problema de *underfitting* ocorre quando a hipótese não se ajusta o suficiente ao *dataset* de treino, ou seja, quando há generalização demais (AALST et al., 2010). Esse tipo de problema acontece se o conjunto de treino não possui muitos exemplos significativos para o aprendizado, ou quando o classificador tem um tamanho muito pequeno. O problema de *underfitting* é ilustrado na Figura 9.

Figura 9: Ilustração do problema de *underfitting*

Fonte: a autora.

2.3.1.9 Ruído

Quando ocorrem problemas no processo de geração, aquisição ou transformação de dados de um conjunto, ou quando há elementos classificados incorretamente, diz-se que este conjunto de exemplos possui ruído, ou seja, possui dados defeituosos que podem impactar no desempenho de classificadores (MONARD; BARANAUSKAS, 2003).

2.3.1.10 Poda

Poda é um processo que faz com que o aprendizado sobre um conjunto de treino gere uma hipótese mais genérica, permitindo lidar com os problemas de *overfitting* e de ruído. Esta técnica geralmente é aplicada em Árvores de Decisão e algoritmos de redes neurais, que serão vistos nas próximas seções.

Pode-se aplicar a poda ao ignorar alguns dos exemplos do conjunto durante a etapa de treinamento de um modelo, forçando erros de classificação (método *pré-poda*), ou ao gerar uma hipótese e depois a generalizar, por meio da eliminação de algumas de suas partes (método *pós-poda*). Na Árvore de Decisão, esta generalização é feita por meio do “corte” de alguns de seus ramos.

Com os conceitos definidos, serão descritos a seguir cada um dos algoritmos de AM estudados durante este trabalho.

2.3.2 Naïve Bayes

O algoritmo *Naïve Bayes* é um método probabilístico para realizar tarefas preditivas, que parte da hipótese de que os valores dos atributos de um determinado objeto independem de sua classe (FACELI et al., 2011). Como o próprio nome indica, é baseado no Teorema de Bayes, dado pela Equação a seguir, conforme Morettin e Bussab (2010):

$$P(A|B) = \frac{P(A) \cdot P(B|A)}{P(B)} \quad (2.14)$$

em que:

- $P(A|B)$ é a probabilidade do evento A ocorrer, dado que o evento B ocorre;
- $P(A)$ é a probabilidade do evento A ocorrer;
- $P(B|A)$ é a probabilidade do evento B ocorrer, dado que o evento A ocorre; e
- $P(B)$ é a probabilidade do evento B ocorrer.

O algoritmo é obtido a partir de uma regra de decisão, determinada pelo método *Maximum A Posteriori* (MAP). Neste método, dado um exemplo x e k classes possíveis, calcula-se a

probabilidade de x pertencer a cada classe y_i , $1 \leq i \leq k$. Então, a classe com maior probabilidade de associação ao exemplo x é selecionada. Esta regra de decisão é dada por:

$$y_{MAP} = \arg \max_i P(y_i | x) \quad (2.15)$$

O cálculo da probabilidade, por sua vez, é dado pela Equação (2.16), também referida como *função discriminante*:

$$P(y_i | x) \propto P(y_i) \prod_{j=1}^d P(x_j | y_i) \quad (2.16)$$

Onde dado um exemplo x com d atributos, calcula-se o produto das probabilidades de cada atributo x_j , $1 \leq j \leq d$, pertencer à classe y_i .

Este classificador obtém um desempenho bom em vários domínios, mesmo em cenários com grande dependência entre os atributos de um objeto (DOMINGOS; PAZZANI, 1997), ou com ruídos e atributos irrelevantes em meio aos dados (KONONENKO, 1991).

2.3.3 Support Vector Machine (SVM)

Support Vector Machine é um método não-paramétrico, somente utilizando os exemplos em si para o aprendizado (BOSER et al., 1992; CRISTIANINI; SHAW-TAYLOR, 2000). Este algoritmo baseia-se na Teoria do Aprendizado Estatístico (VAPNIK, 1999), que é composta por quatro teorias, servindo como base para obtenção de um classificador com boa capacidade de generalização, dentre todos os possíveis classificadores gerados por um certo algoritmo, levando em conta o conjunto de treino. As teorias são as seguintes:

- teoria da consistência de processos de aprendizado;
- teoria não-assintótica da taxa de convergência de processos de aprendizado;
- teoria do controle da generalização de processos de aprendizado; e
- teoria da construção de algoritmos de aprendizado.

A ideia básica deste algoritmo consiste em mapear os vetores de atributos dos dados existentes no conjunto de treino em N dimensões diferentes, utilizando o chamado *truque de kernel*, conforme Russell e Norvig (2009). Com estes dados mapeados, a SVM cria uma separação linear em hiperplano entre estes dados, permitindo classificá-los.

O desempenho de um certo classificador (também chamado de função) no aprendizado sobre um conjunto de treino pode ser medido pelo risco empírico, dado por:

$$R_{emp}(h) = \frac{1}{n} \sum_{i=1}^n c(h(\mathbf{x}_i), y_i) \quad (2.17)$$

onde:

- h representa o classificador ou função em análise;
- $R_{emp}(h)$ = risco empírico associado ao classificador h ; e
- $c(h(\mathbf{x}_i), y_i)$ representa uma função de custo que relaciona a classe prevista para \mathbf{x}_i , dada por $h(\mathbf{x}_i)$, à classe esperada y_i , onde $y_i \in Y = \{-1, 1\}$.

A função de custo supracitada possui diversos tipos, e um deles, citado por [Faceli et al. \(2011\)](#) é o 0-1, definido pela Equação a seguir:

$$c(h(\mathbf{x}_i), y_i) = \frac{1}{2} |y_i - h(\mathbf{x}_i)| \quad (2.18)$$

cujas saída é 0, se a classificação foi a correta, e 1, caso contrário.

Para minimizar o erro do classificador sobre o conjunto de treinamento e sobre novos exemplos a serem classificados futuramente, utiliza-se o princípio de minimização do risco estrutural ([SMOLA; SCHÖLKOPF, 2004](#)), onde é feito um processo de indução para inferir uma função \hat{h} , com menor complexidade possível, que minimize este risco. Conforme [Müller et al. \(2001\)](#), pode-se estabelecer condições para garantir que o risco dos classificadores convirja para o risco esperado.

Para aplicação deste princípio, utiliza-se um limite para este risco esperado. Levando em consideração funções de decisão lineares, o risco esperado relaciona-se à margem de um determinado exemplo, como visto em [Smola e Schölkopf \(2004\)](#), que é uma medida de confiança da previsão de classes, dada pela distância de um exemplo até a fronteira de decisão definida pelo classificador em questão. Tal margem pode ser calculada pela Equação:

$$\varrho(h(\mathbf{x}_i), y_i) = y_i h(\mathbf{x}_i) \quad (2.19)$$

onde $\varrho(h(\mathbf{x}_i), y_i)$ representa a margem de classificação do exemplo \mathbf{x}_i , pela função h .

Com a equação acima, pode-se definir o risco marginal de h , que ainda de acordo com [Smola e Schölkopf \(2004\)](#), equivale à proporção de exemplos no conjunto de treino cuja mar-

gem de confiança está abaixo de um limiar $\rho > 0$. O cálculo do risco marginal pode ser visto na Equação a seguir:

$$R_p(h) = \frac{1}{n} \sum_{i=1}^n I(y_i h(\mathbf{x}_i) < \rho) \quad (2.20)$$

sendo que $I(q) = 1$, se q for uma condição verdadeira, e $I(q) = 0$, caso contrário.

Por fim, o limite para o risco esperado, considerando funções lineares e o risco marginal, pode ser dado por:

$$R(h) \leq R_p(h) + \sqrt{\frac{c}{n} \left(\frac{R^2}{\rho^2} \log^2 \left(\frac{n}{\rho} \right) + \log \left(\frac{1}{\theta} \right) \right)} \quad (2.21)$$

onde existe uma constante c para a qual esse limite acima se aplica, considerando:

- probabilidade de $1 - \theta \in [0, 1]$;
- para todo $\rho > 0$; e
- funções lineares $h(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x}$, com $\|\mathbf{x}\| \leq R$ e $\|\mathbf{w}\| \leq 1$.

Conforme [Faceli et al. \(2011\)](#), ao gerar um classificador utilizando SVMs, o hiperplano buscado deve ter margem ρ alta, com poucos erros marginais, sendo assim considerado um hiperplano ótimo.

Em relação à aplicação em problemas de classificação, existem dois tipos de SVMs, sendo as lineares e não-lineares, detalhadas a seguir.

2.3.3.1 SVMs lineares

As SVMs lineares são aquelas aplicadas, principalmente, a problemas linearmente separáveis, ou seja, onde o conjunto de dados pode ser dividido por meio de uma única reta (em espaços 2D), plano (3D) ou hiperplano (N dimensões). Em outras palavras, no caso de tarefas de classificação, como neste presente trabalho, utilizam-se SVMs lineares somente para classificação binária, entre as classes “positiva” e “negativa”.

Nesta versão das SVMs, introduzida em [Boser et al. \(1992\)](#) e [Cortes e Vapnik \(1995\)](#), as margens de separação entre os exemplos podem ser rígidas ou suaves, onde as margens rígidas são utilizadas em problemas linearmente separáveis e as margens suaves, por sua vez, são aplicadas a conjuntos de dados mais gerais.

Um hiperplano pode ser representado pela Equação a seguir:

$$h(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b \quad (2.22)$$

onde:

- \mathbf{w} representa o vetor normal ao hiperplano;
- \mathbf{x} representa um exemplo do conjunto de dados no espaço X ;
- $\mathbf{w} \cdot \mathbf{x}$ denota o produto escalar entre os dois vetores; e
- b é uma constante qualquer, pertencente aos reais (\mathfrak{R}).

Este hiperplano (ou reta/plano, dependendo da dimensão) divide o espaço de entrada dos dados (dado como X) em duas regiões, onde $\mathbf{w} \cdot \mathbf{x} > 0$ e $\mathbf{w} \cdot \mathbf{x} < 0$. Então, emprega-se uma função sinal, utilizada na classificação dos exemplos, dada pela Equação abaixo.

$$g(\mathbf{x}) = \text{sgn}(h(\mathbf{x})) = \begin{cases} +1, & \text{se } \mathbf{w} \cdot \mathbf{x} + b > 0 \\ -1, & \text{se } \mathbf{w} \cdot \mathbf{x} + b < 0 \end{cases} \quad (2.23)$$

Para este algoritmo, deve-se maximizar as margens de separação dos objetos x_i entre as possíveis classes, para melhor classificação destes. Esta maximização, em relação ao hiperplano $\mathbf{w} \cdot \mathbf{x} = 0$, é feita por meio da minimização de $\|\mathbf{w}\|$, pela Equação (2.24), como visto em [Campbell \(2001\)](#):

$$\underset{\mathbf{w}, b}{\text{Minimizar}} \quad \frac{1}{2} \|\mathbf{w}\|^2 \quad (2.24)$$

Para garantir as margens rígidas, impõe-se uma restrição, dada pela Equação (2.25), para garantir que não haja nenhum dado do conjunto de treino entre as margens de separação:

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 \geq 0, \forall i = 1, \dots, n \quad (2.25)$$

Este problema de otimização para minimizar $\|\mathbf{w}\|$ está na forma primal, e é quadrático. Existe outra forma para realizar tal otimização, conhecida como dual. A forma dual, por sua vez, utiliza apenas o conjunto de treino e seus rótulos, podendo ser obtida a partir da forma primal, ao se introduzir uma função lagrangiana e minimizá-la, chegando à Equação a seguir:

$$\underset{\alpha}{\text{Maximizar}} \quad \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j) \quad (2.26)$$

Para esta maximização, também são aplicadas restrições, vistas na Equação abaixo.

$$\begin{cases} \alpha_i \geq 0, \forall i = 1, \dots, n \\ \sum_{i=1}^n \alpha_i y_i = 0 \end{cases} \quad (2.27)$$

Por fim, levando em conta as formas primal e dual, ambas podem ser relacionadas para chegar ao classificador gerado pela SVM linear com margens rígidas.

Sendo α^* a solução para a otimização de forma dual, e \mathbf{w}^* e b^* as soluções para a forma primal, temos que todos os exemplos \mathbf{x}_i do conjunto cujos $\alpha_i^* > 0$ são considerados os vetores de suporte (*support vectors*, em inglês), pois de acordo com [Borges \(1998\)](#), estes são os exemplos com mais informações úteis para o treinamento do classificador.

Além disso, temos que \mathbf{w}^* pode ser definido por:

$$\mathbf{w}^* = \sum_{i=1}^n \alpha_i^* y_i \mathbf{x}_i \quad (2.28)$$

Enquanto que b^* pode ser definida por:

$$b^* = \frac{1}{n_{SV}} \sum_{\mathbf{x}_j \in SV} \frac{1}{y_j} - \mathbf{w}^* \cdot \mathbf{x}_j = \frac{1}{n_{SV}} \sum_{\mathbf{x}_j \in SV} \left(\frac{1}{y_j} - \sum_{\mathbf{x}_i \in SV} \alpha_i^* y_i \mathbf{x}_i \cdot \mathbf{x}_j \right) \quad (2.29)$$

onde:

- SV = conjunto dos *support vectors*;
- n_{SV} = número de *support vectors*; e
- y_i e y_j são as classes dos objetos x_i e x_j , respectivamente.

O valor de b^* é definido por α^* e pelas condições de Kühn-Tucker ([SMOLA; SCHÖLKOPF, 2004](#)), resumidas na Equação a seguir:

$$\alpha_i^* (y_i (\mathbf{w}^* \cdot \mathbf{x}_i + b^*) - 1) = 0, \forall i = 1, \dots, n \quad (2.30)$$

Logo, o classificador da SVM linear com margens rígidas se dá por:

$$g(\mathbf{x}) = \text{sgn}(h(\mathbf{x})) = \text{sgn} \left(\sum_{\mathbf{x}_i \in SV} \alpha_i^* y_i \mathbf{x}_i \cdot \mathbf{x} + b^* \right) \quad (2.31)$$

A SVM linear com margens suaves, por sua vez, é aplicado a problemas onde há ruído ou *outliers* presentes no conjunto de dados utilizado. Para lidar com estas situações, o classificador anterior é estendido com a adição de uma variável de folga, dada por ξ_i , onde as restrições impostas ao problema de otimização são relaxadas, permitindo que alguns erros de classifica-

ção aconteçam. Dada esta modificação, as restrições passam a ser representadas pela Equação (2.32), vista em [Smola e Schölkopf \(2004\)](#):

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad \forall i = 1, \dots, n \quad (2.32)$$

Para margens suaves, também têm-se os problemas nas formas primal e dual.

Na forma primal, o erro sobre o conjunto de treino pode ser minimizado ao levar em conta a soma total dos ξ_i . De acordo com [Burges \(1998\)](#), o problema de otimização primal pode ser reformulado, conforme a seguinte Equação:

$$\underset{\mathbf{w}, b, \xi}{\text{Minimizar}} \quad \frac{1}{2} \|\mathbf{w}\|^2 + C \left(\sum_{i=1}^n \xi_i \right) \quad (2.33)$$

onde:

- $\xi_i > 1$ indica um erro no conjunto de treino;
- $\sum_{i=1}^n \xi_i$ minimiza erros marginais, uma vez que se $0 < \xi_i \leq 1$, o exemplo \mathbf{x}_i está entre as margens de separação; e
- C é uma constante inserida para dar peso maior à minimização dos erros no conjunto de treino, em relação à minimização da complexidade do classificador obtido.

Assim como na versão com margens rígidas, o problema na forma primal é quadrático. Da mesma forma, pode-se introduzir uma função lagrangiana e chegar à forma dual para margens suaves, vista na Equação abaixo:

$$\underset{\alpha}{\text{Maximizar}} \quad \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j) \quad (2.34)$$

Também há restrições aplicadas à otimização na forma dual, dadas por:

$$\begin{cases} 0 \leq \alpha_i \leq C, \forall i = 1, \dots, n \\ \sum_{i=1}^n \alpha_i y_i = 0 \end{cases} \quad (2.35)$$

No caso das SVMs lineares com margens suaves, sendo α^* a solução do problema dual, e \mathbf{w}^* , b^* e ξ^* as soluções do problema primal, temos que \mathbf{w}^* é determinado ainda pela Equação (2.28), e ξ^* é definido pela Equação (2.36), a seguir:

$$\xi_i^* = \max \left\{ 0, 1 - y_i \sum_{j=1}^n y_j \alpha_j^* \mathbf{x}_j \cdot \mathbf{x}_i + b^* \right\} \quad (2.36)$$

E b^* continua sendo definida por α^* , mas agora, também é definida por outras condições de K uhn-Tucker, dadas por:

$$\alpha_i^*(y_i(\mathbf{w}^* \cdot \mathbf{x}_i + b^*) - 1 + \xi_i^*) = 0, \forall i = 1, \dots, n \quad (2.37)$$

E por:

$$(C - \alpha_i^*)\xi_i^* = 0 \quad (2.38)$$

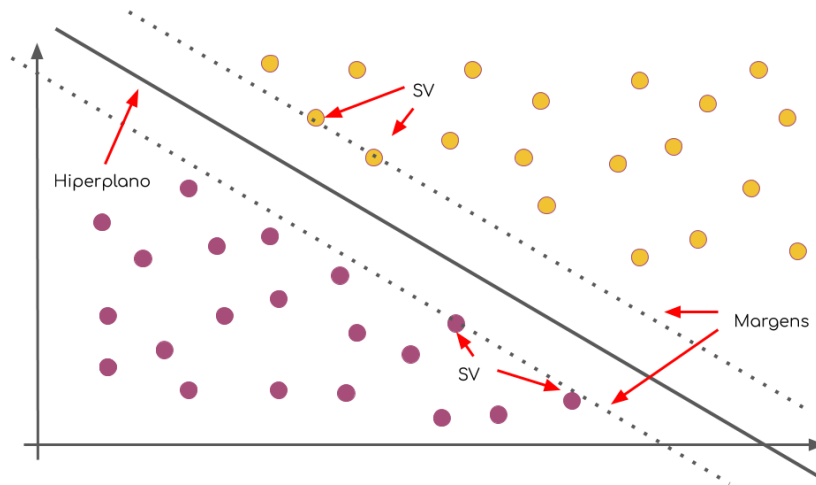
Para SVMs com margens suaves, os exemplos \mathbf{x}_i com $\alpha_i^* > 0$ continuam sendo os *support vectors*, mas estes podem ser divididos em dois tipos:

- **SVs livres**, sendo aqueles com $\alpha_i^* < C$, que est o sobre as margens de separa  o;
- **SVs limitados**, onde $\alpha_i^* = C$, subdividindo-se em:
 - Erro de classifica  o, se $\mathbf{x}_i^* > 1$;
 - Exemplo classificado corretamente e entre as margens, se $0 < \mathbf{x}_i^* \leq 1$; e
 - Exemplo sobre as margens de separa  o, se $\mathbf{x}_i^* = 0$.

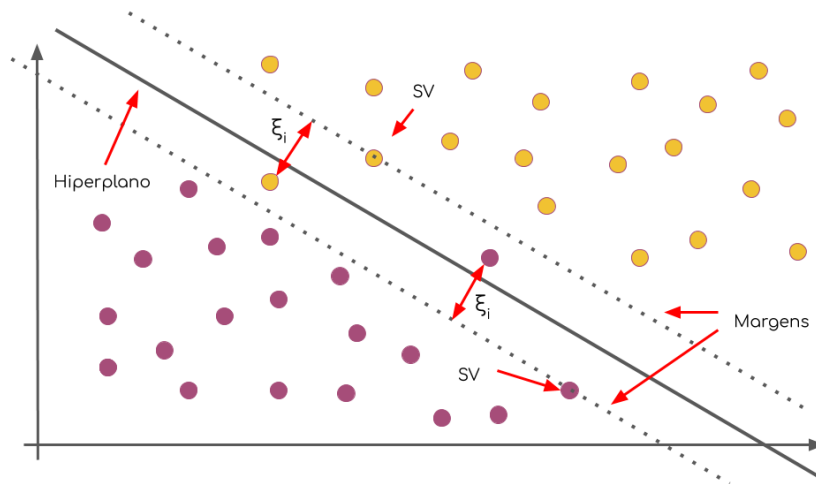
Como visto em [Faceli et al. \(2011\)](#), o valor de b^*   calculado por meio da Equa  o (2.29), considerando todos os *support vectors* cujo $\alpha_i^* < C$. Por fim, o classificador da SVM linear com margens suaves   dado pela mesma equa  o do classificador com margens r gidas (vide Equa  o (2.31)), exceto pelo fato de que o valor de α_i^*   calculado pela forma dual espec fica para margens suaves, vista na Equa  o (2.34).

Ilustra  es dos SVMs lineares abordados nesta se  o podem ser vistas na Figura 10.

Figura 10: SVMs lineares



(a) Margem rígida



(b) Margem suave

Fonte: a autora. Adaptado de [Faceli et al. \(2011\)](#).

2.3.3.2 SVMs não-lineares

Ao contrário das SVMs lineares, as não-lineares podem ser aplicadas a problemas de classificação mais gerais, uma vez que os separadores não são necessariamente retas ou planos, captando mais precisamente as nuances de cada exemplo, permitindo sua classificação entre diversas classes presentes no problema analisado, não somente “positiva” ou “negativa”.

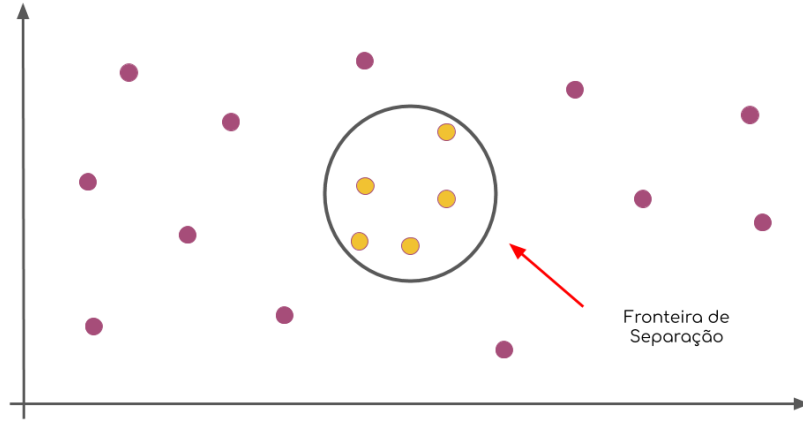
De acordo com [Hearst et al. \(1998\)](#), para lidar com problemas não lineares, faz-se um mapeamento do espaço de entradas X em um espaço de características \mathfrak{S} , denotado por $\Phi : X \rightarrow \mathfrak{S}$.

Este mapeamento é embasado pela teoria de Cover, vista em [Haykin \(1994\)](#), que diz que se a transformação feita for não linear, e a dimensão do espaço de características for suficientemente alta, então X pode ser transformado em \mathfrak{S} , de modo que há alta probabilidade dos

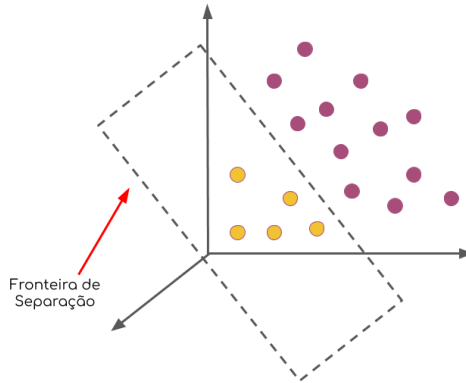
objetos em \mathfrak{S} serem linearmente separáveis, ou seja, o conjunto de dados mapeado neste espaço de características pode ser separado por uma SVM linear, ao determinar um mapeamento Φ adequado.

Um exemplo de mapeamento pode ser visto na Figura 11.

Figura 11: SVMs não-lineares



(a) Dados em espaço 2D, durante o mapeamento



(b) Dataset transformado para 3D, agora linearmente separável

Fonte: a autora. Adaptado de [Faceli et al. \(2011\)](#).

O processo de uma SVM não-linear, segundo [Faceli et al. \(2011\)](#), envolve mapear os objetos de um conjunto de dados em um espaço de dimensão maior, utilizando uma transformação Φ . Posteriormente, é feita a aplicação de uma SVM linear com margens suaves, que encontra o hiperplano com maior margem de separação.

O mapeamento dos objetos é aplicado tanto ao problema de otimização das SVMs de margens suaves, quanto ao cálculo de b^* , de forma que a otimização se dá por:

$$\underset{\alpha}{\text{Maximizar}} \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j y_i y_j (\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)) \quad (2.39)$$

O cálculo de b^* é dado por:

$$b^* = \frac{1}{n_{SV} : \alpha^* < C} \sum_{\mathbf{x}_j \in SV : \alpha_j^* < C} \left(\frac{1}{y_j} - \sum_{\mathbf{x}_i \in SV} \alpha_i^* y_i \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j) \right) \quad (2.40)$$

E finalmente, o classificador do SVM não-linear se dá por:

$$g(\mathbf{x}) = \text{sgn}(h(\mathbf{x})) = \text{sgn} \left(\sum_{\mathbf{x}_i \in SV} \alpha_i^* y_i \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}) + b^* \right) \quad (2.41)$$

Uma vez que o custo do cálculo de Φ pode ser muito alto, devido à dimensão do espaço de características, utiliza-se uma função *kernel* para obter apenas a informação necessária, a respeito dos mapeamentos, para que o processo de treino e classificação das SVMs sejam concluídas. Este é o *truque de kernel* mencionado por [Russell e Norvig \(2009\)](#).

Conforme [Herbrich \(2002\)](#), um *kernel* recebe dois exemplos do espaço de entradas X , sendo \mathbf{x}_i e \mathbf{x}_j , e calcula o produto escalar entre ambos no espaço de características, como visto na Equação a seguir:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j) \quad (2.42)$$

Existem diversos tipos de *kernels*, com três sendo os mais usados, como destacado por [Faceli et al. \(2011\)](#):

- **Polinomial**, dado por $(\delta(\mathbf{x}_i \cdot \mathbf{x}_j) + \kappa)^d$, que implica na obtenção de uma SVM linear, sem mapeamento dos dados;
- **Radial Basis Function (RBF)**, dado por $\exp(-\sigma \|\mathbf{x}_i - \mathbf{x}_j\|^2)$; e
- **Sigmoidal**, dado por $\tanh(\delta(\mathbf{x}_i \cdot \mathbf{x}_j) + \kappa)$.

Portanto, um classificador obtido por SVMs exige a definição de uma função *kernel*, de seus parâmetros e da constante C , que possuem grande influência sobre o desempenho do classificador.

Por fim, considerando a função *kernel*, uma versão geral do classificador da SVM não linear é dada por:

$$g(\mathbf{x}) = \text{sgn}(h(\mathbf{x})) = \text{sgn} \left(\sum_{\mathbf{x}_i \in SV} \alpha_i^* y_i K(\mathbf{x}_i, \mathbf{x}) + b^* \right) \quad (2.43)$$

2.3.3.3 Vantagens e desvantagens

A SVM possui uma flexibilidade para escolher o formato do separador, é tolerante a ruídos ([HAYKIN, 1994](#)), além de ser um algoritmo com grande capacidade de generalização. Também é um algoritmo eficiente, por conta da aplicação de funções *kernel*, que aceleram a construção dos hiperplanos no espaço de características, como visto em [Burges \(1998\)](#).

Dentre as desvantagens, tem-se a falta de transparência dos resultados gerados, além da sensibilidade à função *kernel* escolhida e aos valores dos parâmetros utilizados ([AURIA; MORO, 2008](#)). Também menciona-se em [Burges \(1998\)](#) e [Osuna e Girosi \(1999\)](#) que as SVMs podem ter um tempo de execução e uso de memória muito grandes, dependendo do caso de aplicação, apesar da boa performance na generalização.

2.3.4 Árvores de Decisão e *Random Forest*

Tanto a Árvore de Decisão como o *Random Forest* são métodos baseados na procura por uma solução dentro de um espaço de possíveis respostas a um dado problema, e existem diversos algoritmos que fazem uso de Árvores de Decisão, como CART, ID3 e C4.5, por exemplo.

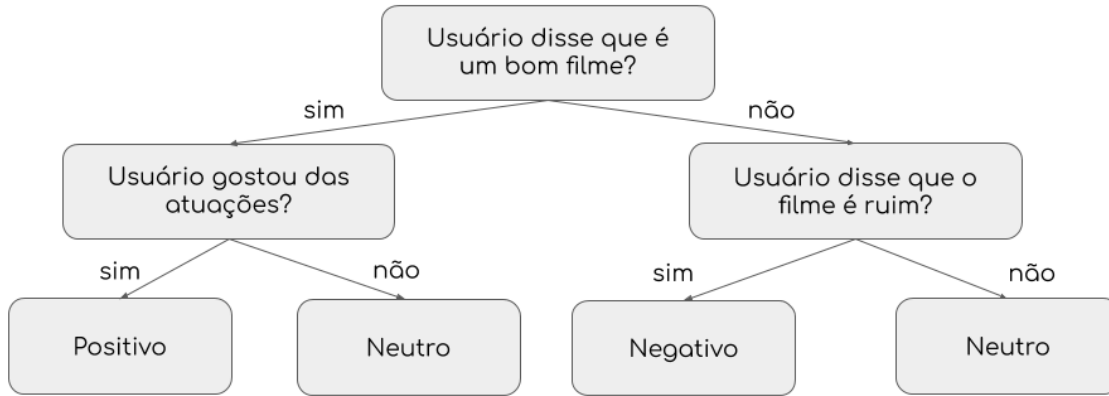
2.3.4.1 Árvores de Decisão

Uma Árvore de Decisão representa uma função que recebe um vetor de atributos de um exemplo, e retorna um resultado único - uma “decisão” sobre a classificação desse exemplo - após realizar uma sequência de testes, utilizando a abordagem de Divisão e Conquista para classificar qualquer tipo de entrada ([RUSSELL; NORVIG, 2009](#); [FACELI et al., 2011](#)). Nesta abordagem de Divisão e Conquista, um problema é dividido em subproblemas mais simples, sendo cada um resolvido recursivamente, permitindo uso de diferentes modelos em cada subproblema. Com estes solucionados, a solução do problema principal se dá pela combinação da solução de todos os subproblemas.

A árvore utilizada é um grafo acíclico e direcionado, onde os nós intermediários (conhecidos como *nós de divisão*) contêm um teste condicional univariado - baseado nos valores de um único atributo por vez -, e os nós folha contêm as classificações possíveis para o exemplo analisado. No contexto de Divisão e Conquista, cada subproblema a ser resolvido se dá por um teste condicional de algum nó intermediário, e o conjunto de respostas geradas por cada teste condicional gera a classificação final do exemplo.

Uma ilustração da aplicação de Árvores de Decisão pode ser vista na Figura 12.

Figura 12: Ilustração de Árvores de Decisão



Fonte: a autora.

Para a definição de quais atributos serão utilizados em cada um dos testes dos nós intermediários, utilizam-se as chamadas *regras de divisão*. Cada atributo possui uma medida de “*goodness of split*”, ou seja, o quão bem ele pode dividir os exemplos entre classes distintas, e a regra de divisão seleciona o atributo com a maior medida.

Cada algoritmo utiliza uma métrica diferente para o “*goodness of split*” como base de suas regras de divisão, dentre várias métricas já apresentadas na literatura, e duas delas são o *ganho de informação* e o *índice Gini*.

2.3.4.2 Métricas para regras de divisão

Em uma Árvore de Decisão, pode-se calcular a entropia de um atributo ou conjunto, que mede o seu grau de pureza. Em outras palavras, mede-se o nível de dificuldade para se prever o valor de um atributo, ou de um conjunto.

O cálculo de entropia é feito por meio da Equação (2.44), vista em [Du e Zhan \(2002\)](#):

$$Entropia(S) = - \sum_{j=1}^m P_j \log P_j \quad (2.44)$$

Onde:

- S representa o conjunto de exemplos utilizado no cálculo;

- m = número de classes presentes no *dataset*; e
- P_j = frequência relativa da classe j no conjunto S .

Note que os valores de um certo atributo dividem um *dataset* em partições diferentes. Dado o conjunto completo de exemplos e estas partições, pode-se calcular o *ganho de informação* com o uso de um certo atributo nas divisões, representado pela diferença entre a entropia do *dataset* completo e a soma ponderada das entropias destas partições, conforme a Equação:

$$Ganho(S, A) = Entropia(S) - \sum_{v \in A} \left(\frac{|S_v|}{|S|} \times Entropia(S_v) \right) \quad (2.45)$$

em que:

- A representa o atributo utilizado para partição de S , e cujo ganho de informação está sendo analisado;
- v representa um dos possíveis valores de A ; e
- S_v representa um subconjunto de exemplos, onde o atributo A possui valor v .

Por fim, o atributo selecionado pela regra de decisão será aquele que gerar o maior ganho de informação, dentre todos os atributos analisados naquele momento da construção da Árvore de Decisão. Segundo [Faceli et al. \(2011\)](#), esta métrica é utilizada em regras de decisão do algoritmo C4.5, por exemplo.

O índice Gini, por outro lado, faz o contrário do cálculo do ganho de informação - calcula o grau de impureza de um atributo ou conjunto. Com esta métrica, o atributo selecionado é aquele que resulta na máxima redução de impureza, segundo [Faceli et al. \(2011\)](#).

Seu cálculo se dá por:

$$Gini(S) = 1 - \sum_{j=1}^m P_j^2 \quad (2.46)$$

sendo:

- m = número de classes presentes no *dataset*; e
- P_j = frequência relativa da classe j no conjunto S .

Esta medida também pode ser usada para calcular ganho de informação, no lugar da entropia, vista anteriormente (DU; ZHAN, 2002). Neste caso, o cálculo do ganho de informação se dá pela Equação (2.47), vista abaixo.

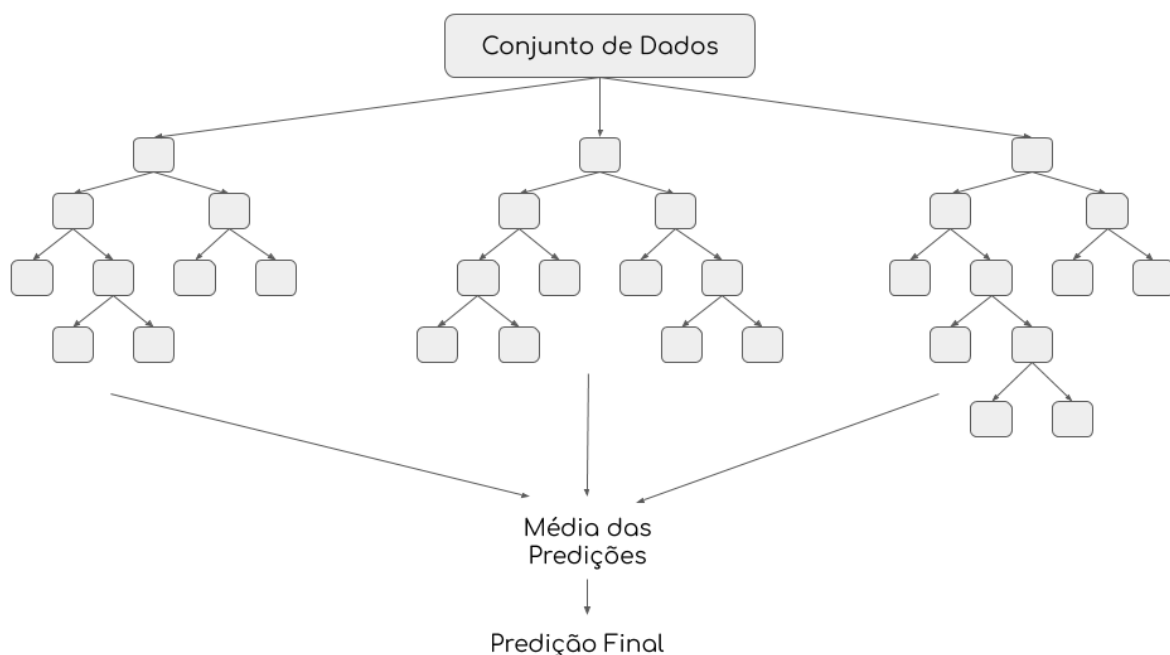
$$Ganho(S, A) = Gini(S) - \sum_{v \in A} \left(\frac{|S_v|}{|S|} \times Gini(S_v) \right) \quad (2.47)$$

2.3.4.3 Random Forest

O algoritmo *Random Forest*, por sua vez, é um método que pode ser utilizado tanto para tarefas de classificação como de regressão (LIAW; WIENER; MATHE, 2004). Introduzido por Breiman (2001), trata-se de uma combinação de diversas Árvores de Decisão, onde cada uma recebe um subconjunto aleatório de todos os atributos do conjunto de dados, e é construída com regras de divisão em cima deste subconjunto.

Este algoritmo utiliza o método *Bagging*, onde diversas versões de um preditor são geradas, e o resultado final consiste em uma combinação dos resultados de cada preditor. No caso do *Random Forest* para classificação, há uma “votação”, onde a classe mais votada é a escolhida para determinado exemplo analisado (BREIMAN, 1996). Uma ilustração do funcionamento do *Random Forest* pode ser visto na Figura 13.

Figura 13: Ilustração do funcionamento do *Random Forest*



Fonte: a autora.

2.3.4.4 Vantagens e desvantagens

Árvores de Decisão podem apresentar o problema de *overfitting*, caracterizado pelo uso de modelos que utilizam mais atributos e/ou abordagens mais complexas do que o necessário, isto é, ocorre um ajuste exagerado ao conjunto de dados (HAWKINS, 2004). Para solucionar este problema, o próprio algoritmo Random Forest é adotado (BREIMAN, 2001), além da realização do processo de poda destas árvores.

As Árvores de Decisão (e *Random Forest*, por consequência) são flexíveis - uma vez que também envolvem um método não-paramétrico -, robustas e eficientes, além de serem conhecidas por sua interpretabilidade. Por outro lado, são instáveis e bastante afetadas pela replicação de testes em diferentes ramos da árvore, e também pela ausência de valores de atributos a serem analisados.

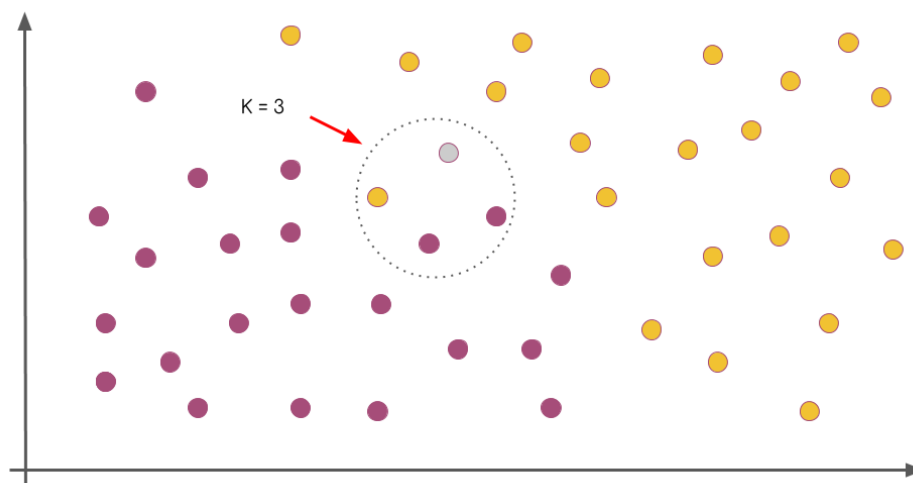
2.3.5 *k*-Nearest Neighbours (k-NN)

O algoritmo dos k vizinhos mais próximos envolve classificar um exemplo ainda não rotulado, com base nas classes de seus vizinhos.

Dado um objeto, representando o exemplo a ser classificado, o algoritmo calcula a distância entre este objeto e cada uma das outras entradas do conjunto de dados, como visto em Faceli et al. (2011), e seleciona os k vizinhos mais próximos, obtendo a classe de cada um deles. Posteriormente, o algoritmo determina qual é a classe predominante entre estes k vizinhos, e classifica o exemplo não rotulado como sendo desta classe.

Um exemplo de aplicação do algoritmo pode ser visto na Figura 14, a seguir.

Figura 14: Ilustração do algoritmo *k*-Nearest Neighbours (k-NN)



Fonte: a autora.

Dentre os diversos cálculos de distância utilizados neste algoritmo, o mais comum é o da distância euclidiana, dado por:

$$d(x, y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2} \quad (2.48)$$

O número k , definido pelo usuário, frequentemente é um número ímpar e pequeno, como $k = 3$ ou $k = 5$. De acordo com a literatura, para determinar k , pode-se estimá-lo por validação cruzada ou atribuir um peso aos vizinhos de um dado exemplo a ser classificado (FACELI et al., 2011), sendo este peso dado por:

$$w_i = \frac{1}{d(x_t, x_i)} \quad (2.49)$$

sendo w_i o peso atribuído ao vizinho x_i , na classificação do exemplo x_t .

O k-NN é um algoritmo considerado simples, já que o aprendizado consiste apenas em armazenar os dados do conjunto, além de ser aplicável a problemas complexos. Por outro lado, justamente pela simplicidade no aprendizado, exige grande espaço de armazenamento, além do desempenho ser afetado pelo cálculo de distância e valor de k , ambos sendo parâmetros determinados pelo usuário (KOTSIANTIS, 2007). Além disso, este algoritmo não obtém uma representação compacta dos exemplos que classifica e é afetado, também, pela presença de atributos redundantes e irrelevantes no conjunto de dados.

2.4 Redes neurais

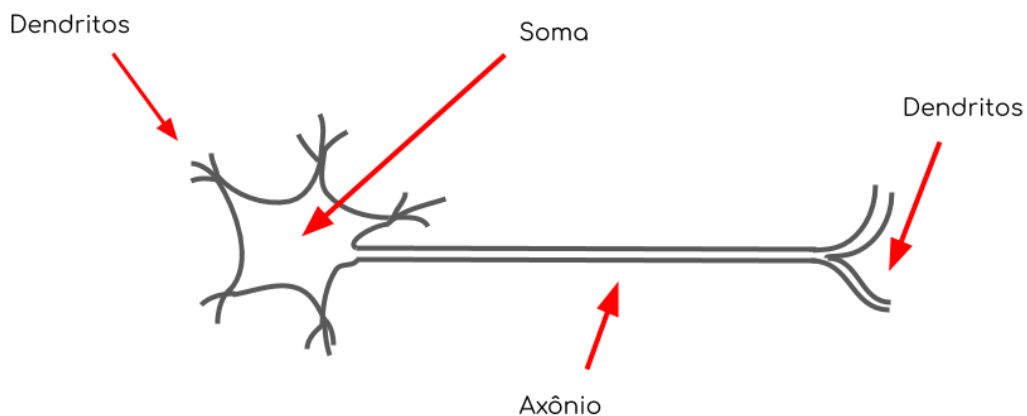
Assim como *Support Vector Machines* (SVMs), detalhadas na Seção 2.3.3, redes neurais artificiais (RNAs) são um método de aprendizado de máquina baseado na otimização de uma função objetivo, com inspiração no sistema nervoso humano, buscando simular nossa capacidade de aprendizado (FACELI et al., 2011).

Dentre as principais características das RNAs:

A capacidade de *aprender* através de exemplos e de *generalizar* a informação aprendida é, sem dúvida, o atrativo principal da solução de problemas através de RNAs [...]. As RNAs são capazes de extrair informações não-apresentadas de forma explícita através dos exemplos. Não obstante, as RNAs são capazes de atuar como mapeadores universais de funções multivariáveis, com custo computacional que cresce apenas linearmente com o número de variáveis. Outra característica importante é a capacidade de auto-organização e de processamento temporal, que, aliada àquelas citadas anteriormente, faz das RNAs uma ferramenta computacional extremamente poderosa e atrativa para a solução de problemas complexos (BRAGA; LUDERMIR; CARVALHO, 2000).

Na Biologia, os neurônios, células que compõem o sistema nervoso, são responsáveis por transmitir impulsos nervosos e responder a estímulos internos e externos. Estes neurônios são compostos de soma, dendritos e axônio. Uma ilustração da estrutura de um neurônio pode ser vista na Figura 15.

Figura 15: Estrutura de um neurônio natural



Fonte: a autora.

Nesta estrutura retratada, os dendritos recebem estímulos de outros neurônios ou do ambiente, sendo estes repassados ao soma, que é responsável por combinar e transmitir as informações processadas ao axônio. O axônio, por sua vez, conduz os impulsos elétricos produzidos até outros neurônios, ou outro local distante, segundo [Faceli et al. \(2011\)](#). Este impulso elétrico, por fim, é transmitido de um neurônio a outro por meio da sinapse, que é uma unidade mediadora das interações - excitatórias ou inibitórias - entre neurônios, dada pelo contato entre o axônio de um neurônio e os dendritos do outro ([HAYKIN, 1994](#)).

Já no conceito de RNAs, os neurônios artificiais são as unidades de processamento que computam funções matemáticas, dispostas em uma ou mais camadas, estando interligadas por várias conexões (simulações das sinapses biológicas), na grande maioria das vezes, unidirecionais, e que possuem um peso associado a elas, sendo positivo ou negativo, dependendo do tipo de interação. Segundo [Braga, Ludermir e Carvalho \(2000\)](#), estes pesos são ajustados durante o aprendizado e são o que codificam o conhecimento que a RNA adquire ao longo deste processo.

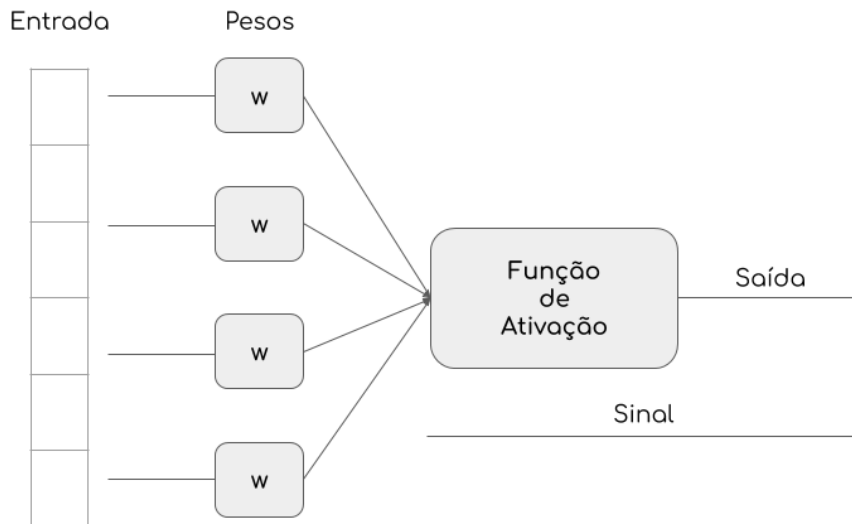
A saída deste neurônio, por sua vez, é dada pelo uso de uma função de ativação sobre a entrada total recebida, podendo ser uma função sigmoideal, linear, limiar, entre outras. Esta fun-

ção de ativação é responsável por restringir a amplitude da saída deste neurônio a um intervalo $[0, 1]$, ou em alguns casos, $[-1, 1]$.

De acordo com [Haykin \(1994\)](#), um modelo de neurônio pode ser não-linear, possuindo uma aplicação externa de *bias* que aumenta ou diminui a entrada líquida da função de ativação, dependendo se este *bias* é positivo ou negativo.

O processo de aprendizado da rede neural depende não só de sua arquitetura - como quantidade de neurônios e suas conexões -, mas também das regras de aprendizado, para ajustar os pesos destas conexões entre os neurônios dela, ponderando as entradas recebidas. A arquitetura básica de um neurônio artificial pode ser dada pela vista na Figura 16.

Figura 16: Arquitetura de um neurônio artificial



Fonte: a autora. Adaptado de [Faceli et al. \(2011\)](#).

Em comparação com um neurônio natural, os terminais de entrada de dados no neurônio representam os dendritos, que recebem os valores iniciais do algoritmo ou de outros neurônios. A entrada total recebida pelo neurônio se dá por:

$$u = \sum_{j=1}^d x_j w_j \quad (2.50)$$

onde:

- u = entrada total do neurônio;

- d representa o total de atributos de um objeto \mathbf{x} , e também o número de terminais de entrada;
- $x_j, 1 \leq j \leq d$ representa o atributo j do objeto \mathbf{x} ; e
- $w_j, 1 \leq j \leq d$ representa o peso j do vetor de pesos \mathbf{w} , aplicado ao terminal de entrada j .

Enquanto isso, o soma é representado por uma função matemática de ativação φ , que pondera e combina as entradas, sendo que a resposta do neurônio àquela entrada se dá pela saída desta função de ativação.

A saída de um neurônio, portanto, é representada por:

$$y = \varphi(u + b) \quad (2.51)$$

sendo:

- y = saída do neurônio;
- φ representando a função de ativação;
- u = entrada total do neurônio; e
- b = *bias* aplicado externamente.

Outro conceito importante é o potencial de ativação do neurônio em sua rede neural, dada por:

$$v = u + b \quad (2.52)$$

sendo v o potencial de ativação deste neurônio.

Como mencionado anteriormente, existem diversas funções de ativação que podem ser usadas, com destaque para linear, limiar e sigmoidal, onde:

- **linear:** a saída da função é o próprio valor de u .
- **limiar:** dado um limiar estabelecido pelo usuário, tem-se o modelo de McCulloch-Pitts, em que:

$$\varphi = \begin{cases} 1, & \text{se entrada total } u > \text{limiar} \\ 0, & \text{caso contrário} \end{cases} = \begin{cases} 1, & \text{se potencial de ativação } v \geq 0 \\ 0, & \text{caso contrário} \end{cases}$$

Onde para a saída do neurônio ser 1, quanto maior o limiar, maior deve ser a entrada total (MCCULLOCH; PITTS, 1943). Caso o intervalo considerado para a saída do neurônio seja $[-1, 1]$, a função limiar pode ser representada por uma função sinal:

$$\varphi(v) = \begin{cases} 1, & \text{se } v > 0 \\ 0, & \text{se } v = 0 \\ -1, & \text{se } v < 0 \end{cases}$$

- **sigmoidal:** aproximação contínua da função limiar, sendo a mais utilizada na literatura, e que pode ser representada, por exemplo, pela função logística, onde:

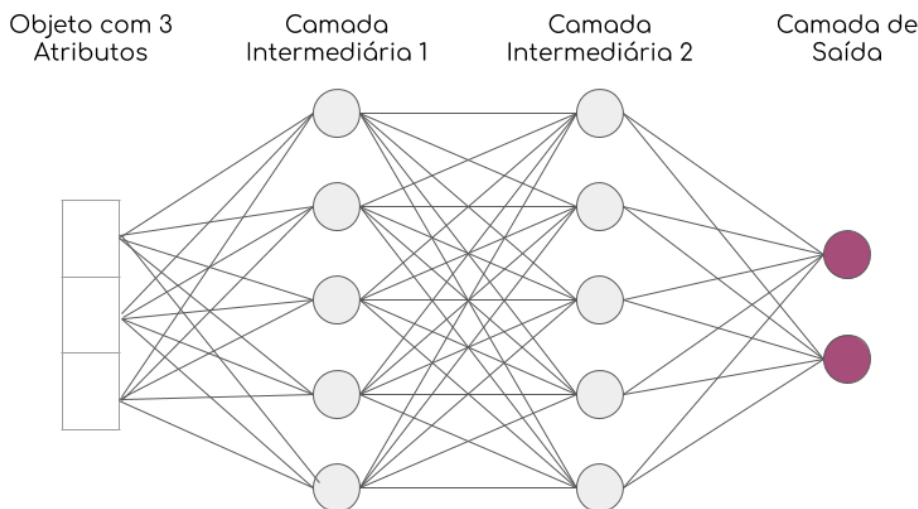
$$\varphi(v) = \frac{1}{1 + \exp(-av)}$$

Sendo a o parâmetro de inclinação desta sigmoide. Assim como na função limiar, se o intervalo da saída for $[-1, 1]$, então esta função pode ser representada por:

$$\varphi(v) = \tanh v$$

Uma rede neural pode ter uma ou mais camadas de neurônios, onde um neurônio pode tanto receber sua entrada de um neurônio da camada anterior, quanto enviar sua saída para outro neurônio na próxima. Existem as camadas intermediárias (também chamadas de *ocultas*), responsáveis pelo aprendizado, e uma de saída, que gera os valores de saída do algoritmo, como pode ser visto na Figura 17.

Figura 17: Exemplo de rede neural com várias camadas



Fonte: a autora. Adaptado de Faceli et al. (2011).

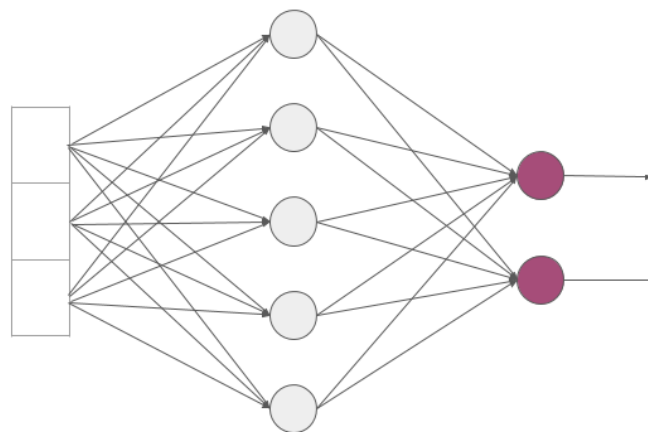
Conforme [Faceli et al. \(2011\)](#), existem três padrões de conectividade entre os neurônios de uma rede multicamadas, sendo estes:

- **completa:** todo neurônio está conectado a todos os neurônios da camada anterior ou seguinte a ele;
- **local:** o neurônio se conecta a alguns neurônios da camada anterior ou seguinte a ele, sendo que estes neurônios ficam em uma região definida dentro da camada; e
- **parcial:** o neurônio se conecta a alguns neurônios da camada anterior ou seguinte a ele, não necessariamente da mesma região.

No geral, existem três arquiteturas de rede neural, cada uma com suas particularidades, como visto em [Haykin \(1994\)](#), sendo a rede neural *feedforward* de camada única, *feedforward* multicamadas e recorrente.

Na rede *feedforward* de camada única, temos uma camada de entrada dos dados, que se projeta sobre uma camada de saída, onde é feito o processamento dessas entradas para emitir a saída do algoritmo. A alimentação é feita somente da entrada em direção à saída, sendo, por este motivo, chamada de *feedforward*. Esta arquitetura pode ser vista na Figura 18.

Figura 18: Arquitetura de uma RNA *feedforward* de camada única



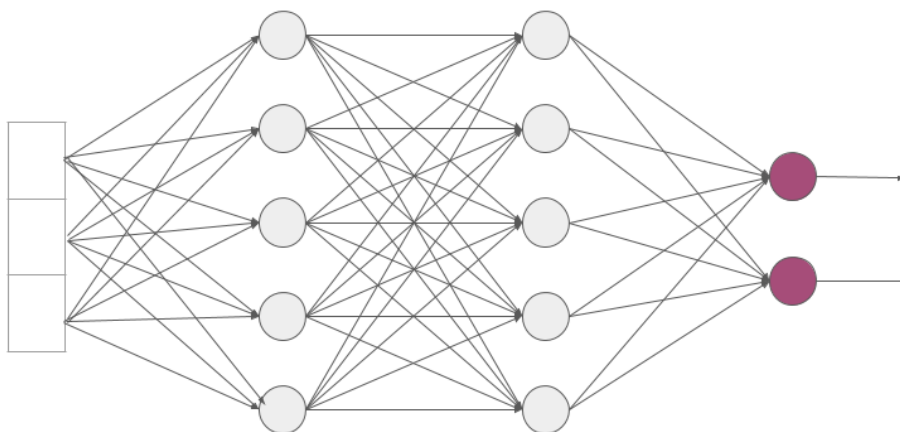
Fonte: a autora. Adaptado de [Faceli et al. \(2011\)](#).

A rede *feedforward* multicamadas, por sua vez, possui uma ou mais camadas ocultas entre as camadas de entrada e saída da rede neural, permitindo que a rede tenha uma perspectiva

global, por conta das conexões sinápticas e dimensão das interações neurais extras (CHURCHLAND; SEJNOWSKI, 1992).

Nesta arquitetura, a primeira camada recebe os dados como entrada, enquanto que cada uma das camadas subsequentes recebem, como entrada, a saída da camada anterior. A arquitetura multicamadas é ilustrada na Figura 19.

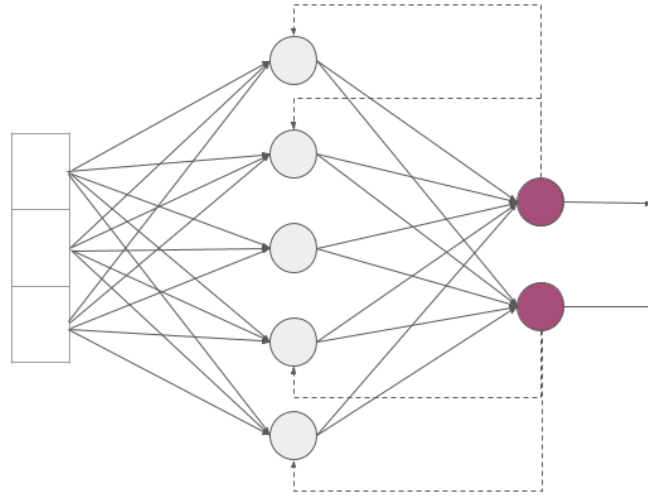
Figura 19: Arquitetura de uma RNA *feedforward* multicamadas



Fonte: a autora. Adaptado de Faceli et al. (2011).

Por fim, a última arquitetura é a recorrente, que se distingue das anteriores pela presença de retroalimentação, onde um neurônio pode receber informações de outros neurônios da mesma camada, ou de camadas posteriores. Também há a possibilidade de um neurônio ser alimentado por sua própria saída. Esta arquitetura pode ser visualizada na Figura 20.

Figura 20: Arquitetura de uma RNA recorrente



Fonte: a autora. Adaptado de [Faceli et al. \(2011\)](#).

Quanto ao aprendizado da RNA, existem quatro grupos de algoritmos de treinamento, para definir os pesos associados às conexões, de modo que o desempenho da rede seja melhorado, sendo eles o de correção de erro, de algoritmos de aprendizado Hebbiano, de algoritmos de aprendizado competitivo e o de máquinas de Boltzmann ([FACELI et al., 2011](#)). Cada algoritmo possui um conjunto de regras com especificações de quando e como devem ser feitas as alterações de peso e de seus *biases*.

Conforme [Haykin \(1994\)](#), o processo de aprendizagem implica em uma sequência de eventos, onde a rede neural é estimulada pelo ambiente, tem seus parâmetros alterados por conta do estímulo e, posteriormente, passa a responder ao ambiente de modo diferente, por conta destas alterações.

O primeiro grupo de algoritmos, como mencionado anteriormente, é o de correção de erro, aplicado principalmente a algoritmos de aprendizado supervisionado. Neste grupo, quando um neurônio na camada de saída emite o resultado obtido por ele, este é comparado com a saída desejada para o exemplo analisado, gerando um sinal de erro. Este sinal aciona o mecanismo de ajuste dos pesos dos neurônios desta RNA, procurando aproximar a saída emitida da esperada, a cada passo de tempo.

O sinal de erro pode ser dado pela Equação a seguir:

$$e_k(n) = d_k(n) - y_k(n) \quad (2.53)$$

onde:

- $e_k(n)$ representa o sinal de erro associado ao neurônio k , no instante n do processamento do aprendizado;
- $d_k(n)$ representa a saída esperada para o neurônio k ; e
- $y_k(n)$ representa a saída emitida pelo neurônio k .

O ajuste aplicado ao peso, por sua vez, se dá pela regra de Widrow-Hoff ([WIDROW; HOFF, 1960](#)):

$$\Delta w_{kj}(n) = \eta e_k(n) x_j(n) \quad (2.54)$$

sendo que:

- η representa uma taxa de aprendizado;
- $x_j(n)$ representa o exemplo classificado pela rede neural; e
- $\Delta w_{kj}(n)$ representa o ajuste aplicado ao peso w_{kj} no instante n .

Com o ajuste calculado, o novo peso atribuído ao neurônio é dado por:

$$w_{kj}(n+1) = w_{kj}(n) + \Delta w_{kj}(n) \quad (2.55)$$

O segundo grupo engloba os algoritmos de aprendizado Hebbiano, baseado na regra de Hebb ([HEBB, 1949](#)), que diz:

Quando um axônio da célula A está perto o suficiente para excitar uma célula B e participa do seu disparo repetida ou persistentemente, então algum processo de crescimento ou modificação metabólica acontece em uma das células ou ambas, de tal forma que a eficiência de A como uma das células que dispara B é aumentada.

Em outras palavras, conforme [Stent \(1973\)](#) e [Changeux e Danchin \(1976\)](#), se ambos os neurônios de uma sinapse - conhecida como *sinapse hebbiana* - são ativados simultaneamente, a conexão entre eles é reforçada. Caso contrário, é enfraquecida ou eliminada. A sinapse hebbiana é caracterizada por um mecanismo dependente do tempo, do local (onde a modificação da sinapse é específica para a entrada do neurônio) e é fortemente interativo ([BROWN, 1990](#)).

O ajuste aplicado ao peso sináptico pode ter diversas formas, segundo [Haykin \(1994\)](#). A primeira delas é a Hipótese de Hebb, semelhante à regra de Widrow-Hoff, dada por:

$$\Delta w_{kj}(n) = \eta y_k(n) x_j(n) \quad (2.56)$$

onde:

- $x_j(n)$ representa o sinal pré-sináptico do neurônio k ; e
- $y_k(n)$ representa o sinal pós-sináptico.

Porém, de acordo com a Hipótese de Hebb, com a aplicação constante do sinal de entrada x_j , aumenta-se o sinal y_k exponencialmente, saturando a conexão. Como alternativa, outra forma de ajuste é a hipótese da covariância (SEJNOWSKI, 1977a; SEJNOWSKI, 1977b). Com esta hipótese, os sinais de entrada e saída são substituídos pelos seus desvios em relação aos seus valores médios, em um dado intervalo de tempo. O ajuste, portanto, é dado por:

$$\Delta w_{kj}(n) = \eta(x_j - \bar{x})(y_k - \bar{y}) \quad (2.57)$$

onde \bar{x} e \bar{y} representam os valores médios dos sinais de entrada e de saída, respectivamente.

Nesta hipótese, a conexão é reforçada se $x_j > \bar{x}$ e $y_k > \bar{y}$, e é enfraquecida se:

- $x_j > \bar{x}$ e $y_k < \bar{y}$; ou
- $x_j < \bar{x}$ e $y_k > \bar{y}$.

O terceiro grupo é composto por algoritmos de aprendizado competitivo, onde há uma competição entre os neurônios da camada de saída para se tornarem ativos, pois somente um neurônio pode estar ativo, em um determinado instante.

Uma regra de aprendizagem competitiva possui três elementos básicos, segundo Rumelhart e Zipser (1985), sendo eles:

1. Conjunto de neurônios iguais que respondem diferentemente a um conjunto de entrada, por conta de pesos sinápticos distribuídos aleatoriamente pela rede;
2. Limite sobre a “força” que cada neurônio possui;
3. Mecanismo onde o neurônio compete pelo direito de responder a um subconjunto da entrada de dados, de modo que em um dado instante, haja apenas um neurônio ativo por grupo de competição;

Os neurônios, com isso, ganham a capacidade de detectar características para diferentes classes de padrões de entrada, pois aprendem a se especializar em agrupamentos de padrões similares, conforme Haykin (1994).

A competição se dá da seguinte forma: o neurônio k será o vencedor se seu potencial de ativação v_k , para uma entrada \mathbf{x} , é o maior entre todos os neurônios daquela rede neural. Com isso, o sinal de saída é alterado com base na Equação:

$$y_k = \begin{cases} 1, & \text{se } v_k > v_j, \forall j \neq k \\ 0, & \text{caso contrário} \end{cases} \quad (2.58)$$

No início de um algoritmo de aprendizado competitivo, cada neurônio recebe uma quantidade fixa de peso sináptico, que é distribuída entre todos os seus nós de entrada. Dado este peso, o aprendizado ocorre quando o neurônio realoca o peso atribuído aos seus nós de entrada inativos para seus nós ativos. Se o neurônio vencer a competição, cada nó de entrada (ativo ou inativo) libera uma parte de seu peso sináptico, que será distribuída novamente entre todos os nós ativos deste neurônio.

Pela regra de aprendizagem competitiva, o vetor de peso sináptico \mathbf{w} do neurônio k é movido em direção à entrada \mathbf{x} , e o ajuste aplicado ao peso w_{kj} é dado por:

$$\Delta w_{kj} = \begin{cases} \eta(x_j - w_{kj}), & \text{se } k \text{ é o neurônio vencedor} \\ 0, & \text{se } k \text{ perdeu a competição} \end{cases} \quad (2.59)$$

Por fim, o último grupo é o dos algoritmos conhecidos como *máquinas de Boltzmann*, sendo algoritmos de aprendizado estocástico com base em princípios de mecânica estatística (ACKLEY; HINTON; SEJNOWSKI, 1985; HINTON; SEJNOWSKI, 1986).

Esta máquina se dá uma por uma função de energia E , onde a rede possui arquitetura recorrente e os neurônios possuem funcionamento binário, sendo representados por $+1$ ou -1 .

A função de energia é dada por:

$$E = -\frac{1}{2} \sum_j \sum_{\substack{k \\ j \neq k}} w_{kj} x_k x_j \quad (2.60)$$

sendo que:

- como $j \neq k$, nenhum neurônio possui autorrealimentação;
- x_j representa o estado do neurônio j ; e
- w_{kj} representa o peso da conexão entre os neurônios j e k .

A máquina de Boltzmann seleciona um neurônio ao acaso e altera seu estado de acordo com a probabilidade dada pela Equação (2.61), onde a constante aplicação desta regra leva a

máquina ao equilíbrio térmico:

$$P(x_k \rightarrow -x_k) = \frac{1}{1 + \exp(-\Delta E_k/T)} \quad (2.61)$$

sendo que:

- x_k é o estado atual do neurônio k ;
- $-x_k$ é o próximo estado;
- ΔE_k é a variação de energia resultante da troca de estado; e
- T é uma pseudotemperatura associada a essa probabilidade.

Os neurônios desta máquina são divididos em neurônios visíveis (interface entre a rede e o ambiente de operação da mesma) e ocultos (operação livre). Estas operações podem ter dois tipos de condição:

- **presa:** neurônios visíveis ficam presos a estados específicos que o ambiente determina; e
- **livre:** todos os neurônios operam livremente, visíveis ou ocultos.

Por fim, o ajuste aplicado a um peso sináptico w_{kj} , do neurônio j ao neurônio k , é dada pela Equação a seguir ([HINTON; SEJNOWSKI, 1986](#)):

$$\Delta w_{kj} = \eta(\rho_{kj}^+ - \rho_{kj}^-), j \neq k \quad (2.62)$$

sendo:

- ρ_{kj}^+ = correlação entre os neurônios j e k em uma rede com condição presa, dado por um valor entre -1 e $+1$;
- ρ_{kj}^- = correlação entre os neurônios j e k em uma rede com condição de operação livre, também dado por um valor entre -1 e $+1$.

Segundo [Haykin \(1994\)](#), as duas correlações vistas na equação anterior representam as médias sobre todos os estados possíveis da máquina de Boltzmann utilizada, durante seu equilíbrio térmico.

Os quatro grupos de algoritmos podem ser resumidos pela tabela a seguir:

Tabela 3: Grupos de algoritmos de treinamento de RNAs

Grupo	Funcionamento	Aprendizado
Correção de Erro	Pesos ajustados para reduzir erros	Supervisionado
Hebbiano	Regra de Hebb: se dois neurônios estão ativos simultaneamente, reforçar sua conexão	Não Supervisionado
Competitivo	Competição onde o neurônio vencedor tem seus pesos ajustados e responde mais fortemente à entrada da RNA	Não Supervisionado
Boltzmann	Algoritmos com base em princípios de mecânica estatística	Não Supervisionado

Fonte: a autora. Adaptado de [Faceli et al. \(2011\)](#).

Um dos algoritmos mais utilizados de redes neurais, e que será aplicado neste trabalho, é o *Multilayer Perceptron* (MLP), detalhado a seguir.

2.4.1 *Multilayer Perceptron (MLP)*

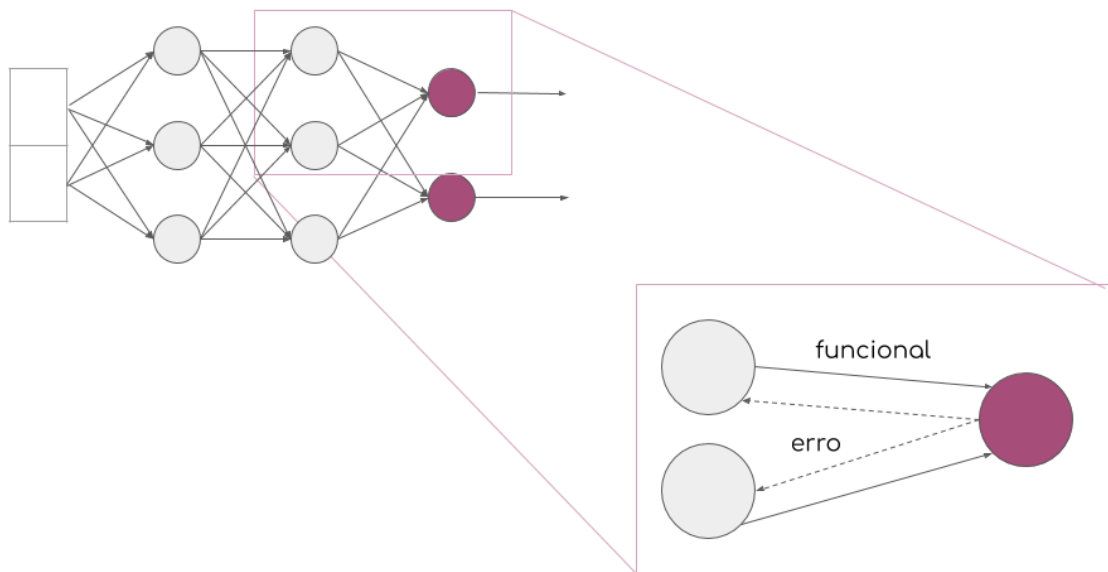
Multilayer Perceptron é um tipo de rede neural *feedforward* multicamadas, obtido da generalização de um *Perceptron* de camada única, possuindo uma camada de entrada, uma ou mais ocultas e uma de saída. Além disso, seus neurônios estão completamente conectados, e utilizam funções de ativação não-lineares, como a sigmoide ([FACELI et al., 2011](#); [HAYKIN, 1994](#)).

Seu aprendizado é supervisionado e ocorre por meio do algoritmo de *backpropagation*, ou retropropagação, sendo pertencente ao grupo de algoritmos de aprendizagem por correção de erro. Neste algoritmo, cada neurônio de uma MLP é responsável pelo cálculo e transmissão de dois tipos de sinal: funcional e de erro ([PARKER, 1987](#)).

Um sinal funcional, também chamado de *sinal de entrada*, é calculado como uma função das entradas de um neurônio, junto aos pesos associados a elas, e é propagado para a frente, de neurônio a neurônio, até a saída. Enquanto isso, um sinal de erro é originado em um neurônio de saída e se propaga para trás, sendo calculado por uma função dependente do erro cometido pela rede, durante o processo de aprendizado.

Todos os neurônios das camadas ocultas e de saída calculam o sinal funcional e uma estimativa do vetor gradiente, utilizado na retropropagação, contendo os gradientes da superfície de erro, em relação aos pesos associados às suas entradas.

Um exemplo de MLP, com seus sinais funcionais e de erro, pode ser visto na Figura 21.

Figura 21: *Multilayer Perceptron* com sinais envolvidos na retropropagação

Fonte: a autora. Adaptado de [Haykin \(2010\)](#).

Existem dois métodos de implementação do *backpropagation*, sendo eles o método sequencial e o método por lote. Segundo [Haykin \(1994\)](#), o sequencial possui diversas vantagens em relação ao modo por lote, dentre elas:

- as conexões sinápticas da rede exigem menos espaço de armazenamento;
- busca no espaço de exemplos possui natureza estocástica;
- o modo sequencial possui mais vantagens em conjuntos redundantes; e
- implementação simples que soluciona problemas grandes e difíceis com eficiência.

Por estas razões, neste trabalho, será apresentado somente o modo sequencial.

Para a inicialização do algoritmo, devem ser escolhidos bons valores iniciais para os pesos sinápticos e limiares, de modo a evitar saturação do neurônio (fenômeno que acarreta em diminuição da velocidade do aprendizado) e evitar que o processo de *backpropagation* ocorra em uma região muito plana em torno da origem da superfície de erro, o que também prejudica o aprendizado da rede neural.

Uma estratégia usada para esta escolha envolve selecionar os pesos sinápticos a partir de um conjunto de números com distribuição uniforme, tal que a média desta distribuição seja zero, e a variância seja definida de modo que o desvio padrão dos potenciais de ativação v (ver

Equação (2.52)) dos neurônios esteja entre a parte linear e a parte saturada da função de ativação escolhida.

Com os pesos inicializados, inicia-se o processo de aprendizagem da rede com a apresentação de uma época à ela, sendo que uma época é uma apresentação completa do conjunto de treino sendo processado (HAYKIN, 1994).

Cada exemplo nesta época é representado por um par $(\mathbf{x}(n), \mathbf{d}(n))$, onde $\mathbf{x}(n)$ representa o dado em si, com todos os seus atributos, na n -ésima iteração, e $\mathbf{d}(n)$ representa a saída esperada para o exemplo. No caso de problemas de classificação, a saída é dada por um vetor, onde a posição com valor 1 representa a classe esperada, enquanto as outras posições recebem valor 0.

No modo sequencial de *backpropagation*, dada uma época $(\mathbf{x}(1), \mathbf{d}(1)), \dots, (\mathbf{x}(N), \mathbf{d}(N))$, os N exemplos são apresentados aos neurônios da rede, um de cada vez, fazendo a computação desse dado e os ajustes dos pesos a cada recebimento de dado, até que todos os exemplos desta época tenham sido processados. Em cada uma das épocas, a ordem de apresentação dos exemplos deve ser aleatória.

A computação dos exemplos ocorre em duas etapas, sendo a primeira delas a computação para frente, também chamada de *propagação*.

Nesta etapa, considerando o exemplo $(\mathbf{x}(n), \mathbf{d}(n))$, tem-se que $\mathbf{x}(n)$ é apresentado à camada de entrada, e $\mathbf{d}(n)$, à camada de saída. Dadas estas informações, para cada neurônio, calcula-se o seu potencial de ativação que, no caso de redes multicamadas, como a MLP, é dada pela Equação:

$$v_j^{(l)}(n) = \sum_{i=0}^{m_0} w_{ji}^{(l)}(n) y_i^{(l-1)}(n) \quad (2.63)$$

onde:

- $w_{ji}^{(l)}$ = peso sináptico entre a saída do neurônio i (na camada $l - 1$) e a entrada do neurônio j (na camada l);
- $y_i^{(l-1)}(n)$ = sinal funcional de saída do neurônio i , presente na camada $l - 1$, durante a iteração n ; e
- m_0 representa o tamanho da camada de entrada.

O sinal de saída do neurônio, por sua vez, é dado por:

$$y_j^{(l)} = \varphi_j(v_j(n)) \quad (2.64)$$

sendo φ_j uma função de ativação sigmoide. Como visto na Seção 2.4, um exemplo de função sigmoide é a logística, que no caso da MLP, sua forma geral pode ser dada por:

$$\varphi_j(v_j(n)) = \frac{1}{1 + \exp(-av_j(n))}, a > 0, -\infty < v_j(n) < \infty \quad (2.65)$$

Também pode-se utilizar a função tangente hiperbólica, cuja forma geral é dada por:

$$\varphi_j(v_j(n)) = a \tanh(bv_j(n)), (a, b) > 0 \quad (2.66)$$

Existem alguns casos especiais para o cálculo do sinal de saída. Temos que se o neurônio j está na primeira camada intermediária, onde $l = 1$, então $y_j^{(0)}(n) = x_j(n)$, enquanto que se estiver na camada de saída, onde $l = L$, sendo L a profundidade da rede, será dada por:

$$y_j^{(L)}(n) = o_j(n) \quad (2.67)$$

Sendo que $o_j(n)$ representa o j -ésimo elemento do vetor de saída (HAYKIN, 1994).

Com as saídas calculadas, o último passo da etapa de propagação é dado pelo cálculo do sinal de erro, para cada um dos neurônios da camada de saída da rede neural. Este cálculo é dado por:

$$e_j(n) = d_j(n) - o_j(n) \quad (2.68)$$

Sendo $d_j(n)$ o j -ésimo elemento do vetor que contém a saída esperada, $\mathbf{d}(n)$.

Com o erro calculado, pode-se definir a *energia média do erro quadrado*, dada por:

$$\mathcal{E}_{med} = \frac{1}{N} \sum_{n=1}^N \mathcal{E}(n) \quad (2.69)$$

onde:

- N representa o número de exemplos no conjunto de treino; e
- $\mathcal{E}(n)$ representa o valor instantâneo da energia total do erro durante a iteração n .

Este valor instantâneo da energia total do erro, por sua vez, se dá por:

$$\mathcal{E}(n) = \frac{1}{2} \sum_{j \in C} e_j^2(n) \quad (2.70)$$

sendo que C é o conjunto dos neurônios da camada de saída, e $e_j(n)$ representa o sinal de erro emitido pelo neurônio j na saída da rede.

A energia média, para um dado conjunto de treino, é uma função de custo, representando uma medida do desempenho da aprendizagem da rede, sendo uma função de todos os parâmetros livres. Com isso, pode-se afirmar que o objetivo do algoritmo de *backpropagation* é ajustar os parâmetros da rede neural, de modo a minimizar \mathcal{E}_{med} , fornecendo uma aproximação para a trajetória a se seguir no espaço de pesos (HAYKIN, 1994; KARNIN, 1990; SILVA; ALMEIDA, 1990).

Enfim, após o cálculo do sinal de erro, definido na Equação (2.68), passa-se para a próxima etapa do algoritmo, de computação para trás (retropropagação), onde são feitos os ajustes dos pesos sinápticos de acordo com os sinais de erro obtidos na camada de saída.

O ajuste a um dado peso sináptico $w_{ji}(n)$ é proporcional ao gradiente local do neurônio j , ou seja, a um fator de sensibilidade que determina a direção de busca no espaço de pesos. O objetivo é sempre obter uma descida neste gradiente, buscando uma direção neste espaço que diminua a energia total de erro $\mathcal{E}(n)$.

O ajuste pode ser dado pela *regra delta*:

$$\Delta w_{ji}(n) = \eta \delta_j(n) y_i(n) \quad (2.71)$$

sendo que:

- $\Delta w_{ji}(n)$ = ajuste aplicado ao peso sináptico $w_{ji}(n)$;
- η = taxa de aprendizagem da rede neural;
- $\delta_j(n)$ = gradiente local (do neurônio j); e
- $y_i(n)$ = saída do neurônio i , recebida como entrada pelo neurônio j .

O cálculo do gradiente local depende de qual é a camada em que o neurônio avaliado está presente, conforme Haykin (1994). Temos os dois casos descritos na Equação a seguir:

$$\delta_j^{(l)}(n) = \begin{cases} e_j^{(L)}(n) \varphi'_j(v_j^{(L)}(n)), & \text{se neurônio } j \text{ está na camada de saída} \\ \varphi'_j(v_j^{(l)}(n))' \sum_k \delta_k^{(l+1)}(n) w_{kj}^{(l+1)}(n), & \text{se está na camada oculta } l \end{cases} \quad (2.72)$$

sendo que $\varphi'_j(\cdot)$ representa a derivada da função de ativação associada, em relação ao argumento desta.

A taxa de aprendizagem η tem influência direta sobre a velocidade de aprendizado e estabilidade da rede neural. Se seu valor for baixo, os pesos sinápticos sofrerão variações menores, mas o aprendizado será mais lento. Por outro lado, se for alto, o aprendizado será mais rápido, mas a rede se torna instável, oscilatória.

Para permitir o uso de uma taxa de aprendizagem mais alta com menor impacto à estabilidade do algoritmo, [Rumelhart, Hinton e Williams \(1986\)](#) propuseram uma modificação da regra delta, dada por:

$$\Delta w_{ji}(n) = \alpha w_{ji}(n-1) + \eta \delta_j(n) y_i(n) \quad (2.73)$$

sendo que α é uma constante de momento, responsável pelo controle da realimentação do ajuste $\Delta w_{ji}(n)$.

Com o ajuste do peso feito, termina-se a retropropagação, e por consequência, a computação feita para cada neurônio. Esta computação para frente e para trás é feita para todas as épocas apresentadas à MLP, até que se atinja o critério de parada definido para a rede, encerrando o aprendizado e a execução do algoritmo.

Para formulação de critérios de parada, considera-se um vetor de peso mínimo sobre a superfície de erro, dado por \mathbf{w}^* . Para que seja mínimo, o vetor gradiente $\mathbf{g}(\mathbf{w})$, dado pela derivada parcial de primeira ordem da superfície de erro, em relação ao vetor de peso \mathbf{w} , deve ser zero quando $\mathbf{w} = \mathbf{w}^*$. É importante destacar que $\mathbf{g}(\mathbf{w}^*) = 0$ é condição necessária, mas não suficiente.

Um critério de parada, proposto em [Kramer e Sangiovanni-Vincentelli \(1989\)](#), diz que o algoritmo se encerra quando a norma euclidiana de $\mathbf{g}(\mathbf{w})$ atinge um limiar suficientemente pequeno. Outro critério de parada, citado por [Haykin \(1994\)](#), enuncia que o algoritmo de *back-propagation* termina quando a taxa de variação do erro médio quadrado, por época, for suficientemente pequena. Como parâmetro, considera-se como suficientemente pequena uma taxa de variação entre 0,1% e 1% para cada época.

Cada critério citado possui suas desvantagens. O método proposto no trabalho de [Kramer e Sangiovanni-Vincentelli \(1989\)](#) requer o cálculo de todo o vetor gradiente $\mathbf{g}(\mathbf{w})$ e pode aumentar o tempo de aprendizagem, enquanto o segundo critério proposto pode encerrar o aprendizado prematuramente.

Por fim, dentre as vantagens do MLP, vistas em [Gardner e Dorling \(1998\)](#), há o fato de que este algoritmo não depende de nenhuma suposição sobre a distribuição dos dados no conjunto de treino. Além disso, as medidas de entrada mais discriminantes são selecionadas durante o treinamento da rede neural, com o ajuste dos pesos nas conexões entre os neurônios.

Entretanto, o aprendizado é mais difícil de ser visualizado, por existirem neurônios ocultos na rede. Também é um processo mais complexo, visto que é necessário que a rede neural decida quais características do conjunto de dados serão representadas por estes neurônios ocultos, tendo que levar em conta um espaço maior de possíveis funções ([HINTON, 1989](#)).

2.4.2 Vantagens e desvantagens das RNAs

Como já mencionado por [Braga, Ludermir e Carvalho \(2000\)](#) e [Haykin \(1994\)](#), as RNAs possuem grande capacidade de generalização e de tolerância a falhas e ruídos no conjunto de exemplos, além de uma taxa de erros baixa em várias aplicações, com destaque para tarefas de visão computacional e robótica ([FACELI et al., 2011](#)).

Outras vantagens incluem um treinamento estatístico menos formal em relação a outros algoritmos, habilidade de detectar relações não-lineares complexas entre variáveis dependentes e independentes, e também, de detectar todas as interações possíveis entre variáveis de entrada ([TU, 1996](#)).

Dentre as desvantagens, há a baixa interpretabilidade dos resultados gerados pelas RNAs ([BASHEER; HAJMEER, 2000](#)), sendo referenciadas como “caixas-pretas”, por conta da dificuldade de compreensão das decisões tomadas pela rede neural, ao longo do processo de aprendizado, em comparação com outros algoritmos de aprendizado de máquina, como as Árvores de Decisão. Por fim, também há a dificuldade na definição de parâmetros para a arquitetura da RNA, e o fato de que modelos de redes neurais artificiais são suscetíveis ao problema de *overfitting*.

2.5 Avaliação dos algoritmos de classificação

A análise do desempenho dos algoritmos citados anteriormente é essencial para sua otimização e na decisão de qual é o mais adequado para solucionar determinado problema, conforme [Forbes \(1995\)](#). A seguir, são detalhadas as métricas mais utilizadas na literatura atualmente, e que serão aplicadas neste trabalho.

2.5.1 *k-Fold cross validation*

O método *k-Fold* de validação cruzada é um dos mais utilizados para validar modelos de aprendizado de máquina, gerados por cada classificador.

Trata-se de um *estimador*, que de acordo com a área da inferência estatística, é uma função de uma certa amostra populacional, usada para estimar um parâmetro desta ([MORETTIN; BUSSAB, 2010](#)). Ou seja, o *k-Fold cross validation* utiliza partições dos conjuntos de exemplos para estimar uma certa métrica de desempenho dos classificadores avaliados - neste caso, a acurácia, que será detalhada posteriormente.

Neste método, o conjunto de exemplos é particionado em k subconjuntos diferentes, chamados de *folhas*, em que todas possuem aproximadamente o mesmo tamanho. Em seguida, são gerados k modelos diferentes, realizando k rodadas de geração, treino e teste dos modelos. Em todas as rodadas, uma destas folhas é selecionada para teste, enquanto todas as outras

ficam para treino do modelo, sendo uma folha de teste diferente a cada rodada (MONARD; BARANAUSKAS, 2003; HAN; KAMBER; PEI, 2012).

Com isso, o estimador para a acurácia do algoritmo de classificação avaliado se dá pela Função:

$$\text{acurácia estimada} = \frac{\text{total de acertos nas } k \text{ rodadas}}{\text{total de exemplos no conjunto original}} \quad (2.74)$$

Existem dois casos especiais de validação cruzada *k-Fold*, sendo:

- **Leave-one-out**: considerando um conjunto de n exemplos, tem-se n folhas com apenas um exemplo em cada, ficando somente um exemplo para teste dos modelos (WONG, 2015);
- **Estratificada**: cada folha segue aproximadamente a distribuição de classes observada no conjunto de exemplos original.

A versão mais recomendada para estimativa da acurácia é a estratificada com 10 folhas ($k = 10$), por conta de sua variância e viés baixos, segundo Han, Kamber e Pei (2012). Já a versão *Leave-one-out* é mais custosa, sendo indicada para uso com amostras pequenas.

2.5.2 Matriz de confusão

A matriz de confusão é uma métrica muito usada, que demonstra o número de classificações corretas em contraste com o de classificações preditas para cada classe. Pode ser representado por uma tabela em duas dimensões, com as classes verdadeiras e as preditas (MONARD; BARANAUSKAS, 2003).

Na diagonal principal desta matriz, localizam-se os acertos do algoritmo de classificação avaliado, enquanto as outras posições representam seus erros, o que pode ser melhor visualizado na Tabela 4, tomando como exemplo um problema envolvendo quatro classes possíveis para o conjunto de dados.

Tabela 4: Exemplo de matriz de confusão para um problema de 4 classes

Classe	Predita C1	Predita C2	Predita C3	Predita C4
Verdadeira C1	acertos em C1	erro de predição	erro de predição	erro de predição
Verdadeira C2	erro de predição	acertos em C2	erro de predição	erro de predição
Verdadeira C3	erro de predição	erro de predição	acertos em C3	erro de predição
Verdadeira C4	erro de predição	erro de predição	erro de predição	acertos em C4

Fonte: a autora. Adaptado de Monard e Baranauskas (2003).

A versão mais utilizada, porém, é a de classificação binária, que serve como base para o cálculo de diversas outras métricas de desempenho, como será visto nos próximos tópicos.

Tabela 5: Matriz de confusão para classificação binária

Classe	Predita P	Predita N
Verdadeira P	Verdadeiro Positivo	Falso Negativo
Verdadeira N	Falso Positivo	Verdadeiro Negativo

Fonte: a autora.

Nesta matriz de confusão, como visto na Tabela 5, existem quatro situações possíveis, descritas abaixo:

- Verdadeiro Positivo (ou *True Positive* (T_P)): algoritmo predisse corretamente a classe Positiva;
- Verdadeiro Negativo (ou *True Negative* (T_N)): algoritmo predisse corretamente a classe Negativa;
- Falso Positivo (ou *False Positive* (F_P)): algoritmo predisse como Positivo, sendo que a classe correta era a Negativa; e
- Falso Negativo (ou *False Negative* (F_N)): algoritmo predisse como Negativo, sendo que a classe correta era a Positiva.

2.5.3 Acurácia

A acurácia é uma medida que demonstra a porcentagem de classificações corretas feitas pelo algoritmo sobre a quantidade total de classificações (TAN et al., 2019), conforme a Equação (2.75), mostrada abaixo:

$$\text{acurácia} = \frac{T_P + T_N}{T_P + T_N + F_P + F_N} \quad (2.75)$$

Esta métrica também é estimada no método de validação cruzada *k-Fold*, conforme Seção 2.5.1.

Como visto em Han, Kamber e Pei (2012), não é recomendada para casos em que as classes estão desbalanceadas, isto é, quando a grande maioria dos exemplos pertencem à classe Negativa. Por conta da grande quantidade de exemplos negativos, o algoritmo pode estar apenas classificando estes corretamente, e gerando classificações incorretas para todos os exemplos da classe Positiva, o que torna a acurácia uma métrica não confiável, neste caso.

2.5.4 Erro

O erro (ou também, *taxa de erro*) é uma medida oposta à acurácia: demonstra a porcentagem de classificações erradas sobre o total de classificações, conforme a Equação a seguir:

$$\text{erro} = \frac{F_P + F_N}{T_P + T_N + F_P + F_N} = 1 - \text{acurácia} \quad (2.76)$$

2.5.5 Precision

A precisão, por sua vez, é uma medida de exatidão, dada pela porcentagem que expressa quantos exemplos classificados pelo algoritmo como Positivo pertencem de fato a esta classe (HAN; KAMBER; PEI, 2012). O cálculo é realizado com a Equação a seguir:

$$\text{precision} = \frac{T_P}{T_P + F_P} \quad (2.77)$$

2.5.6 Recall

Já o *recall*, medida também conhecida como *sensitividade*, expressa a porcentagem de exemplos positivos que foram corretamente classificados como tal. Esta métrica inclui também os Falsos Negativos (F_N), conforme a Equação:

$$\text{recall} = \frac{T_P}{T_P + F_N} \quad (2.78)$$

2.5.7 F-Score

As medidas de *precision* e *recall* possuem uma relação inversa entre si: quanto maior a precisão, menor o *recall*, e vice-versa.

Por conta dessa relação, elas frequentemente são utilizadas em conjunto, na medida *F-Score*, também conhecida por *F Measure* ou *F₁ Score*, que dá um peso igual aos valores de *precision* e *recall* (HAN; KAMBER; PEI, 2012). Esta medida é calculada pela Equação a seguir:

$$F = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \quad (2.79)$$

3 Revisão da literatura

Como base para os estudos desenvolvidos na Seção 2, foram reunidos 15 trabalhos onde a temática de análise de sentimentos é abordada, tanto com enfoque em críticas de filmes (mesmo tema deste trabalho), como em outros domínios. A seguir, são descritos detalhes de cada um dos artigos consultados.

3.1 Críticas de filmes

O trabalho de Koumpouri, Mporas e Megalooikonomou (2015) propõe quatro métodos diferentes para análise de opiniões de filmes, sendo: um modelo estatístico; um modelo de *Bag-of-Words*; um terceiro método baseado no conteúdo da opinião; e o último, baseado em léxico. Os autores utilizaram os algoritmos *k-Nearest Neighbours* (k-NN) e *Random Forest* como classificadores em todos os quatro modelos, que foram treinados e testados com críticas obtidas do site *Rotten Tomatoes*, apenas removendo a pontuação das frases no pré-processamento do texto.

Diversas outras pesquisas, por sua vez, foram desenvolvidas utilizando conjuntos de dados advindos da *Internet Movie Database* (IMDb) (BHOIR; KOLTE, 2015; PANG; LEE; VAITHYANATHAN, 2002; SAHU; AHUJA, 2016; SINGH et al., 2013; WHITE LAW; GARG; ARGAMON, 2005).

Sahu e Ahuja (2016) realizam mais etapas de pré-processamento, utilizando as técnicas de *stemming* com o algoritmo de Porter, remoção de *stop words* e *POS Tagging*. A análise é feita com uma abordagem léxica, fazendo uso do SentiWordNet (ESULI; SEBASTIANI, 2006), um recurso léxico amplamente empregado na literatura para mineração de opiniões. Os dados do IMDb são, então, classificados com *Random Forest*, *Bagging*, Árvore de Decisão, k-NN, *Naïve Bayes* e com classificação via regressão.

A pesquisa desenvolvida por Bhoir e Kolte (2015) é semelhante, também utilizando SentiWordNet, além de uma segunda abordagem com o algoritmo *Naïve Bayes*, para realizar a análise de sentimentos com base em aspectos variados do filme. O processamento de texto envolve *POS Tagging*, juntamente com uma conversão de gírias da língua inglesa para palavras apropriadas.

A abordagem usada por Singh et al. (2013) não envolve aprendizado de máquina, analisando os sentimentos com base apenas no SentiWordNet. Os algoritmos implementados pelos autores realizam a análise em nível de aspecto (buscando opiniões sobre os efeitos visuais, por exemplo, e analisando a polaridade destas) e em nível de documento (análise da polaridade da opinião como um todo). A etapa de pré-processamento do texto envolve somente a técnica de *POS Tagging*.

Pang, Lee e Vaithyanathan (2002) também realizam análise em nível de documento com *POS Tagging* no pré-processamento, além de utilizar *tags* para indicar sentimento negativo em expressões como “não muito bom”, por exemplo. Os classificadores utilizados foram *Naïve Bayes*, algoritmo de entropia máxima (MaxEnt) e SVM.

Dentre todas as pesquisas vistas nesta revisão da literatura, relacionadas a críticas de filmes, o trabalho de Whitelaw, Garg e Argamon (2005) traz a abordagem mais distinta, com base na Teoria da Avaliatividade (também conhecida como *Appraisal Theory*), que envolve a extração e análise de grupos de avaliação, sendo estes conjuntos de palavras que, juntas, expressam uma atitude particular, como “não muito bom”, por exemplo. Cada crítica é pré-processada em frases individuais, que são convertidas para caixa baixa e passam pelo processo de *POS Tagging*. A classificação, por fim, é feita com o algoritmo de *Sequential Minimal Optimisation* (SMO), utilizando a implementação disponível no software WEKA.

Hodeghatta (2013), por sua vez, realizou a coleta de cerca de um milhão de mensagens postadas no Twitter para análise, sobre 6 filmes diferentes, e utilizou o texto original de cada postagem, sem nenhum pré-processamento. Posteriormente, a classificação destas críticas em três classes diferentes - positivo, negativo ou cognitivo - é feita com os algoritmos *Naïve Bayes* e de entropia máxima (MaxEnt), similar ao trabalho de Pang, Lee e Vaithyanathan (2002).

Outro trabalho com foco em análise de sentimentos de *posts* do Twitter é o de Amolik et al. (2016). O texto é processado com remoção de *stop words*, espaços em branco, palavras repetidas, *emoticons* e *hashtags*, antes da geração do vetor de atributos para cada opinião. A classificação do sentimento dos *tweets* é feita com *Naïve Bayes* e SVM.

Nanda, Dua e Nanda (2018) analisaram sentimentos em opiniões retiradas de diversas redes sociais, como Facebook e Twitter, além de utilizar dados do IMDb, conforme feito na maioria dos trabalhos apresentados neste capítulo. Os textos foram submetidos à remoção de *stop words* e, posteriormente, classificados com os algoritmos *Random Forest* e SVM.

Zhang et al. (2018), por fim, propõem um modelo baseado na rede neural do tipo *Long Short-Term Memory* (LSTM), em conjunto com um mecanismo de atenção, para fazer a análise de críticas em língua chinesa, retiradas do *site* Douban. Nenhum tipo de pré-processamento foi realizado no texto das opiniões.

3.2 Outros domínios

Na Seção 2.1.3, foram abordadas algumas das áreas de aplicação de análise de sentimentos, como hotelaria e alimentação, destacadas por alguns trabalhos já citados.

No trabalho de Guzman e Maalej (2014), faz-se uma análise de opiniões sobre aplicativos de celular, juntamente com extração de atributos e modelagem de tópicos, de modo a gerar um relatório completo sobre o desempenho de determinado aplicativo, com base em cada

uma das características mais mencionadas pelos usuários. Os dados utilizados foram extraídos das lojas *Google Play Store*, de aplicativos Android, e *App Store*, para iOS. Antes da etapa de extração de atributos, as técnicas de remoção de *stop words* e lematização foram aplicadas aos exemplos. Para a análise de sentimentos em si, foi utilizada uma ferramenta lexical para extração de sentimentos, conhecida como SentiStrength (THELWALL et al., 2010).

Kasper e Vela (2011) apresentam um sistema que integra o projeto BESAHOT, uma aplicação Web onde se pode analisar os *feedbacks* dados por clientes de hotéis da Alemanha. Neste sistema, ocorre uma extração constante de dados de diversos *websites* do país, destinados a avaliações de hotéis pelos seus hóspedes, e estes textos são submetidos a pré-processamento e classificação de sentimentos, onde são combinados com seus resultados e armazenados em bancos de dados, para serem apresentados ao usuário do sistema, posteriormente. O pré-processamento envolve aplicação de tokenização, *POS Tagging*, verificação ortográfica e, por fim, *stemming*. Então, a classificação da polaridade é feita com base em modelagem de linguagem N-grama (onde o N-grama constitui uma sequência contígua de n itens de um texto) e técnica de *smoothing*, vistas nos trabalhos de Chen e Goodman (1999), Steffen (2004).

Outra aplicação vista em Liau e Tan (2014) mostra o uso de análise de sentimentos em opiniões sobre serviços de companhias aéreas, obtidas também do Twitter, em relação a duas companhias da Malásia. Neste caso, o pré-processamento de texto se dá pela remoção de pontuação, *emojis*, *links*, dígitos e caracteres especiais, conversão para caixa baixa, além de tokenização, *stemming* e remoção de *stop words*. A análise é feita também com base em léxico, calculando-se o *score* de sentimentos para cada crítica e utilizando SentiStrength para avaliar suas polaridades. Posteriormente, os autores realizam uma tarefa de *clustering*, apresentando a polaridade das opiniões de acordo com cada *cluster*, ou seja, cada tópico abordado, como compra de passagens ou cancelamento de voos.

Os últimos dois trabalhos vistos na literatura, por sua vez, realizam análise de opiniões de clientes de restaurantes, em *sites* especializados. Kang, Yoo e Han (2012) realizou o pré-processamento com a remoção de caracteres especiais e, em seguida, *POS Tagging*. Para a classificação, os autores utilizaram uma versão modificada do algoritmo *Naïve Bayes*, além do SVM. Por fim, Zhang et al. (2011) utiliza dados obtidos do *site* OpenRice, de Hong Kong, coletando críticas em cantonês (dialeto da língua chinesa), não realizando pré-processamento do texto, apenas utilizando modelagem de linguagem N-grama. Assim como Kang, Yoo e Han (2012), os algoritmos utilizados para a análise de sentimentos foram *Naïve Bayes* e *Support Vector Machine*.

3.3 Síntese da revisão

A seguir, na Tabela 6, pode-se ver uma relação de todos os artigos apresentados nesta revisão da literatura, por ano de publicação.

Tabela 6: Informações sobre as publicações

ID	Autores	Ano de Publicação
A01	Pang, Lee e Vaithyanathan (2002)	2002
A02	Whitelaw, Garg e Argamon (2005)	2005
A03	Kasper e Vela (2011)	2011
A04	Zhang et al. (2011)	2011
A05	Kang, Yoo e Han (2012)	2012
A06	Hodeghatta (2013)	2013
A07	Singh et al. (2013)	2013
A08	Guzman e Maalej (2014)	2014
A09	Liau e Tan (2014)	2014
A10	Bhoir e Kolte (2015)	2015
A11	Koumpouri, Mporas e Megalooikonomou (2015)	2015
A12	Amolik et al. (2016)	2016
A13	Sahu e Ahuja (2016)	2016
A14	Nanda, Dua e Nanda (2018)	2018
A15	Zhang et al. (2018)	2018

Fonte: a autora.

Por fim, na Tabela 7, apresenta-se uma síntese de todas as características dos trabalhos citados, incluindo abordagem e técnicas de processamento de texto utilizadas.

Tabela 7: Síntese das abordagens utilizadas em cada publicação

ID	Abordagem	Dataset/Aplicação	Pré-processamento
A01	MaxEnt Naïve Bayes SVM	IMDb	POS Tagging Tags em palavras negativas
A02	SMO	IMDb	Frases individuais Caixa baixa POS Tagging
A03	N-grama Smoothing	sites especializados	Tokenização POS Tagging Verificação ortográfica Stemming
A04	N-grama Naïve Bayes SVM	OpenRice	N/A
A05	Naïve Bayes SVM	sites especializados	Caracteres especiais POS Tagging
A06	MaxEnt Naïve Bayes	Twitter	N/A
A07	biblioteca SentiWordNet	IMDb	POS Tagging
A08	SentiStrength	Google Play Store App Store	Stop words Lematização
A09	SentiStrength Clustering	Twitter	Tokenização Stop words Caixa baixa Stemming Pontuação, Emojis e especiais
A10	Naïve Bayes	IMDb	POS Tagging Conversão de gírias
A11	k-NN Random Forest	Rotten Tomatoes	Remoção de pontuação
A12	Naïve Bayes SVM	Twitter	Stop words Espaços em branco Palavras repetidas Emoticons e hashtags
A13	Árvore de Decisão Bagging Classificação via Regressão k-NN Naïve Bayes Random Forest	IMDb	Stemming Stop words POS Tagging
A14	Random Forest SVM	Facebook IMDb Twitter	Stop words
A15	rede neural LSTM	Douban	N/A

Fonte: a autora.

3.4 Considerações

Nota-se que além das redes sociais mais conhecidas atualmente (como Facebook e Twitter), o *site* IMDb compõe as bases de dados mais utilizadas pela maioria dos trabalhos apresentados anteriormente.

Outro ponto a se destacar é o grande emprego do algoritmo *Naïve Bayes* na etapa de classificação dos sentimentos, seguido de *Random Forest* e SVM. Além disso, também pode-se ressaltar o grande uso das técnicas para pré-processamento estudadas no Capítulo 2.2.

Por fim, conforme destacado na Seção 2.1.3, a Tabela 6 reforça o fato de que a análise de sentimentos começou a ser mais explorada durante os anos 2000, demonstrando a crescente importância de tal área atualmente.

4 Proposta do trabalho

Nesta seção, são apresentados em mais detalhes os objetivos do trabalho e a metodologia empregada.

4.1 Objetivos

O principal objetivo do trabalho proposto é realizar um estudo de diferentes algoritmos de processamento de textos e aprendizado de máquina aplicados à análise de sentimentos em documentos de texto, com enfoque em críticas de filmes, advindas de *sites* especializados neste segmento.

A seguir, são detalhados os objetivos específicos do presente trabalho:

- Estudar as principais técnicas para processamento de texto, como lematização, *stemming* e geração de vetores de atributos.
- Pesquisar os principais algoritmos de aprendizado de máquina utilizados em tarefas de classificação.
- Aplicar técnicas de processamento de texto estudadas aos conjuntos de dados usando Python.
- Implementar algoritmos de aprendizado de máquina e de redes neurais artificiais, em linguagem Python.
- Realizar experimentos utilizando os conjuntos de dados processados.
- Avaliar os resultados obtidos em cada algoritmo implementado.

4.2 Metodologia

A seguir são descritos os métodos, técnicas e ferramentas computacionais que serão utilizadas no trabalho.

4.2.1 Bibliotecas utilizadas

A implementação dos algoritmos a serem estudados neste trabalho será feita em linguagem Python, na versão 3.7, com a utilização das bibliotecas listadas a seguir.

- **Matplotlib**: biblioteca com recursos específicos para plotagem de gráficos em duas dimensões, usada para melhor visualização de informações durante a análise exploratória de *datasets*, por exemplo.
- **Natural Language Toolkit (NLTK)**: plataforma com recursos para processamento de texto em linguagem humana, como interface para o modelo Word2vec, *stemming* e *POS Tagging*.
- **NumPy**: biblioteca que contém ferramentas sofisticadas para computação científica, permitindo trabalhar com arranjos, vetores, matrizes de N dimensões, além de também realizar operações de álgebra linear, por exemplo.
- **Pandas**: biblioteca largamente utilizada na área de análise de dados, com estruturas de dados específicas (como o *DataFrame*, também utilizado na linguagem R) e operações para manipulação delas.
- **Scikit-learn**: biblioteca que fornece diversas ferramentas para se trabalhar com aprendizado de máquina, mineração e análise de dados em Python, contendo métodos para implementação dos algoritmos abordados neste trabalho.

4.2.2 Datasets

Para o estudo dos algoritmos de aprendizado de máquina selecionados, neste trabalho, foram utilizados dois conjuntos de dados, contendo críticas a diversos filmes.

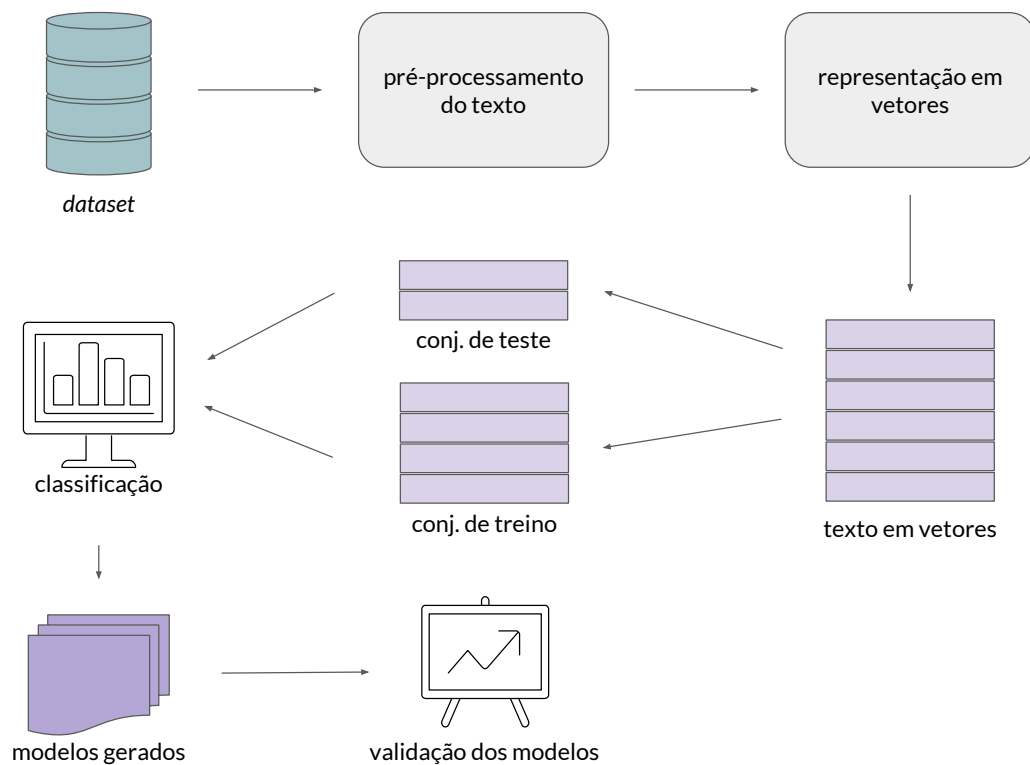
O primeiro *dataset* é o *Large Movie Review Dataset*, contendo 50.000 críticas, hospedadas no *Internet Movie Database* (IMDb) e coletadas por [Maas et al. \(2011\)](#). O *dataset* é originalmente dividido em dois conjuntos, destinados a treino e teste dos classificadores, em que cada um contém 25.000 opiniões, sendo 12.500 classificadas como positivas e outras 12.500 pertencentes à classe negativa.

O segundo conjunto de dados, por sua vez, é composto de cerca de 55.000 *reviews* retiradas do site *Rotten Tomatoes*, divididas também entre opiniões positivas e negativas. Neste caso, originalmente, o *dataset* não possuía uma divisão em conjuntos de treino e teste. Esta divisão foi realizada posteriormente, durante a fase de experimentos deste trabalho.

4.2.3 Fluxo de trabalho

O fluxo de trabalho a ser desenvolvido se divide em três etapas, ilustradas na Figura 22 e detalhados nas próximas seções.

Figura 22: Fluxo de trabalho proposto



Fonte: a autora.

4.2.3.1 Processamento de texto

A primeira etapa envolve o processamento do texto contido em todas as opiniões de filmes, englobando o pré-processamento e, posteriormente, a representação do texto como vetores de atributos, interpretáveis pelos classificadores a serem utilizados nas próximas fases.

Todas as críticas contidas em ambos os *datasets* serão pré-processadas com o uso das seguintes técnicas:

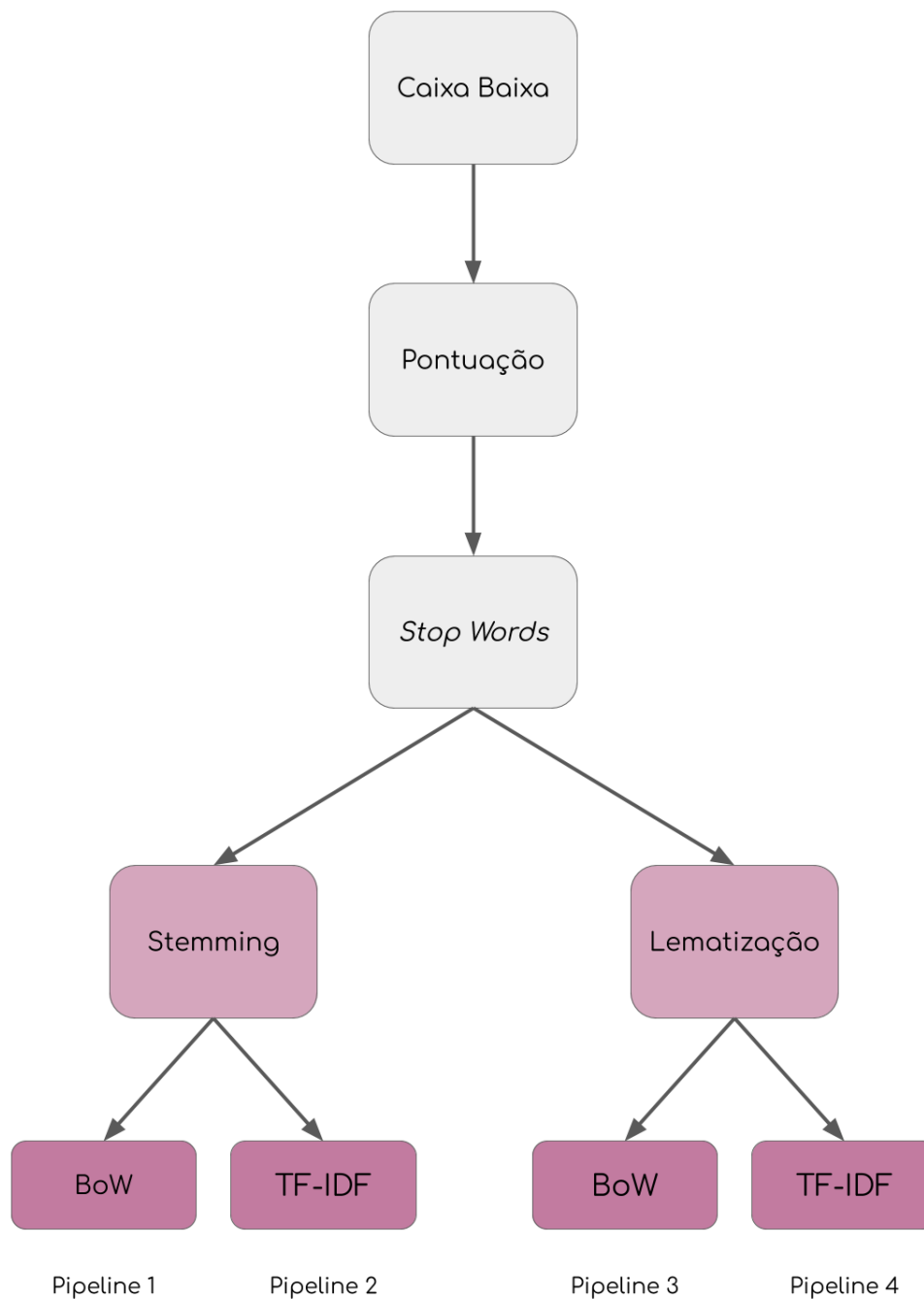
- Conversão para caixa baixa;
- Remoção de pontuações (incluindo *tags* HTML presentes no *dataset* do IMDb); e
- Remoção de *stop words*.

Conforme mencionado na Seção 2.2, as técnicas de *stemming* e de lematização não são utilizadas juntas. Além disso, existem dois métodos possíveis para representação dos textos como vetores de atributos, sendo o *Bag-of-Words* - gerado com a medida de *Term Frequency* (TF) - e o TF-IDF. Portanto, a partir deste ponto, o processamento de texto realizado neste trabalho pode ser dividido em quatro *pipelines* distintos, sendo eles:

1. Aplicação de *stemming*, representando textos por *Bag-of-Words*;
2. Aplicação de *stemming*, representando textos por TF-IDF;
3. Aplicação de lematização, representando textos por *Bag-of-Words*; e
4. Aplicação de lematização, representando textos por TF-IDF.

Estes quatro *pipelines* podem ser vistos na Figura 23.

Figura 23: *Pipelines* de processamento de textos



Fonte: a autora.

Após o processamento das opiniões por um dos *pipelines* descritos acima, obtêm-se os vetores de atributos que serão, finalmente, usados na etapa de classificação.

4.2.3.2 Classificação

Utilizando os vetores de atributos representando cada crítica, a próxima etapa do fluxo de trabalho envolve classificar o sentimento de cada uma das opiniões obtidas dos *datasets*, aplicando técnicas de aprendizado de máquina.

Para esta tarefa, serão utilizados seis algoritmos de classificação diferentes, sendo eles:

- *Naïve Bayes*;
- *Support Vector Machine* (SVM);
- Árvore de Decisão;
- *Random Forest*;
- *k-Nearest Neighbours* (k-NN); e
- *Multilayer Perceptron* (MLP).

4.2.3.3 Validação

Com os modelos de classificação gerados, será feita uma validação de todos eles, com o objetivo de avaliar o desempenho de cada algoritmo na etapa anterior e realizar comparações entre eles.

Para isto, serão usadas as seguintes métricas, listadas a seguir:

- *k-Fold cross validation*;
- Matriz de confusão;
- Acurácia;
- *Precision*;
- *Recall*; e
- F_1 Score.

É importante ressaltar que apesar de o *k-Fold cross validation* ser considerado uma métrica de desempenho, ele será aplicado durante o treino e teste dos classificadores, feitos na etapa anterior.

4.2.4 Tipos de modelos para classificação

Considerando os quatro *pipelines* de processamento de texto descritos anteriormente, e os seis classificadores selecionados, existem, no total, 24 modelos a serem implementados para a tarefa de análise de sentimentos, proposta neste trabalho.

Uma ilustração das combinações de técnicas e algoritmos possíveis pode ser vista na Figura 24.

Figura 24: Modelos propostos para análise de sentimentos



Fonte: a autora.

Na próxima seção, serão descritos os resultados da implementação de todos os 24 modelos propostos acima.

5 Resultados

Neste capítulo, apresentam-se os resultados da análise de sentimentos realizada sobre os dois conjuntos de dados selecionados, contendo *reviews* de filmes. Inicialmente, é realizada uma exploração visual e estatística destes dados. Em seguida, apresenta-se a configuração e resultados dos experimentos realizados com o uso de aprendizado de máquina. Por fim, é feita uma discussão de todos os resultados obtidos.

5.1 Análise exploratória dos conjuntos de dados

As primeiras análises feitas durante a fase experimental deste trabalho envolvem uma visualização dos conjuntos de dados selecionados para o desenvolvimento dos modelos, em relação à distribuição das classes das opiniões e às palavras mais frequentemente encontradas em cada *dataset*.

O primeiro *dataset* analisado é o do site *Rotten Tomatoes*. Este conjunto de dados, disponível no *Kaggle*¹, é composto de dois arquivos do tipo TSV. Um deles contém informações variadas a respeito do filme, incluindo sinopse, gênero, nome do diretor, tempo de duração, bilheteria, entre outras. O outro, por sua vez, além das opiniões dadas sobre os filmes, possui mais dados relacionados, como o nome do crítico e a nota dada por ele ao filme. Para o treino dos modelos desenvolvidos neste trabalho, apenas o arquivo com as *reviews* foi utilizado, considerando-se somente a crítica e sua classe.

Originalmente, o *dataset Rotten Tomatoes* possuía aproximadamente 54.500 exemplos, sendo que cerca de 5.500 deles continham valores faltantes. Dado o fato de que estes valores faltantes seriam do tipo texto, estes exemplos foram descartados. Desta forma, após a limpeza, o *dataset* passou a ser composto de 49.000 exemplos.

As opiniões contidas neste conjunto de dados receberam classificação binária, ou seja, apenas como “positivo” (1) ou “negativo” (0). As duas classes apresentam desbalanceamento entre si, sendo que cerca de 60% dos exemplos são classificados como positivos, enquanto os 40% restantes são negativos. A distribuição dos exemplos pode ser observada na Figura 25.

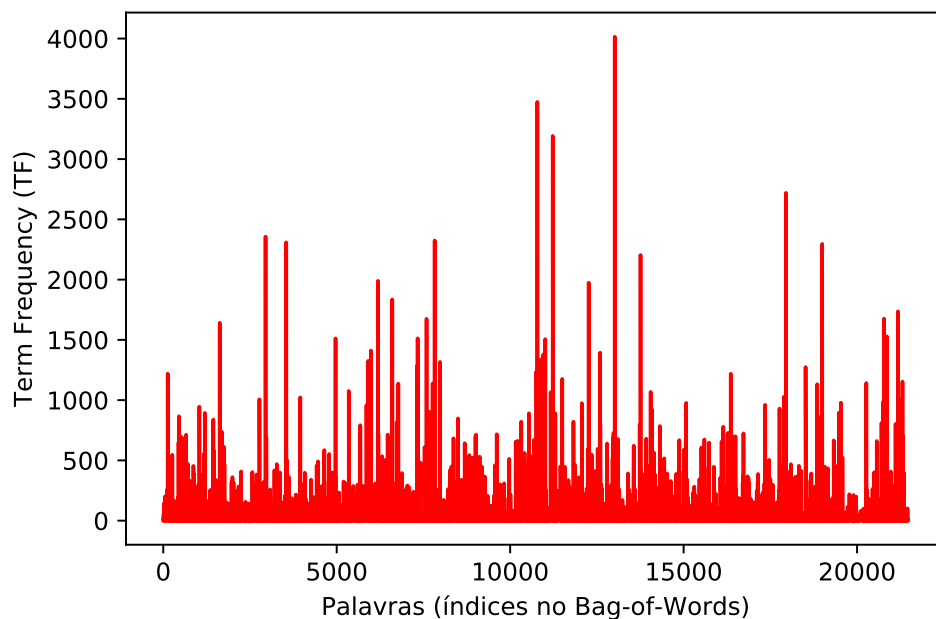
¹ <https://www.kaggle.com/rpnuser8182/rotten-tomatoes>

Por esta nuvem de palavras, pode-se notar alguns dos aspectos mais importantes em um filme e destacados pelos consumidores nas críticas recebidas pelo *Rotten Tomatoes*, como:

- as performances dos atores participantes (“*character*”, “*performances*”);
- a qualidade do enredo (“*moments*”, “*script*”);
- o gênero (“*comedy*”, “*drama*”, “*thriller*”); e
- as emoções dos espectadores durante a experiência do filme analisado (“*humor*”, “*funny*”).

Ao representar este conjunto de dados como um *Bag-of-Words*, são gerados 21.474 atributos, e a distribuição de frequências destes pode ser visualizada na Figura 27.

Figura 27: Frequência dos termos contidos no *dataset* do site *Rotten Tomatoes*



Fonte: a autora.

Nota-se que poucas palavras possuem frequência mais alta, enquanto que grande parte delas possuem frequência baixa, sendo estas as mais importantes para a classificação dos sentimentos em um dado texto (RUNESON; ALEXANDERSSON; NYHOLM, 2007).

O *dataset* do IMDb, por sua vez, foi submetido a uma conversão para dois arquivos do tipo CSV, sendo que o conjunto de treino possui 25.000 exemplos, também classificados binariamente. Este *dataset* foi disponibilizado por Maas et al. (2011)².

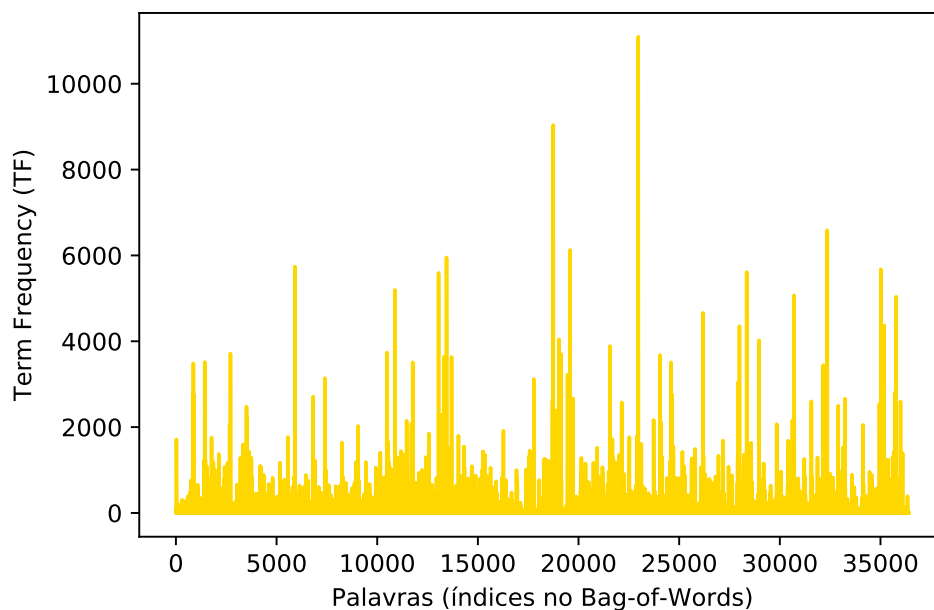
² <https://ai.stanford.edu/amaas/data/sentiment/>

Para este caso, nota-se que os tópicos principais são um pouco distintos dos abordados nas críticas do *Rotten Tomatoes*. Neste *dataset*, as opiniões possuem enfoque maior em:

- enredo do filme (“*better*”, “*great*”, “*plot*”, “*scenes*”); e
- atuações (“*director*”, “*acting*”, “*characters*”).

Por fim, após a geração do *Bag-of-Words* com o *dataset* completo, constatou-se que este conjunto de dados gera 72.537 atributos, no total. Ao realizar a extração de um subconjunto contendo 10.000 exemplos selecionados de forma aleatória, foram gerados 36.420 atributos. Analisando-se a frequência dos termos contidos neste subconjunto, obtêm-se os resultados visualizados na Figura 30.

Figura 30: Frequência dos termos contidos em um subconjunto do *dataset* do site IMDb



Fonte: a autora.

Assim como para o *Rotten Tomatoes*, existem poucas palavras no *dataset* obtido do IMDb com frequência mais alta, sendo a maior parte do conjunto de dados composta de palavras com frequências menores.

5.2 Configuração dos experimentos realizados

Na Seção 4.2.4, foram descritos os quatro *pipelines* de processamento de texto a serem utilizados com cada um dos algoritmos selecionados, totalizando 24 tipos diferentes de modelos a serem desenvolvidos e treinados sobre cada um dos *datasets*.

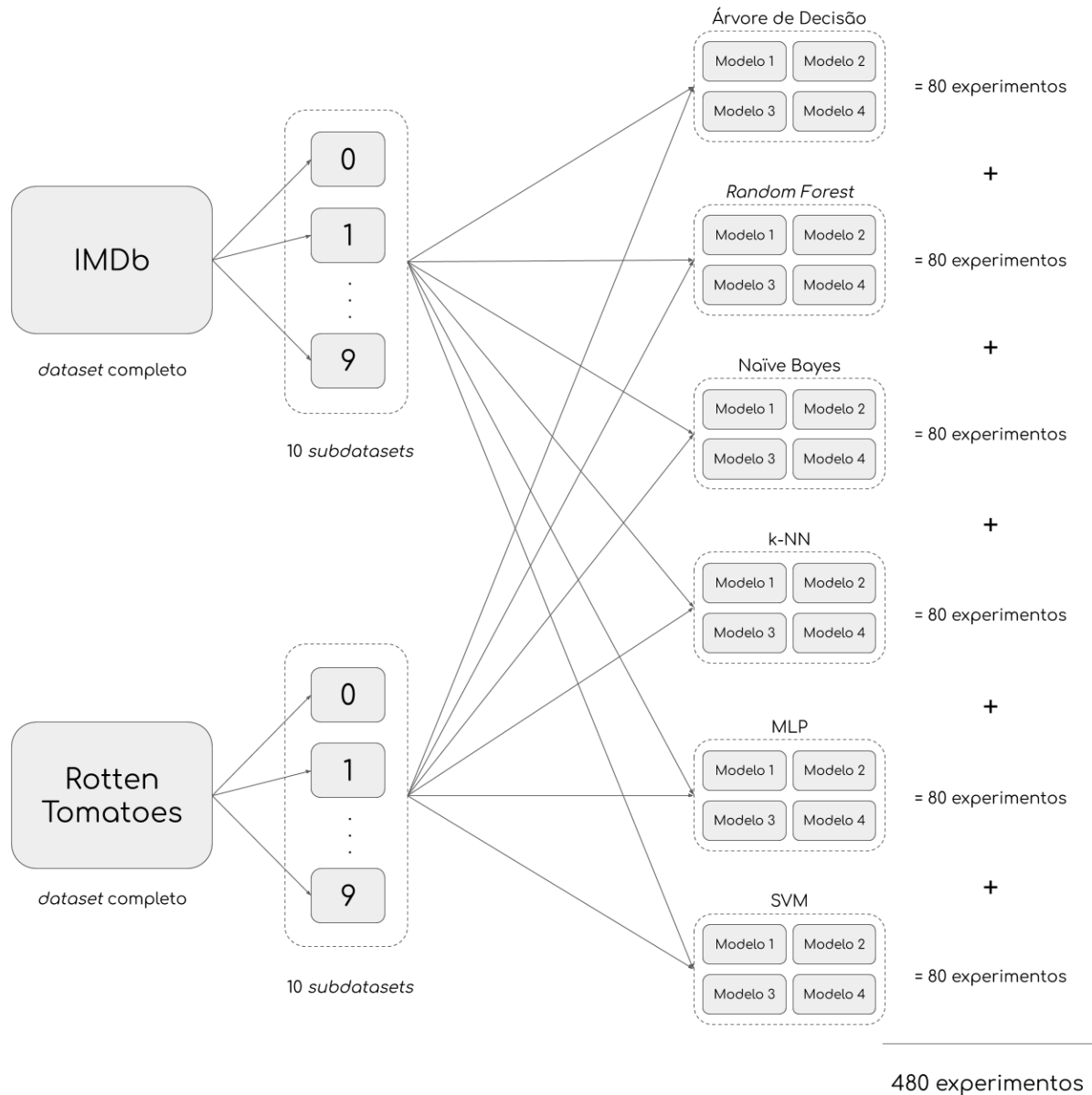
Entretanto, como dito anteriormente, durante a análise exploratória, foi constatado que o *dataset* do *site* IMDb possui 72.537 atributos, enquanto que o do *site* *Rotten Tomatoes* possui 21.474 atributos. Com a grande quantidade de *features* presentes, a representação do conjunto de dados do IMDb em *Bag-of-Words* ou TF-IDF implicaria em um grande custo computacional para sua geração e armazenamento.

Por conta de indisponibilidade de *hardware* com capacidade para tal tarefa, cada um dos 24 modelos implementados foi submetido a 10 experimentos, onde cada um era feito com um subconjunto distinto (não necessariamente disjunto) de 10.000 exemplos do *dataset* utilizado, sendo estes exemplos selecionados aleatoriamente, com variação da semente aleatória utilizada na geração de cada *subdataset*. Todos os experimentos deste trabalho foram executados com um processador Cloud TPU v2, projetado pelo Google³, utilizando memória RAM de 12,72 GB.

Considerando os dois *datasets* selecionados para este trabalho, no total, foram realizados 480 experimentos, além de 20 experimentos adicionais, feitos com um dos modelos implementados com o algoritmo *Support Vector Machine* (SVM), conforme será visto na Seção 5.3.2. Esta estratégia de divisão em 10 *subdatasets* foi utilizada de modo a garantir a representatividade dos dados, evitando viés no resultado final.

Uma ilustração da estratégia utilizada pode ser vista na Figura 31, para melhor entendimento.

³ <https://cloud.google.com/tpu/>

Figura 31: Estratégia utilizada para divisão dos *datasets* originais

Fonte: a autora.

5.3 Resultados dos experimentos

Após a realização dos 10 experimentos sobre um determinado modelo, todos os resultados obtidos com as métricas de desempenho e matrizes de confusão (ver Seção 2.5) foram resumidas em **médias aritméticas**, fornecendo uma visão geral de seu desempenho e qualidade. Dentre os quatro modelos implementados para cada algoritmo, o critério utilizado para escolher o melhor entre eles foi o maior valor de F_1 Score.

Nas próximas seções, para cada algoritmo, é demonstrado o resultado médio obtido por cada um dos quatro modelos implementados, juntamente com a média de atributos presentes

nos 10 *subdatasets* utilizados.

É importante lembrar que todos os modelos foram treinados e validados com o uso de *k-Fold cross validation*, para $k = 10$. Nesta abordagem, a divisão entre treino e validação destes 10 *Folds* é dada pela razão entre a quantidade de exemplos presentes no conjunto de dados e o número de *Folds*. Desta forma, os 10000 exemplos contidos no *dataset* foram divididos em 10 *Folds* de 1000 exemplos, onde 9 deles foram utilizados para o treino do modelo, e um deles para a validação.

Os resultados detalhados, com todos os 500 experimentos realizados neste presente trabalho, podem ser conferidos nos Apêndices A e B.

5.3.1 Naïve Bayes

Tanto para o *dataset* do IMDb quanto do *Rotten Tomatoes*, os quatro modelos que utilizam o algoritmo de *Naïve Bayes* foram implementados com os parâmetros padrões, definidos pela biblioteca Scikit-learn, e fazendo uso do modelo Multinomial.

Nesta biblioteca, existem dois tipos de modelos deste algoritmo que são utilizados em tarefas de classificação textual, sendo o Multinomial e o de Bernoulli. O modelo de Bernoulli considera textos representados como vetores binários, onde o valor de uma posição indica se uma determinada palavra está, ou não, presente no texto classificado. Já o modelo Multinomial, por sua vez, utiliza a frequência das palavras em um dado texto para suas análises (MCCALLUM; NIGAM, 1998), sendo o mais adequado para nossos experimentos.

Como pode ser visto na Tabela 8, referente ao *dataset* do IMDb, todos os quatro *pipelines* de processamento de texto levaram a um resultado semelhante, com F_1 Score de aproximadamente 0,84. Nota-se também que a representação de texto por TF-IDF levou a uma pequena melhoria, em comparação com os modelos que utilizaram *Bag-of-Words*.

Tabela 8: Experimentos realizados com *Naïve Bayes*, utilizando o *dataset* do site IMDb

Modelo	Pipeline	Acurácia	F_1 Score	Precision	Recall	Atributos
1	Stemming + BoW	0,8402	0,8355	0,8581	0,8144	36218,8
2	Stemming + TF-IDF	0,8463	0,8406	0,8703	0,8137	36218,8
3	Lematização + BoW	0,8403	0,8353	0,8600	0,8123	45340
4	Lematização + TF-IDF	0,8471	0,8405	0,8763	0,8083	45340

Fonte: a autora.

As matrizes de confusão médias também apresentaram semelhança, com o Modelo 4 apresentando o menor número de falsos positivos, e o Modelo 1, o menor número de falsos negativos.

Tabela 9: Matrizes de confusão dos experimentos com *Naïve Bayes*, sobre o *dataset* do site IMDb

Modelo 1		Modelo 2		Modelo 3		Modelo 4	
4061,7	926,8	4057,8	930,7	4051,1	937,4	4030,8	957,7
671,3	4340,2	606,1	4405,4	659,3	4352,2	570,9	4440,6

Fonte: a autora.

Já para o *dataset* do *Rotten Tomatoes*, cujos resultados estão presentes na Tabela 10, as acurácias obtidas foram menores, com os melhores resultados próximos a 0,80 de F_1 Score. Para este conjunto de dados, a representação por *Bag-of-Words* levou a uma pequena melhoria, em relação ao TF-IDF.

Tabela 10: Experimentos realizados com *Naïve Bayes*, utilizando o *dataset* do site *Rotten Tomatoes*

Modelo	Pipeline	Acurácia	F_1 Score	Precision	Recall	Atributos
1	Stemming + BoW	0,7489	0,8022	0,7712	0,8359	12060,2
2	Stemming + TF-IDF	0,7064	0,7993	0,6850	0,9598	12060,2
3	Lematização + BoW	0,7478	0,8013	0,7704	0,8350	14665,1
4	Lematização + TF-IDF	0,7012	0,7971	0,6799	0,9636	14665,1

Fonte: a autora.

As matrizes de confusão estão bastante distintas entre si, com os Modelos 2 e 4 apresentando valores bem menores de falsos negativos, em comparação com os outros dois modelos. Já os Modelos 1 e 3, por sua vez, obtiveram os menores números de falsos positivos, conforme Tabela 11.

Tabela 11: Matrizes de confusão dos experimentos do algoritmo *Naïve Bayes*, sobre o *dataset* do site *Rotten Tomatoes*

Modelo 1		Modelo 2		Modelo 3		Modelo 4	
5095,2	999,7	5849,9	245	5089,5	1005,4	5873	221,9
1511,2	2393,9	2690,7	1214,4	1516,6	2388,5	2766	1139,1

Fonte: a autora.

5.3.2 Support Vector Machine

Conforme visto na Seção 2.3.3, para tarefas de classificação, existem as SVMs lineares e não lineares. Neste trabalho, foram feitos experimentos com ambos os tipos de classificadores.

Considerando o fato de que as palavras com frequência mais alta não possuem uma importância tão grande para a classificação de um texto, e para treino mais rápido de nossos modelos, especialmente para o SVM, os experimentos foram feitos com descarte de termos com frequência menor (em outras palavras, com valor de TF baixo).

Para o *dataset* do IMDb, primeiramente, foram feitos 20 experimentos com o modelo que utiliza o *Pipeline* 1 de processamento de textos (aplicação de *stemming*, representando o texto por *Bag-of-Words*). Nestes experimentos, utilizou-se o classificador não linear, com função *kernel* do tipo radial, $\sigma = 0.1$ e demais parâmetros de acordo com o padrão da biblioteca usada. Nos dez primeiros experimentos, foram desconsideradas as palavras com $TF \leq 4$, e nos outros dez, com $TF \leq 5$.

Como pode ser visto nas Tabelas 12 e 13, com estas configurações, o treino do modelo não atingiu resultados satisfatórios. Em ambos os treinos do modelo, utilizou-se cerca de somente 35% dos atributos presentes nos *subdatasets* completos, e o F_1 Score atingido foi de apenas 0,31, aproximadamente.

Tabela 12: Primeiros experimentos realizados com *Support Vector Machine*, utilizando o *dataset* do site IMDb e função *kernel* radial

TFs Descartados	Acurácia	F_1 Score	Precision	Recall	Atributos
≤ 4	0,5347	0,3100	0,7122	0,3315	13420,2
≤ 5	0,5356	0,3094	0,7127	0,3264	12083,1

Fonte: a autora.

Nas matrizes de confusão, como se pode notar pela Tabela 13, os números de falsos negativos e falsos positivos foram bem altos.

Tabela 13: Matrizes de confusão dos primeiros experimentos com *Support Vector Machine*, sobre o *dataset* do site IMDb, utilizando função *kernel* radial

TF ≤ 4		TF ≤ 5	
1644,9	3343,6	1621,2	3367,3
1309,1	3702,4	1277	3734,5

Fonte: a autora.

Dado este cenário, para os experimentos seguintes com todos os quatro modelos, adotou-se o classificador linear. Além disso, apenas as palavras com $TF = 1$ foram desconsideradas na representação dos textos. Tais experimentos geraram resultados bem melhores, como visto na Tabela 14.

Ao remover as palavras de $TF = 1$, os Modelos 1 e 2 utilizaram aproximadamente 62% dos atributos gerados com os *subdatasets* originais, enquanto que os Modelos 3 e 4 usa-

ram cerca de 59%. Com isso, atingiram-se resultados próximos a 0,87 de F_1 Score, usando a representação por TF-IDF, enquanto que os modelos que envolviam *Bag-of-Words* obtiveram resultados também bons, mas com F_1 Score um pouco inferior.

Tabela 14: Experimentos realizados com *Support Vector Machine*, utilizando o *dataset* do site IMDb

Modelo	Pipeline	Acurácia	F_1 Score	Precision	Recall	Atributos
1	Stemming + BoW	0,8345	0,8344	0,8326	0,8366	22418,5
2	Stemming + TF-IDF	0,8729	0,8740	0,8639	0,8845	22418,5
3	Lematização + BoW	0,8356	0,8356	0,8333	0,8383	26939,9
4	Lematização + TF-IDF	0,8726	0,8738	0,8630	0,8852	26939,9

Fonte: a autora.

Pode-se observar pela Tabela 15 que os Modelos 2 e 4 obtiveram os menores números de falsos positivos e falsos negativos.

Tabela 15: Matrizes de confusão dos experimentos com *Support Vector Machine*, sobre o *dataset* do site IMDb

Modelo 1		Modelo 2		Modelo 3		Modelo 4	
4172,7	815,8	4412,1	576,4	4181,3	807,2	4415,3	573,2
839,2	4172,3	694,9	4316,6	836,5	4175	700,9	4310,6

Fonte: a autora.

Com o *dataset* do *Rotten Tomatoes*, os modelos foram parametrizados com função *kernel* radial, $\sigma = 0.1$, e demais parâmetros de acordo com o *default* do Scikit-learn, gerando resultados de F_1 Score um pouco menores do que os gerados para o IMDb, com os melhores modelos obtendo F_1 Score de aproximadamente 0,79, conforme Tabela 16.

Novamente, assim como visto com os modelos implementados com o algoritmo *Naïve Bayes*, os melhores resultados foram obtidos pelos modelos com representação por *Bag-of-Words*.

Tabela 16: Experimentos realizados com *Support Vector Machine*, utilizando o *dataset* do site *Rotten Tomatoes*

Modelo	Pipeline	Acurácia	F_1 Score	Precision	Recall	Atributos
1	Stemming + BoW	0,7238	0,7980	0,7196	0,8958	7281,5
2	Stemming + TF-IDF	0,6590	0,7783	0,6444	0,9828	7281,5
3	Lematização + BoW	0,7190	0,7956	0,7144	0,8978	8186,5
4	Lematização + TF-IDF	0,6498	0,7740	0,6377	0,9847	8186,5

Fonte: a autora.

As matrizes de confusão também apresentaram grandes diferenças de valores entre si. Os Modelos 1 e 3 apresentaram os menores números de falsos positivos, enquanto que os Modelos 2 e 4 obtiveram bem menos falsos negativos.

Tabela 17: Matrizes de confusão dos experimentos com *Support Vector Machine*, sobre o *dataset* do site *Rotten Tomatoes*

Modelo 1		Modelo 2		Modelo 3		Modelo 4	
5460	634,9	5990,1	104,8	5472,3	622,6	6001,8	93,1
2127,6	1777,5	3305,2	599,9	2187,4	1717,7	3409,3	495,8

Fonte: a autora.

5.3.3 Árvore de Decisão

Para o uso deste algoritmo, o único parâmetro definido pela autora deste trabalho, ao utilizar o classificador, foi a métrica para o “*goodness of split*”, conforme visto na Seção 2.3.4.1. Para este trabalho, a métrica escolhida foi o índice de Gini. Novamente, assim como para os algoritmos demonstrados até o momento, os outros parâmetros para a Árvore de Decisão foram configurados de acordo com os padrões definidos pela biblioteca Scikit-learn.

Com a Árvore de Decisão, para o *dataset* do IMDb, os resultados obtidos foram um pouco menores do que com os dois algoritmos anteriores, apresentando F_1 Score de aproximadamente 0,71. Tanto com *Bag-of-Words* quanto com TF-IDF, os resultados foram semelhantes entre os quatro modelos experimentados, como visto na Tabela 18.

Tabela 18: Experimentos realizados com Árvore de Decisão, utilizando o *dataset* do site IMDb

Modelo	Pipeline	Acurácia	F_1 Score	Precision	Recall	Atributos
1	Stemming + BoW	0,7110	0,7107	0,7095	0,7124	36218,8
2	Stemming + TF-IDF	0,7050	0,7038	0,7048	0,7033	36218,8
3	Lematização + BoW	0,7081	0,7074	0,7072	0,7080	45340
4	Lematização + TF-IDF	0,7052	0,7041	0,7047	0,7041	45340

Fonte: a autora.

Nas matrizes de confusão vistas na Tabela 19, os resultados também foram muito próximos entre si, com os Modelos 1 e 3 apresentando números de falsos positivos e falsos negativos levemente abaixo dos obtidos pelos Modelos 2 e 4.

Tabela 19: Matrizes de confusão dos experimentos com Árvore de Decisão, sobre o *dataset* do site IMDb

Modelo 1		Modelo 2		Modelo 3		Modelo 4	
3553,5	1435	3507,5	1481	3531,9	1456,6	3512,4	1476,1
1454,9	3556,6	1468,9	3542,6	1462	3549,5	1471,9	3539,6

Fonte: a autora.

Por sua vez, para o *dataset* do Rotten Tomatoes, todos os resultados de F_1 Score se aproximaram de 0,70, com os modelos que aplicaram a técnica de *stemming* durante a etapa de processamento de linguagem natural obtendo os maiores resultados, por alguns décimos. As acurácias, por sua vez, foram menores do que as obtidas com o *dataset* do IMDb.

Tabela 20: Experimentos realizados com Árvore de Decisão, utilizando o *dataset* do site Rotten Tomatoes

Modelo	Pipeline	Acurácia	F_1 Score	Precision	Recall	Atributos
1	Stemming + BoW	0,6434	0,7073	0,7074	0,7080	12060,2
2	Stemming + TF-IDF	0,6375	0,7044	0,6999	0,7095	12060,2
3	Lematização + BoW	0,6373	0,7027	0,7018	0,7045	14665,1
4	Lematização + TF-IDF	0,6295	0,6984	0,6928	0,7049	14665,1

Fonte: a autora.

Nas matrizes de confusão presentes na Tabela 21, observa-se que o Modelo 2 obteve o menor número de falsos negativos, enquanto que o Modelo 1 foi o que obteve menos falsos positivos.

Tabela 21: Matrizes de confusão dos experimentos com Árvore de Decisão, sobre o *dataset* do site Rotten Tomatoes

Modelo 1		Modelo 2		Modelo 3		Modelo 4	
4315,3	1779,6	4324,9	1770	4293,9	1801	4296,3	1798,6
1786,9	2118,2	1854,7	2050,4	1825,8	2079,3	1906,3	1998,8

Fonte: a autora.

5.3.4 Random Forest

Para o *Random Forest*, os parâmetros definidos foram a métrica para o “goodness of split” e a quantidade de estimadores usados (em outras palavras, quantidade de Árvores de Decisão) no treino do modelo. Neste caso, também se utilizou o índice de Gini como métrica,

com outros parâmetros configurados com o *default* da biblioteca Scikit-learn, e cada modelo foi treinado com 10 estimadores.

No caso do IMDb, os resultados foram bastante semelhantes entre si, com os maiores valores sendo obtidos pela representação de texto por *Bag-of-Words*, com F_1 Score de aproximadamente 0,72, conforme Tabela 22.

Tabela 22: Experimentos realizados com *Random Forest*, utilizando o *dataset* do site IMDb

Modelo	Pipeline	Acurácia	F_1 Score	Precision	Recall	Atributos
1	Stemming + BoW	0,7481	0,7271	0,7905	0,6737	36218,8
2	Stemming + TF-IDF	0,7431	0,7205	0,7874	0,6647	36218,8
3	Lematização + BoW	0,7487	0,7271	0,7927	0,6720	45340
4	Lematização + TF-IDF	0,7436	0,7213	0,7878	0,6657	45340

Fonte: a autora.

Já nas matrizes de confusão, observa-se que o Modelo 1 obteve a menor quantidade de falsos negativos, e o Modelo 3, de falsos positivos.

Tabela 23: Matrizes de confusão dos experimentos com *Random Forest*, sobre o *dataset* do site IMDb

Modelo 1		Modelo 2		Modelo 3		Modelo 4	
3359,8	1628,7	3314,6	1673,9	3351,6	1636,9	3319,9	1668,6
890,7	4120,8	895,5	4116	876	4135,5	895,3	4116,2

Fonte: a autora.

Enquanto isso, para o *dataset* do *Rotten Tomatoes*, apesar de as acurácias terem sido menores, os valores de F_1 Score obtidos com este algoritmo foram melhores do que com o conjunto de dados anterior, próximos de 0,75. Neste caso, o uso de TF-IDF levou a uma pequena melhoria, em relação ao uso de *Bag-of-Words*.

Tabela 24: Experimentos realizados com *Random Forest*, utilizando o *dataset* do site *Rotten Tomatoes*

Modelo	Pipeline	Acurácia	F_1 Score	Precision	Recall	Atributos
1	Stemming + BoW	0,6840	0,7456	0,7318	0,7609	12060,2
2	Stemming + TF-IDF	0,6824	0,7514	0,7181	0,7890	12060,2
3	Lematização + BoW	0,6758	0,7395	0,7242	0,7565	14665,1
4	Lematização + TF-IDF	0,6743	0,7462	0,7104	0,7868	14665,1

Fonte: a autora.

Nas matrizes de confusão, por sua vez, os Modelos 1 e 3 apresentaram os menores números de falsos negativos, enquanto os Modelos 2 e 4 obtiveram as menores quantidades de falsos positivos, como se pode observar pela Tabela 25.

Tabela 25: Matrizes de confusão dos experimentos com *Random Forest*, sobre o *dataset* do site *Rotten Tomatoes*

Modelo 1		Modelo 2		Modelo 3		Modelo 4	
4637,9	1457	4809,2	1285,7	4611,7	1483,2	4796,1	1298,8
1702,8	2202,3	1890,8	2014,3	1758,7	2146,4	1957,8	1947,3

Fonte: a autora.

5.3.5 *k*-Nearest Neighbours

No desenvolvimento do modelo baseado no algoritmo de *k*-Nearest Neighbours, de todos os parâmetros exigidos pela biblioteca usada, dois deles foram configurados com valores específicos.

O primeiro deles foi o número de vizinhos considerados na classificação, sendo definido como $k = 5$, para os dois *datasets*. O outro parâmetro, por sua vez, foi o cálculo de distância entre os vizinhos. Para este trabalho, utilizou-se a distância euclidiana.

Considerando o *dataset* do IMDb, houve uma diferença significativa de desempenho entre os modelos que utilizam representação de texto por *Bag-of-Words* e TF-IDF. Os melhores resultados foram obtidos pelos modelos com TF-IDF, com F_1 Scores próximos de 0,74, conforme Tabela 26.

Tabela 26: Experimentos realizados com *k*-Nearest Neighbours, utilizando o *dataset* do site IMDb

Modelo	Pipeline	Acurácia	F_1 Score	Precision	Recall	Atributos
1	Stemming + BoW	0,6244	0,6267	0,6211	0,6384	36218,8
2	Stemming + TF-IDF	0,7363	0,7461	0,7176	0,7773	36218,8
3	Lematização + BoW	0,6281	0,6214	0,6307	0,6199	45340
4	Lematização + TF-IDF	0,7349	0,7425	0,7198	0,7672	45340

Fonte: a autora.

As matrizes de confusão referentes a este *dataset* também apresentam diferenças entre si, com os Modelos 2 e 4 apresentando os menores números tanto de falsos negativos, quanto de falsos positivos, como visto na Tabela 27.

Tabela 27: Matrizes de confusão dos experimentos com *k-Nearest Neighbours*, sobre o *dataset* do *site* IMDb

Modelo 1		Modelo 2		Modelo 3		Modelo 4	
3185,8	1802,7	3877,2	1111,3	3093,7	1894,8	3827	1161,5
1953,3	3058,2	1525,3	3486,2	1824,2	3187,3	1489,5	3522

Fonte: a autora.

Já para o *Rotten Tomatoes*, as acurácias foram menores (assim como ocorreu com os algoritmos anteriores), com os melhores valores de F_1 Scores próximos de 0,71. Novamente, os modelos que fazem uso de TF-IDF apresentaram melhor desempenho, em comparação com os modelos que usam *Bag-of-Words* nas representações de texto.

Tabela 28: Experimentos realizados com *k-Nearest Neighbours*, utilizando o *dataset* do *site* *Rotten Tomatoes*

Modelo	Pipeline	Acurácia	F_1 Score	Precision	Recall	Atributos
1	Stemming + BoW	0,5867	0,6477	0,6720	0,6417	12060,2
2	Stemming + TF-IDF	0,6624	0,7197	0,7246	0,7353	12060,2
3	Lematização + BoW	0,5775	0,6317	0,6713	0,6140	14665,1
4	Lematização + TF-IDF	0,6634	0,7189	0,7279	0,7311	14665,1

Fonte: a autora.

As matrizes de confusão também apresentam distinções, com os Modelos 2 e 4 obtendo as menores quantidades de falsos negativos e falsos positivos, conforme Tabela 29.

Tabela 29: Matrizes de confusão dos experimentos com *k-Nearest Neighbours*, sobre o *dataset* do *site* *Rotten Tomatoes*

Modelo 1		Modelo 2		Modelo 3		Modelo 4	
3913,8	2181,1	4484,8	1610,1	3744,7	2350,2	4459,2	1635,7
1951,7	1953,4	1765,7	2139,4	1874,8	2030,3	1730,6	2174,5

Fonte: a autora.

5.3.6 Multilayer Perceptron

Por fim, para o algoritmo *Multilayer Perceptron*, quatro parâmetros foram configurados, sendo a quantidade de camadas ocultas, número de neurônios por camada oculta, taxa inicial de aprendizado e número de épocas.

Apenas uma camada oculta contendo 50 neurônios foi utilizada nos quatro modelos. Além disso, foi definido um máximo de 200 épocas para o treino da rede neural, com uma taxa

inicial de aprendizado de 0.01. Assim como para todos os outros algoritmos apresentados, os demais parâmetros receberam as configurações padrão do Scikit-learn.

Os experimentos realizados com o MLP sobre o *dataset* do IMDb foram os que obtiveram alguns dos melhores resultados, com acurácias e F_1 Scores próximos de 0,86. Os modelos que aplicaram a técnica de lematização obtiveram uma pequena melhora em relação aos resultados obtidos com *stemming*, como visto na Tabela 30.

Tabela 30: Experimentos realizados com *Multilayer Perceptron*, utilizando o *dataset* do site IMDb

Modelo	Pipeline	Acurácia	F_1 Score	Precision	Recall	Atributos
1	Stemming + BoW	0,8600	0,8608	0,8536	0,8683	36218,8
2	Stemming + TF-IDF	0,8610	0,8615	0,8561	0,8672	36218,8
3	Lematização + BoW	0,8629	0,8634	0,8575	0,8697	45340
4	Lematização + TF-IDF	0,8632	0,8636	0,8585	0,8692	45340

Fonte: a autora.

As matrizes de confusão também são bastante semelhantes, com os menores valores de falsos negativos e falsos positivos sendo obtidos pelo Modelo 3.

Tabela 31: Matrizes de confusão dos experimentos com *Multilayer Perceptron*, sobre o *dataset* do site IMDb

Modelo 1		Modelo 2		Modelo 3		Modelo 4	
4331,2	657,3	4325,7	662,8	4338,5	650	4335,7	652,8
742,8	4268,7	727,5	4284	721,4	4290,1	715,3	4296,2

Fonte: a autora.

Encerrando os experimentos realizados neste trabalho, no *dataset* do *Rotten Tomatoes*, os melhores resultados de F_1 Score foram de aproximadamente 0,77, com a acurácia novamente um pouco menor em relação ao IMDb. Para os modelos deste algoritmo, o *Bag-of-Words* gerou uma pequena melhoria, em comparação com o TF-IDF.

Tabela 32: Experimentos realizados com *Multilayer Perceptron*, utilizando o *dataset* do site *Rotten Tomatoes*

Modelo	Pipeline	Acurácia	F_1 Score	Precision	Recall	Atributos
1	Stemming + BoW	0,7192	0,7735	0,7603	0,7876	12060,2
2	Stemming + TF-IDF	0,7101	0,7676	0,7501	0,7873	12060,2
3	Lematização + BoW	0,7172	0,7717	0,7592	0,7852	14665,1
4	Lematização + TF-IDF	0,7113	0,7696	0,7491	0,7921	14665,1

Fonte: a autora.

Para este conjunto de dados, nas matrizes de confusão presentes na Tabela 33, o Modelo 1 foi o responsável pelos menores números de falsos positivos e falsos negativos.

Tabela 33: Matrizes de confusão dos experimentos com *Multilayer Perceptron*, sobre o *dataset* do *site Rotten Tomatoes*

Modelo 1		Modelo 2		Modelo 3		Modelo 4	
4800,5	1175,6	4798,4	1296,5	4785,7	1309,2	4828,4	1266,5
1513,8	2391,3	1603	2302,1	1519,1	2386	1620,4	2082,7

Fonte: a autora.

5.4 Discussão dos resultados obtidos

Analisando-se o desempenho de todas as *features*, nota-se que a representação de textos por TF-IDF, em geral, levou aos melhores resultados. Como visto na Seção 2.2.3 e em Runeson, Alexandersson e Nyholm (2007), os termos que ocorrem em muitos documentos não são bons discriminadores da polaridade de um texto. Com isso, dá-se menos peso a estas palavras, em relação a termos que ocorrem em apenas alguns documentos, e tal ajuste de pesos foi determinante para o desempenho superior, em relação aos modelos que utilizaram *Bag-of-Words*.

Ao realizar uma avaliação dos métodos de geração de vetores de atributos a nível de *datasets*, para o conjunto de dados do IMDb, a maioria dos modelos que obtiveram os melhores resultados fizeram uso da representação por *Bag-of-Words* (em 4 de 6 classificadores). Enquanto isso, o TF-IDF foi o mais utilizado pelos melhores modelos treinados com o *dataset* do *Rotten Tomatoes* (também em 4 de 6 algoritmos), demonstrando que um método de representação de textos pode se adequar melhor a um conjunto de dados do que outros.

Considerando-se os seis classificadores utilizados neste trabalho, tanto para o *dataset* do IMDb quanto do *Rotten Tomatoes*, *Multilayer Perceptron* (MLP) e *Support Vector Machine* (SVM) foram os que mais se destacaram. Com o IMDb, o algoritmo MLP obteve F_1 Score próximo de 0,86, enquanto que o SVM obteve aproximadamente 0,87. Já com o *Rotten Tomatoes*, atingiram-se F_1 Scores de 0,77 e de 0,79 para o MLP e para o SVM, respectivamente.

Ambos os classificadores possuem funções matemáticas mais complexas e que se mostraram mais eficientes nesta aplicação. Os dois algoritmos fazem uso de *kernels*, responsáveis por aumentar a dimensionalidade do conjunto de dados e permitir, dessa forma, que as entradas fornecidas sejam melhor separadas entre si, impactando positivamente no aprendizado do modelo de classificação desenvolvido. O *Support Vector Machine* já havia sido explorado em alguns trabalhos da literatura, vistos na Seção 3, ao contrário do *Multilayer Perceptron*, que se mostrou bastante promissor para próximos trabalhos sobre o tópico de análise de sentimentos.

Ao analisar o desempenho em relação a cada um dos *datasets* utilizados neste presente

trabalho, nota-se que o do IMDb obteve acurácias maiores do que o conjunto de dados do *Rotten Tomatoes*, em todos os 24 tipos de modelos experimentados. Isso se deve ao fato de que o IMDb é um *dataset* totalmente balanceado, contendo 12.500 exemplos de cada polaridade, enquanto o *Rotten Tomatoes* possui classes desbalanceadas, seguindo uma proporção de 60% de exemplos positivos e 40% de negativos em todos os experimentos realizados.

Por fim, pode-se destacar o fato de que a quantidade de atributos presentes em cada conjunto de dados teve uma influência no custo computacional, mas não foi determinante para a acurácia obtida pelos seis classificadores. O *dataset* original do IMDb possuía 72.537 atributos, e o do *Rotten Tomatoes* possuía 21.474 atributos, no total. Porém, mesmo com esta discrepância nas quantidades, as acurácias não apresentaram tanta diferença entre um *dataset* e outro.

6 Considerações finais

Ao longo deste trabalho, desenvolveu-se um estudo aprofundado de todas as técnicas principais de processamento de textos, aprendizado de máquina e redes neurais, a serem aplicadas à tarefa de análise de sentimentos no contexto de críticas de filmes, disponíveis em *sites* como IMDb e *Rotten Tomatoes*, por exemplo. Após todos os experimentos realizados, pode-se concluir que tais algoritmos de aprendizado de máquina apresentam um bom desempenho nesta tarefa, com destaque para o *Multilayer Perceptron* e para o *Support Vector Machine*, com os quais foram obtidos os melhores resultados presentes neste estudo.

No decorrer dos experimentos, foram constatadas algumas dificuldades ao utilizar conjuntos de dados desbalanceados para treino dos modelos implementados, que geraram resultados passíveis de melhora. Para futuros trabalhos sobre este tema, pode ser interessante realizar testes com outros algoritmos de redes neurais, presentes em bibliotecas como Keras e TensorFlow, além de utilizar técnicas de detecção de ironia e sarcasmo, que podem ter um impacto significativo no desempenho do classificador, dependendo do *dataset* selecionado.

Outro ponto a se destacar é que já existe atualmente uma quantidade considerável de pesquisas na literatura que abordam análise de sentimentos presentes em opiniões sobre filmes, extraídas de redes sociais e de outros *sites* especializados. Tal fato demonstra a importância crescente da aplicação da análise de sentimentos não somente na área acadêmica, mas na área comercial, permitindo que as empresas façam uso de um *feedback* dado diretamente por seu público-alvo para produção de campanhas de *marketing* mais efetivas, para melhoria contínua da qualidade de seu atendimento, de seus produtos e, principalmente, da experiência que os clientes têm ao utilizar seus serviços.

Com este estudo, espera-se incentivar ainda mais a pesquisa e o desenvolvimento de ferramentas de inteligência de negócios e técnicas de análise textual, essenciais neste momento de crescente atenção às necessidades dos clientes, e de uso cada vez maior de dados pelas mais diversas áreas de pesquisa e atuação.

Referências

AALST, W. M. P. van der et al. Process mining: a two-step approach to balance between underfitting and overfitting. *Software & Systems Modeling*, v. 9, n. 1, p. 87–111, jan 2010. ISSN 1619-1366. Disponível em: <<http://link.springer.com/10.1007/s10270-008-0106-z>>. Citado na página 45.

ACKLEY, D. H.; HINTON, G. E.; SEJNOWSKI, T. J. A learning algorithm for boltzmann machines. *Cognitive Science*, v. 9, n. 1, p. 147–169, 1985. ISSN 03640213. Citado na página 72.

AMOLIK, A. et al. Twitter sentiment analysis of movie reviews using machine learning techniques. *International Journal of Engineering and Technology*, v. 7, n. 6, p. 1–7, 2016. Citado 2 vezes nas páginas 86 e 88.

ARORA, D.; LI, K. F.; NEVILLE, S. W. Consumers' sentiment analysis of popular phone brands and operating system preference using twitter data: A feasibility study. *Proceedings - International Conference on Advanced Information Networking and Applications, AINA*, IEEE, v. 2015-April, p. 680–686, 2015. ISSN 1550445X. Citado na página 30.

AURIA, L.; MORO, R. A. Support Vector Machines (SVM) as a Technique for Solvency Analysis. *SSRN Electronic Journal*, v. 811, n. August, 2008. ISSN 1556-5068. Disponível em: <<http://www.ssrn.com/abstract=1424949>>. Citado na página 57.

BASHEER, I.; HAJMEER, M. Artificial neural networks: fundamentals, computing, design, and application. *Journal of Microbiological Methods*, v. 43, n. 1, p. 3–31, dec 2000. ISSN 01677012. Disponível em: <[https://linkinghub.elsevier.com/retrieve/pii-S0167701200002013](https://linkinghub.elsevier.com/retrieve/pii/S0167701200002013)>. Citado na página 80.

BASUROY, S.; CHATTERJEE, S.; RAVID, S. A. How Critical Are Critical Reviews? The Box Office Effects of Film Critics, Star Power, and Budgets. *Journal of Marketing*, American Marketing Association, v. 67, n. 4, p. 103–117, 2003. ISSN 00222429. Disponível em: <<http://www.jstor.org/stable/30040552>>. Citado na página 27.

BENGIO, Y. et al. A Neural Probabilistic Language Model. *Journal of Machine Learning Research*, v. 3, n. 1, p. 1137–1155, 2003. ISSN 15364046. Disponível em: <<http://www.jmlr.org/papers/v3/bengio03a.html>>. Citado na página 37.

BHOIR, P.; KOLTE, S. Sentiment analysis of movie reviews using lexicon approach. In: *2015 IEEE International Conference on Computational Intelligence and Computing Research (ICIC)*. IEEE, 2015. p. 1–6. ISBN 978-1-4799-7848-9. Disponível em: <<http://ieeexplore.ieee.org/document/7435796/>>. Citado 2 vezes nas páginas 85 e 88.

BOSER, E. et al. A training algorithm for optimal margin classifiers. In: *ACM. Proceedings of the fifth annual workshop on Computational learning theory*. 1992. p. 144–152. Disponível em: <<https://dl.acm.org/citation.cfm?id=130401>>. Citado 2 vezes nas páginas 47 e 49.

BOULIS, C.; OSTENDORF, M. Text Classification by Augmenting the Bag-of-Words Representation with Redundancy-Compensated Bigrams. In: *Proceedings of the SIAM International Conference on Data Mining at the Workshop on Feature Selection in Data*

Mining (SIAM-FSDM 2005). [s.n.], 2005. Disponível em: <<http://www.icsi.berkeley.edu/pubs/speech/bagofwords05.pdf>>. Citado na página 35.

BRAGA, A. d. P.; LUDERMIR, T. B.; CARVALHO, A. C. P. d. L. F. de. *Redes Neurais Artificiais: Teoria e Aplicações*. 1. ed. Rio de Janeiro: LTC, 2000. 262 p. Citado 3 vezes nas páginas 62, 63 e 80.

BREIMAN, L. Bagging Predictors. *Machine Learning*, v. 24, n. 2, p. 123–140, aug 1996. ISSN 0885-6125. Disponível em: <<http://link.springer.com/10.1007/BF00058655>>. Citado na página 60.

BREIMAN, L. Random Forests. *Machine Learning*, v. 45, n. 1, p. 5–32, oct 2001. ISSN 1573-0565. Disponível em: <<https://doi.org/10.1023/A:1010933404324>>. Citado 2 vezes nas páginas 60 e 61.

BROWN, T. Hebbian Synapses: Biophysical Mechanisms And Algorithms. *Annual Review of Neuroscience*, v. 13, n. 1, p. 475–511, 1990. ISSN 0147006X. Citado na página 70.

BUCKLEY, C.; SALTON, G.; ALLAN, J. The SMART Information Retrieval Project. *Proceedings of the workshop on Human Language Technology*, p. 392–392, 1993. Citado na página 32.

BURGES, C. J. A Tutorial on Support Vector Machines for Pattern Recognition. *Data Mining and Knowledge Discovery*, v. 2, p. 121–167, 1998. ISSN 13845810. Disponível em: <<https://link.springer.com/article/10.1023/A:1009715923555>>. Citado 3 vezes nas páginas 51, 52 e 57.

CAMPBELL, C. An Introduction to Kernel Methods. *Studies in Fuzziness and Soft Computing*, v. 66, p. 155–192, 2001. Citado na página 50.

CHANGEUX, J.-P.; DANCHIN, A. Selective stabilisation of developing synapses as a mechanism for the specification of neuronal networks. *Nature*, v. 264, n. 5588, p. 705–712, 1976. ISSN 1476-4687. Disponível em: <<https://doi.org/10.1038/264705a0>>. Citado na página 70.

CHEN, S. F.; GOODMAN, J. An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, Elsevier, v. 13, n. 4, p. 359–394, 1999. Citado na página 87.

CHURCHLAND, P. S.; SEJNOWSKI, T. J. *The Computational Brain*. 1st. ed. Cambridge, MA, USA: MIT Press, 1992. ISBN 0262531208. Citado na página 68.

COMSCORE; The Kelsey Group. Online Consumer-Generated Reviews Have Significant Impact on Offline Purchase Behavior. *Comscore*, Reston, VA, nov 2007. Disponível em: <<https://www.comscore.com/Insights/Press-Releases/2007/11/Online-Consumer-Reviews-Impact-Offline-Purchasing-Behavior>>. Citado na página 27.

CORTES, C.; VAPNIK, V. Support-vector networks. *Machine learning*, Springer, v. 20, n. 3, p. 273–297, 1995. Citado na página 49.

CRISTIANINI, N.; SHAWE-TAYLOR, J. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge: Cambridge University Press, 2000. ISBN 9780511801389. Disponível em: <<http://ebooks.cambridge.org/ref/id/CBO9780511801389>>. Citado na página 47.

DOMINGOS, P.; PAZZANI, M. On the Optimality of the Simple Bayesian Classifier under Zero-One Loss. *Machine Learning*, v. 29, n. 2, p. 103–130, nov 1997. ISSN 1573-0565. Disponível em: <<https://doi.org/10.1023/A:1007413511361>>. Citado na página 47.

DU, W.; ZHAN, Z. Building Decision Tree Classifier on Private Data. In: *Proceedings of the IEEE International Conference on Privacy, Security and Data Mining - Volume 14*. Darlinghurst, Australia, Australia: Australian Computer Society, Inc., 2002. (CRPIT '14), p. 1–8. ISBN 0-909-92592-5. ISSN 0034-4257. Disponível em: <<http://dl.acm.org/citation.cfm?id=850782.850784>>. Citado 2 vezes nas páginas 58 e 60.

ESULI, A.; SEBASTIANI, F. SentiWordNet: A Publicly Available Lexical Resource for Opinion Mining. In: *Proceedings of the Fifth International Conference on Language Resources and Evaluation, LREC 2006*. [s.n.], 2006. p. 417–422. ISBN 2-9517408-6-7. ISSN 09255273. Disponível em: <<http://nmis.isti.cnr.it/sebastiani/Publications/LREC06.pdf>>. Citado na página 85.

FACELI, K. et al. *Inteligência Artificial: Uma Abordagem de Aprendizado de Máquina*. Rio de Janeiro: LTC, 2011. 394 p. ISBN 9788521618805. Citado 21 vezes nas páginas 28, 42, 46, 48, 49, 53, 54, 55, 56, 57, 59, 61, 62, 63, 64, 66, 67, 68, 69, 74 e 80.

FORBES, A. D. Classification-algorithm evaluation: Five performance measures based on confusion matrices. *Journal of Clinical Monitoring*, v. 11, n. 3, p. 189–206, may 1995. ISSN 0748-1977. Disponível em: <<http://link.springer.com/10.1007/BF01617722>>. Citado na página 80.

GAN, Q. et al. A Text Mining and Multidimensional Sentiment Analysis of Online Restaurant Reviews. *Journal of Quality Assurance in Hospitality & Tourism*, Routledge, v. 18, n. 4, p. 465–492, 2017. Disponível em: <<https://doi.org/10.1080/1528008X.2016.1250243>>. Citado na página 30.

GARDNER, M.; DORLING, S. Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. *Atmospheric Environment*, v. 32, n. 14-15, p. 2627–2636, aug 1998. ISSN 13522310. Disponível em: <<https://linkinghub.elsevier.com/retrieve/pii/S1352231097004470>>. Citado na página 79.

GUTMANN, M. U.; HYVÄRINEN, A. Noise-contrastive Estimation of Unnormalized Statistical Models, with Applications to Natural Image Statistics. *J. Mach. Learn. Res.*, JMLR.org, Boston, MA, v. 13, n. 1, p. 307–361, feb 2012. ISSN 1532-4435. Disponível em: <<http://dl.acm.org/citation.cfm?id=2503308.2188396>>. Citado na página 41.

GUZMAN, E.; MAALEJ, W. How do users like this feature? A fine grained sentiment analysis of App reviews. *2014 IEEE 22nd International Requirements Engineering Conference, RE 2014 - Proceedings*, IEEE, p. 153–162, 2014. Citado 3 vezes nas páginas 30, 86 e 88.

HAN, J.; KAMBER, M.; PEI, J. *Data Mining: Concepts and Techniques*. 3. ed. [S.l.]: Elsevier, 2012. 744 p. (Data Management Systems Series). ISBN 9780123814791. Citado 3 vezes nas páginas 81, 82 e 83.

HAWKINS, D. M. The Problem of Overfitting. *Journal of Chemical Information and Computer Sciences*, v. 44, n. 1, p. 1–12, 2004. Disponível em: <<https://doi.org/10.1021/ci0342472>>. Citado na página 61.

HAYKIN, S. *Neural Networks: A Comprehensive Foundation*. 1st. ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1994. ISBN 0023527617. Citado 16 vezes nas páginas 54, 57, 63, 64, 67, 69, 70, 71, 73, 74, 75, 76, 77, 78, 79 e 80.

HAYKIN, S. *Neural Networks and Learning Machines, 3/E*. [S.l.]: Pearson Education India, 2010. Citado na página 75.

HEARST, M. A. Direction-based text interpretation as an information access refinement. *Text-based intelligent systems: current research and practice in information extraction and retrieval*, Lawrence Erlbaum Associates, Mahwah, NJ, p. 257–274, 1992. Citado na página 29.

HEARST, M. A. et al. Support vector machines. *IEEE Intelligent Systems and their applications*, IEEE, v. 13, n. 4, p. 18–28, 1998. Citado na página 54.

HEBB, D. O. *The organization of behavior: A neuropsychological theory*. New York: Wiley, 1949. Hardcover. ISBN 0-8058-4300-0. Citado na página 70.

HERBRICH, R. *Learning Kernel Classifiers: Theory and Algorithms*. The MIT Press, 2002. 382 p. ISSN 08936080. ISBN 026208306X. Disponível em: <<http://linkinghub.elsevier.com/retrieve/pii/S0893608002000862>>. Citado na página 56.

HINTON, G. E. Deterministic boltzmann learning performs steepest descent in weight-space. *Neural Computation*, v. 1, n. 1, p. 143–150, March 1989. ISSN 0899-7667. Citado na página 79.

HINTON, G. E.; SEJNOWSKI, T. J. Learning and Relearning in Boltzmann Machines. In: RUMELHART, D. E.; MCCLELLAND, J. L.; PDP Research Group, C. (Ed.). *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*. Cambridge, MA, USA: MIT Press, 1986. v. 20, n. 1, cap. Learning a, p. 282–317. ISBN 0-262-68053-X. Disponível em: <<http://dl.acm.org/citation.cfm?id=104279.104291>>. Citado 2 vezes nas páginas 72 e 73.

HIRSCHBERG, J.; MANNING, C. D. Advances in natural language processing. *Science*, v. 349, n. 6245, p. 261–266, jul 2015. ISSN 0036-8075. Disponível em: <<http://www.sciencemag.org/cgi/doi/10.1126/science.aaa8685>>. Citado na página 28.

HODEGHATTA, U. R. Sentiment analysis of Hollywood movies on Twitter. In: *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining - ASONAM '13*. New York, New York, USA: ACM Press, 2013. p. 1401–1404. ISBN 9781450322409. Disponível em: <<http://dl.acm.org/citation.cfm?doid=2492517.2500290>>. Citado 2 vezes nas páginas 86 e 88.

HORRIGAN, J. B. *Online Shopping*. Washington, D.C.: [s.n.], 2008. 32 p. Disponível em: <<https://www.pewinternet.org/2008/02/13/online-shopping/>>. Citado na página 27.

KAELBLING, L. P.; LITTMAN, M. L.; MOORE, A. W. Reinforcement Learning: A Survey. *Journal of Artificial Intelligence Research*, v. 4, p. 237–285, may 1996. ISSN 1076-9757. Disponível em: <<https://jair.org/index.php/jair/article/view/10166>>. Citado na página 42.

KANG, H.; YOO, S. J.; HAN, D. Senti-lexicon and improved Naïve Bayes algorithms for sentiment analysis of restaurant reviews. *Expert Systems with Applications*, Elsevier Ltd, v. 39, n. 5, p. 6000–6010, 2012. ISSN 09574174. Disponível em: <<http://dx.doi.org/10.1016/j.eswa.2011.11.107>>. Citado 3 vezes nas páginas 30, 87 e 88.

KARNIN, E. D. A simple procedure for pruning back-propagation trained neural networks. *IEEE Transactions on Neural Networks*, v. 1, n. 2, p. 239–242, June 1990. ISSN 1045-9227. Citado na página 78.

KASPER, W.; VELA, M. Sentiment Analysis for Hotel Reviews. *Proceedings of the Computational Linguistics-Applications Conference*, v. 231527, n. March, p. 45–52, 2011. Citado 3 vezes nas páginas 30, 87 e 88.

KONONENKO, I. Semi-naive Bayesian classifier. In: KODRATOFF, Y. (Ed.). *Machine Learning — EWSL-91*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1991. p. 206–219. ISBN 978-3-540-46308-5. Citado na página 47.

KORENIUS, T. et al. Stemming and lemmatization in the clustering of finnish text documents. In: *Proceedings of the Thirteenth ACM conference on Information and knowledge management - CIKM '04*. New York, New York, USA: ACM Press, 2004. p. 625. ISBN 1581138741. Disponível em: <<http://portal.acm.org/citation.cfm?doid=1031171.1031285>>. Citado 2 vezes nas páginas 33 e 34.

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, 2007. v. 31, p. 3–24. ISBN 978-1-58603-780-2. Disponível em: <<http://dl.acm.org/citation.cfm?id=1566770.1566773>>. Citado na página 62.

KOUMPOURI, A.; MPORAS, I.; MEGALOOIKONOMOU, V. Evaluation of Four Approaches for “Sentiment Analysis on Movie Reviews”. In: *Proceedings of the 16th International Conference on Engineering Applications of Neural Networks (INNS) - EANN '15*. New York, New York, USA: ACM Press, 2015. p. 1–5. ISBN 978-1-4503-3580-5. Disponível em: <<http://dl.acm.org/citation.cfm?doid=2797143.2797182>>. Citado 2 vezes nas páginas 85 e 88.

KRAMER, A. H.; SANGIOVANNI-VINCENTELLI, A. Efficient Parallel Learning Algorithms for Neural Networks. In: TOURETZKY, D. S. (Ed.). *Advances in Neural Information Processing Systems I*. Morgan-Kaufmann, 1989. p. 40–48. Disponível em: <<http://papers.nips.cc/paper/134-efficient-parallel-learning-algorithms-for-neural-networks.pdf>>. Citado na página 79.

KWON, N.; SHULMAN, S. W.; HOVY, E. Multidimensional text analysis for eRulemaking. In: *Proceedings of the 2006 national conference on Digital government research - dg.o '06*. New York, New York, USA: ACM Press, 2006. p. 157. Disponível em: <<http://portal.acm.org/citation.cfm?doid=1146598.1146649>>. Citado na página 30.

LAVER, M.; BENOIT, K.; GARRY, J. Extracting policy positions from political texts using words as data. *American Political Science Review*, Cambridge University Press, v. 97, n. 2, p. 311–331, 2003. Citado na página 30.

LIAU, B. Y.; TAN, P. P. Gaining customer knowledge in low cost airlines through text mining. *Industrial Management & Data Systems*, v. 114, n. 9, p. 1344–1359, 2014. Disponível em: <<https://doi.org/10.1108/IMDS-07-2014-0225>>. Citado 3 vezes nas páginas 30, 87 e 88.

- LIAW, A.; WIENER; MATHE. Classification and regression by randomForest. *Journal of Dental Research*, v. 83, n. 5, p. 434–438, 2004. ISSN 0022-0345. Disponível em: <<http://journals.sagepub.com/doi/10.1177/154405910408300516>>. Citado na página 60.
- LING, W. et al. Not All Contexts Are Created Equal: Better Word Representations with Variable Attention. In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Stroudsburg, PA, USA: Association for Computational Linguistics, 2015. p. 1367–1372. Disponível em: <<http://aclweb.org/anthology/D15-1161>>. Citado na página 37.
- LITA, L. V. et al. Qualitative Dimensions in Question Answering: Extending the Definitional QA Task. *Proceedings of the national conference on artificial intelligence*, v. 20, n. 4, p. 1616–1617, 2005. Citado na página 30.
- LIU, B.; ZHANG, L. A Survey of Opinion Mining and Sentiment Analysis. In: AGGARWAL, C. C.; ZHAI, C. (Ed.). *Mining Text Data*. Boston, MA: Springer US, 2012. p. 415–463. ISBN 978-1-4614-3223-4. Disponível em: <<http://www.cs.unibo.it/~montesi/CBD/Articoli-/SurveyOpinionMining.pdf>>. Citado na página 29.
- MAAS, A. L. et al. Learning Word Vectors for Sentiment Analysis. In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Portland, Oregon, USA: Association for Computational Linguistics, 2011. v. 2015, p. 142–150. ISBN 0918-6158. ISSN 17414288. Disponível em: <<http://www.aclweb.org/anthology/P11-1015>>. Citado 2 vezes nas páginas 92 e 99.
- MALOUF, R.; MULLEN, T. A Preliminary Investigation into Sentiment Analysis of Informal Political Discourse. *AAAI Spring Symposium: Computational Approaches to Analyzing Weblogs*, p. 159–162, apr 2006. Disponível em: <<https://www.aaai.org/Papers/Symposia-/Spring/2006/SS-06-03/SS06-03-031.pdf>>. Citado na página 30.
- MANNING, C. D.; SCHÜTZE, H. *Foundations of Statistical Natural Language Processing*. MIT Press, 1999. 680 p. (Mit Press). ISBN 9780262133609. Disponível em: <<https://books.google.com.br/books?id=YiFDxbEX3SUC>>. Citado na página 28.
- MÀRQUEZ, L.; RODRÍGUEZ, H. Part-of-speech tagging using decision trees. In: NÉDELLEC, C.; ROUVEIROL, C. (Ed.). *Machine Learning: ECML-98*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998. p. 25–36. ISBN 978-3-540-69781-7. Disponível em: <<http://link.springer.com/10.1007/BFb0026668>>. Citado na página 34.
- MAZUROWSKI, M. A. et al. Training neural network classifiers for medical decision making: The effects of imbalanced datasets on classification performance. *Neural Networks*, v. 21, n. 2-3, p. 427–436, mar 2008. ISSN 08936080. Disponível em: <<https://linkinghub.elsevier.com/retrieve/pii/S0893608007002407>>. Citado na página 44.
- MCCALLUM, A.; NIGAM, K. A comparison of event models for naive bayes text classification. In: CITESEER. *AAAI-98 workshop on learning for text categorization*. [S.l.], 1998. v. 752, n. 1, p. 41–48. Citado na página 104.
- MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, v. 5, n. 4, p. 115–133, 1943. ISSN 00074985. Citado na página 66.

- MICHALSKI, R. S.; KAUFMAN, K. A. Data Mining and Knowledge Discovery: A Review of Issues and a Multistrategy Approach. *Machine Learning and Data Mining: Methods and Applications*, v. 1, n. August 2013, p. 1–42, 1998. Disponível em: <<http://mars.gmu.edu/handle/1920/1834>>. Citado na página 43.
- MIKOLOV, T. et al. *Efficient Estimation of Word Representations in Vector Space*. 2013. Disponível em: <<http://arxiv.org/abs/1301.3781>>. Citado 4 vezes nas páginas 36, 38, 40 e 41.
- MIKOLOV, T. et al. Distributed Representations of Words and Phrases and Their Compositionality. In: *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*. USA: Curran Associates Inc., 2013. (NIPS'13), p. 3111–3119. Disponível em: <<http://dl.acm.org/citation.cfm?id=2999792.2999959>>. Citado 2 vezes nas páginas 36 e 40.
- MITCHELL, T. M. *The Need for Biases in Learning Generalizations*. [S.l.], 1980. Citado na página 44.
- MITCHELL, T. M. *Machine Learning*. 1. ed. New York, NY, USA: McGraw-Hill, Inc., 1997. 414 p. ISBN 0070428077, 9780070428072. Citado na página 28.
- MONARD, M. C.; BARANAUSKAS, J. A. Conceitos sobre Aprendizado de Máquina. In: *Sistemas Inteligentes: Fundamentos e Aplicações*. 1. ed. Barueri: Manole Ltda., 2003. cap. 4, p. 89–114. ISBN 85-204-168. Citado 4 vezes nas páginas 42, 43, 46 e 81.
- MORETTIN, P. A.; BUSSAB, W. d. O. *Estatística Básica*. 6. ed. São Paulo: Saraiva, 2010. 540 p. ISBN 9788502081772. Citado 2 vezes nas páginas 46 e 80.
- MORIN, F.; BENGIO, Y. Hierarchical Probabilistic Neural Network Language Model. In: COWELL, R. G.; GHAHRAMANI, Z. (Ed.). *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*. Society for Artificial Intelligence and Statistics, 2005. p. 246–252. ISSN 00900036. Disponível em: <<http://www.iro.umontreal.ca/~lisa/pointeurs-/hierarchical-nnml-aistats05.p>>. Citado na página 40.
- MÜLLER, K.-R. et al. An introduction to kernel-based learning algorithms. *Handbook of Neural Network Signal Processing*, v. 12, n. September, 2001. Citado na página 48.
- NANDA, C.; DUA, M.; NANDA, G. Sentiment Analysis of Movie Reviews in Hindi Language Using Machine Learning. In: *2018 International Conference on Communication and Signal Processing (ICCSP)*. IEEE, 2018. p. 1069–1072. ISBN 978-1-5386-3521-6. Disponível em: <<https://ieeexplore.ieee.org/document/8524223/>>. Citado 2 vezes nas páginas 86 e 88.
- ORENGO, V.; HUYCK, C. A stemming algorithm for the portuguese language. In: *Proceedings Eighth Symposium on String Processing and Information Retrieval*. IEEE, 2005. p. 186–193. ISBN 0-7695-1192-9. Disponível em: <<http://ieeexplore.ieee.org/document/989755/>>. Citado na página 34.
- OSUNA, E. E.; GIROSI, F. Reducing the Run-time Complexity in Support Vector Machines. In: SCHÖLKOPF, B.; BURGESS, C. J. C.; SMOLA, A. J. (Ed.). *Advances in Kernel Methods*. Cambridge, MA, USA: MIT Press, 1999. v. 113, n. 2, cap. Reducing t, p. 271–283. ISBN 0-262-19416-3. Disponível em: <<http://dl.acm.org/citation.cfm?id=299094.299109>>. Citado na página 57.

- PANG, B.; LEE, L. *Opinion Mining and Sentiment Analysis*. Now Publishers, 2008. 137 p. (Foundations and trends in information retrieval, v. 22). ISSN 9781601981509. ISBN 9781601981509. Disponível em: <<https://books.google.com.br/books?id=XQswsqLLKrEC>>. Citado 4 vezes nas páginas 27, 29, 30 e 34.
- PANG, B.; LEE, L.; VAITHYANATHAN, S. Thumbs Up?: Sentiment Classification Using Machine Learning Techniques. In: *Proceedings of the ACL-02 conference on Empirical methods in natural language processing - EMNLP '02*. Morristown, NJ, USA: Association for Computational Linguistics, 2002. v. 10, n. July, p. 79–86. ISSN 0003-5696. Disponível em: <<http://portal.acm.org/citation.cfm?doid=1118693.1118704>>. Citado 5 vezes nas páginas 28, 30, 85, 86 e 88.
- PARKER, D. B. Optimal algorithms for adaptive networks: Second order back propagation, second order direct propagation, and second order Hebbian learning. In: *Proceedings of the IEEE First International Conference on Neural Networks (San Diego, CA)*. [S.l.]: Piscataway, NJ: IEEE, 1987. II, p. 593–600. Citado na página 74.
- PENTHENY, J. R. *The Influence of Movie Reviews on Consumers*. 2015. Disponível em: <<https://scholars.unh.edu/honors/265>>. Citado na página 27.
- PORTER, M. An algorithm for suffix stripping. *Program*, v. 14, n. 3, p. 130–137, mar 1980. ISSN 0033-0337. Disponível em: <<http://www.emeraldinsight.com/doi/10.1108/eb046814>>. Citado na página 34.
- POUSHTER, J.; STEWART, R. Smartphone Ownership and Internet Usage Continues to Climb in Emerging Economies. *Pew Research Center*, n. February, p. 1–44, 2016. ISSN 02727358. Disponível em: <<https://www.pewglobal.org/2016/02/22/smartphone-ownership-and-internet-usage-continues-to-climb-in-emerging-economies/>>. Citado na página 27.
- RAINEE, L.; HERRIGAN, J. B. *Election 2006 Online*. Washington, D.C.: Pew Research Center, 2007. 25 p. Disponível em: <<https://www.pewinternet.org/2007/01/17/election-2006-online/>>. Citado na página 27.
- RAMOS, J. Using TF-IDF to determine word relevance in document queries. *New Educational Review*, v. 42, n. 4, p. 40–51, 2003. ISSN 17326729. Citado na página 36.
- RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. *Nature*, v. 323, n. 6088, p. 533–536, 1986. ISSN 1476-4687. Disponível em: <<https://doi.org/10.1038/323533a0>>. Citado na página 79.
- RUMELHART, D. E.; ZIPSER, D. Feature discovery by competitive learning. *Cognitive Science*, v. 9, n. 1, p. 75–112, mar 1985. ISSN 03640213. Disponível em: <[http://doi.wiley.com/10.1016/S0364-0213\(85\)80010-0](http://doi.wiley.com/10.1016/S0364-0213(85)80010-0)>. Citado na página 71.
- RUNESON, P.; ALEXANDERSSON, M.; NYHOLM, O. Detection of duplicate defect reports using natural language processing. In: *Proceedings of the 29th International Conference on Software Engineering*. Washington, DC, USA: IEEE Computer Society, 2007. (ICSE '07), p. 499–510. ISBN 0-7695-2828-7. Disponível em: <<https://doi.org/10.1109/ICSE.2007.32>>. Citado 2 vezes nas páginas 99 e 114.

RUSSELL, S.; NORVIG, P. *Artificial Intelligence: A Modern Approach*. 3rd. ed. Upper Saddle River, NJ, USA: Prentice Hall Press, 2009. 487–492 p. ISBN 0136042597, 9780136042594. Citado 4 vezes nas páginas 28, 47, 56 e 57.

SAHLGREN, M.; CÖSTER, R. Using Bag-of-concepts to Improve the Performance of Support Vector Machines in Text Categorization. In: *Proceedings of the 20th International Conference on Computational Linguistics*. Stroudsburg, PA, USA: Association for Computational Linguistics, 2004. (COLING '04). Disponível em: <<https://doi.org/10.3115/1220355-1220425>>. Citado na página 35.

SAHU, T. P.; AHUJA, S. Sentiment analysis of movie reviews: A study on feature selection & classification algorithms. In: *2016 International Conference on Microelectronics, Computing and Communications (MicroCom)*. IEEE, 2016. p. 1–6. ISBN 978-1-4673-6621-2. Disponível em: <<http://ieeexplore.ieee.org/document/7522583/>>. Citado 2 vezes nas páginas 85 e 88.

SALTON, G.; BUCKLEY, C. Term-Weighting Approaches in Automatic Text Retrieval. *Information Processing and Management*, v. 24, n. 5, p. 513–523, 1988. Citado na página 36.

SEJNOWSKI, T. J. Statistical constraints on synaptic plasticity. *Journal of Theoretical Biology*, v. 69, n. 2, p. 385–389, 1977. ISSN 10958541. Citado na página 71.

SEJNOWSKI, T. J. Storing covariance with nonlinearly interacting neurons. *Journal of Mathematical Biology*, v. 4, n. 4, p. 303–321, 1977. ISSN 14321416. Citado na página 71.

SHI, H. X.; LI, X. J. A sentiment analysis model for hotel reviews based on supervised learning. *Proceedings - International Conference on Machine Learning and Cybernetics*, IEEE, v. 3, p. 950–954, 2011. ISSN 2160133X. Citado na página 30.

SILGE, J.; ROBINSON, D. *Text Mining with R: A Tidy Approach*. O'Reilly, 2017. ISBN 9781491981658. Disponível em: <<https://books.google.com.br/books?id=7bQzMQAACAAJ>>. Citado na página 36.

SILVA, F. M.; ALMEIDA, L. B. Acceleration techniques for the backpropagation algorithm. In: ALMEIDA, L. B.; WELLEKENS, C. J. (Ed.). *Neural Networks*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1990. p. 110–119. ISBN 978-3-540-46939-1. Citado na página 78.

SINGH, V. K. et al. Sentiment analysis of movie reviews: A new feature-based heuristic for aspect-level sentiment classification. In: *2013 International Mutli-Conference on Automation, Computing, Communication, Control and Compressed Sensing (iMac4s)*. IEEE, 2013. p. 712–717. ISBN 978-1-4673-5090-7. Disponível em: <<http://ieeexplore.ieee.org/document/6526500/>>. Citado 3 vezes nas páginas 28, 85 e 88.

SMOLA, A. J.; SCHÖLKOPF, B. *A tutorial on support vector regression*. Netherlands, 2004. v. 14, n. 7, 199–222 p. Citado 3 vezes nas páginas 48, 51 e 52.

SRIVIDHYA, V.; ANITHA, R. Evaluating Preprocessing Techniques in Text Categorization. *International Journal of Computer Science and Application*, p. 49–51, 2010. ISSN 0974-0767. Citado na página 32.

STEFFEN, J. N-gram language modeling for robust multi-lingual document classification. In: *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC'04)*. Lisbon, Portugal: European Language Resources Association (ELRA), 2004.

Disponível em: <<http://www.lrec-conf.org/proceedings/lrec2004/pdf/510.pdf>>. Citado na página 87.

STENT, G. S. A Physiological Mechanism for Hebb's Postulate of Learning. *Proceedings of the National Academy of Sciences*, v. 70, n. 4, p. 997–1001, 1973. ISSN 0027-8424. Citado na página 70.

TABOADA, M. et al. Lexicon-Based Methods for Sentiment Analysis. *Computational Linguistics*, v. 37, n. 2, p. 267–307, jun 2011. ISSN 0891-2017. Disponível em: <https://www.mitpressjournals.org/doi/abs/10.1162/COLI_a_00049>. Citado na página 30.

TAN, P.-N. et al. Classification: Basic Concepts, and Techniques. In: *Introduction to Data Mining*. [S.l.]: Pearson, 2019. p. 839. Citado na página 82.

TATEMURA, J. Virtual reviewers for collaborative exploration of movie reviews. In: *Proceedings of the 5th international conference on Intelligent user interfaces*. [S.l.: s.n.], 2000. p. 272–275. Citado na página 30.

THELWALL, M. et al. Sentiment strength detection in short informal text. *Journal of the American Society for Information Science and Technology*, Wiley Online Library, v. 61, n. 12, p. 2544–2558, 2010. Citado na página 87.

TU, J. V. Advantages and disadvantages of using artificial neural networks versus logistic regression for predicting medical outcomes. *Journal of Clinical Epidemiology*, v. 49, n. 11, p. 1225–1231, 1996. ISSN 08954356. Citado na página 80.

UYSAL, A. K.; GUNAL, S. The impact of preprocessing on text classification. *Information Processing and Management*, Elsevier Ltd, v. 50, n. 1, p. 104–112, 2014. ISSN 03064573. Disponível em: <<http://dx.doi.org/10.1016/j.ipm.2013.08.006>>. Citado 2 vezes nas páginas 31 e 32.

VAPNIK, V. N. An overview of statistical learning theory. *IEEE Transactions on Neural Networks*, v. 10, n. 5, p. 988–999, 1999. ISSN 10459227. Citado na página 47.

WAN, Y.; GAO, Q. An ensemble sentiment classification system of twitter data for airline services analysis. In: *2015 IEEE International Conference on Data Mining Workshop (ICDMW)*. [S.l.: s.n.], 2015. p. 1318–1325. ISSN 2375-9259. Citado na página 30.

WHITELAW, C.; GARG, N.; ARGAMON, S. Using appraisal groups for sentiment analysis. In: *Proceedings of the 14th ACM international conference on Information and knowledge management - CIKM '05*. New York, New York, USA: ACM Press, 2005. p. 625. ISBN 1595931406. Disponível em: <<http://portal.acm.org/citation.cfm?doid=1099554.1099714>>. Citado 3 vezes nas páginas 85, 86 e 88.

WIDROW, B.; HOFF, M. E. *Adaptive switching circuits*. [S.l.], 1960. Citado na página 70.

WIEBE, J. M.; RAPAPORT, W. J. A Computational Theory of Perspective and Reference in Narrative. In: *Proceedings of the 26th Annual Meeting on Association for Computational Linguistics*. Stroudsburg, PA, USA: Association for Computational Linguistics, 1988. (ACL '88), p. 131–138. Disponível em: <<https://doi.org/10.3115/982023.982039>>. Citado na página 29.

WILKS, Y.; BIEN, J. Beliefs, points of view, and multiple environments. *Cognitive Science*, v. 7, n. 2, p. 95–119, 1983. ISSN 03640213. Citado na página 29.

WONG, T. T. Performance evaluation of classification algorithms by k-fold and leave-one-out cross validation. *Pattern Recognition*, Elsevier, v. 48, n. 9, p. 2839–2846, 2015. ISSN 00313203. Disponível em: <<http://dx.doi.org/10.1016/j.patcog.2015.03.009>>. Citado na página 81.

ZHANG, S. et al. Movie Short-Text Reviews Sentiment Analysis Based on Multi-Feature Fusion. In: *Proceedings of the 2018 International Conference on Algorithms, Computing and Artificial Intelligence - ACAI 2018*. New York, New York, USA: ACM Press, 2018. p. 1–6. ISBN 9781450366250. Disponível em: <<http://dl.acm.org/citation.cfm?doid=3302425-.3302469>>. Citado 3 vezes nas páginas 28, 86 e 88.

ZHANG, Z. et al. Sentiment classification of internet restaurant reviews written in cantonese. *Expert Systems with Applications*, Elsevier, v. 38, n. 6, p. 7674–7682, 2011. Citado 2 vezes nas páginas 87 e 88.

ZHU, X. J. *Semi-supervised learning literature survey*. [S.l.], 2005. Disponível em: <<http://digital.library.wisc.edu/1793/60444>>. Citado na página 42.

Apêndices

APÊNDICE A – Experimentos completos com IMDb

A.1 *Naïve Bayes*

PIPELINE 1: STEMMING + BAG-OF-WORDS											
SEMENTE	ATRIBUTOS	POSITIVOS	NEGATIVOS	MÉTRICAS				MATRIZ DE CONFUSÃO			
				ACURÁCIA	F1 SCORE	PRECISION	RECALL	TN	FP	FN	TP
0	36078	4965	5035	0.8404	0.8352	0.8566	0.8151	4358	677	919	4046
1	36380	5061	4939	0.8438	0.8402	0.8687	0.8137	4319	620	942	4119
2	36369	4969	5031	0.8386	0.8331	0.8560	0.8117	4353	678	936	4033
3	36263	5003	4997	0.8345	0.8307	0.8505	0.8121	4283	714	941	4062
4	36420	4954	5046	0.8436	0.8377	0.8630	0.8139	4404	642	922	4032
5	35886	5017	4983	0.8450	0.8420	0.8618	0.8233	4321	662	888	4129
6	35976	5029	4971	0.8405	0.8371	0.8604	0.8152	4306	665	930	4099
7	36313	4913	5087	0.8421	0.8360	0.8532	0.8199	4393	694	885	4028
8	35958	5008	4992	0.8424	0.8386	0.8615	0.8171	4334	658	918	4090
9	36545	4966	5034	0.8310	0.8248	0.8499	0.8013	4331	703	987	3979
	36218.8	4988.5	5011.5	0.8402	0.8355	0.8581	0.8144	4340.2	671.3	926.8	4061.7

PIPELINE 2: STEMMING + TF-IDF											
SEMENTE	ATRIBUTOS	POSITIVOS	NEGATIVOS	MÉTRICAS				MATRIZ DE CONFUSÃO			
				ACURÁCIA	F1 SCORE	PRECISION	RECALL	TN	FP	FN	TP
0	36078	4965	5035	0.8434	0.8360	0.8703	0.8048	4440	595	971	3994
1	36380	5061	4939	0.8557	0.8556	0.8653	0.8467	4273	666	777	4284
2	36369	4969	5031	0.8435	0.8359	0.8721	0.8031	4446	585	980	3989
3	36263	5003	4997	0.8409	0.8364	0.8618	0.8128	4345	652	939	4064
4	36420	4954	5046	0.8480	0.8393	0.8823	0.8004	4516	530	990	3964
5	35886	5017	4983	0.8496	0.8476	0.8620	0.8345	4313	670	834	4183
6	35976	5029	4971	0.8499	0.8473	0.8681	0.8279	4338	633	868	4161
7	36313	4913	5087	0.8426	0.8303	0.8827	0.7844	4573	514	1060	3853
8	35958	5008	4992	0.8476	0.8435	0.8691	0.8201	4373	619	905	4103
9	36545	4966	5034	0.8420	0.8345	0.8697	0.8023	4437	597	983	3983
	36218.8	4988.5	5011.5	0.8463	0.8406	0.8703	0.8137	4405.4	606.1	930.7	4057.8

PIPELINE 3: LEMATIZAÇÃO + BAG-OF-WORDS											
SEMENTE	ATRIBUTOS	POSITIVOS	NEGATIVOS	MÉTRICAS				MATRIZ DE CONFUSÃO			
				ACURÁCIA	F1 SCORE	PRECISION	RECALL	TN	FP	FN	TP
0	45139	4965	5035	0.8417	0.8363	0.8589	0.8151	4371	664	919	4046
1	45514	5061	4939	0.8415	0.8378	0.8670	0.8106	4312	627	958	4103
2	45641	4969	5031	0.8384	0.8326	0.8564	0.8104	4357	674	942	4027
3	45216	5003	4997	0.8369	0.8327	0.8555	0.8114	4311	686	945	4058
4	45585	4954	5046	0.8461	0.8399	0.8678	0.8138	4430	616	923	4031
5	44936	5017	4983	0.8424	0.8387	0.8616	0.8174	4325	658	918	4099
6	45153	5029	4971	0.8379	0.8339	0.8598	0.8098	4308	663	958	4071
7	45489	4913	5087	0.8446	0.8385	0.8567	0.8213	4411	676	878	4035
8	45048	5008	4992	0.8416	0.8375	0.8619	0.8149	4338	654	930	4078
9	45679	4966	5034	0.8322	0.8253	0.8546	0.7981	4359	675	1003	3963
	45340	4988.5	5011.5	0.8403	0.8353	0.8600	0.8123	4352.2	659.3	937.4	4051.1

PIPELINE 4: LEMATIZAÇÃO + TF-IDF											
SEMENTE	ATRIBUTOS	POSITIVOS	NEGATIVOS	MÉTRICAS				MATRIZ DE CONFUSÃO			
				ACURÁCIA	F1 SCORE	PRECISION	RECALL	TN	FP	FN	TP
0	45139	4965	5035	0.8450	0.8368	0.8766	0.8012	4475	560	990	3975
1	45514	5061	4939	0.8560	0.8558	0.8667	0.8455	4282	657	783	4278
2	45641	4969	5031	0.8438	0.8347	0.8790	0.7951	4488	543	1019	3950
3	45216	5003	4997	0.8480	0.8424	0.8760	0.8117	4422	575	945	4058
4	45585	4954	5046	0.8471	0.8369	0.8880	0.7915	4551	495	1034	3920
5	44936	5017	4983	0.8488	0.8459	0.8660	0.8276	4340	643	869	4148
6	45153	5029	4971	0.8477	0.8438	0.8716	0.8184	4364	607	916	4113
7	45489	4913	5087	0.8443	0.8312	0.8896	0.7808	4608	479	1078	3835
8	45048	5008	4992	0.8487	0.8438	0.8746	0.8157	4406	586	927	4081
9	45679	4966	5034	0.8420	0.8334	0.8751	0.7957	4470	564	1016	3950
	45340	4988.5	5011.5	0.8471	0.8405	0.8763	0.8083	4440.6	570.9	957.7	4030.8

Fonte: a autora.

A.2 Support Vector Machine

A.2.1 Classificador linear

PIPELINE 1: STEMMING + BAG-OF-WORDS											
SEMENTE	ATRIBUTOS	POSITIVOS	NEGATIVOS	MÉTRICAS				MATRIZ DE CONFUSÃO			
				ACURÁCIA	F1 SCORE	PRECISION	RECALL	TN	FP	FN	TP
0	22326	4965	5035	0.8326	0.8317	0.8296	0.8340	4185	850	824	4141
1	22544	5061	4939	0.8365	0.8385	0.8370	0.8402	4113	826	809	4252
2	22504	4969	5031	0.8379	0.8375	0.8340	0.8412	4199	832	789	4180
3	22320	5003	4997	0.8361	0.8365	0.8343	0.8390	4164	833	806	4197
4	22519	4954	5046	0.8373	0.8356	0.8370	0.8345	4240	806	821	4133
5	22312	5017	4983	0.8380	0.8388	0.8364	0.8413	4159	824	796	4221
6	22166	5029	4971	0.8308	0.8326	0.8288	0.8369	4101	870	822	4207
7	22594	4913	5087	0.8337	0.8306	0.8312	0.8301	4259	828	835	4078
8	22419	5008	4992	0.8357	0.8367	0.8332	0.8407	4148	844	799	4209
9	22481	4966	5034	0.8264	0.8256	0.8240	0.8277	4155	879	857	4109
	22418.5	4988.5	5011.5	0.8345	0.8344	0.8326	0.8366	4172.3	839.2	815.8	4172.7

PIPELINE 2: STEMMING + TF-IDF											
SEMENTE	ATRIBUTOS	POSITIVOS	NEGATIVOS	MÉTRICAS				MATRIZ DE CONFUSÃO			
				ACURÁCIA	F1 SCORE	PRECISION	RECALL	TN	FP	FN	TP
0	22326	4965	5035	0.8686	0.8691	0.8589	0.8797	4318	717	597	4368
1	22544	5061	4939	0.8759	0.8784	0.8693	0.8879	4265	674	567	4494
2	22504	4969	5031	0.8728	0.8726	0.8672	0.8784	4363	668	604	4365
3	22320	5003	4997	0.8717	0.8737	0.8602	0.8880	4275	722	561	4442
4	22519	4954	5046	0.8734	0.8734	0.8655	0.8816	4367	679	587	4367
5	22312	5017	4983	0.8760	0.8782	0.8643	0.8928	4281	702	538	4479
6	22166	5029	4971	0.8735	0.8758	0.8653	0.8870	4276	695	570	4459
7	22594	4913	5087	0.8732	0.8723	0.8636	0.8813	4403	684	584	4329
8	22419	5008	4992	0.8763	0.8781	0.8676	0.8891	4311	681	556	4452
9	22481	4966	5034	0.8673	0.8681	0.8573	0.8794	4307	727	600	4366
	22418.5	4988.5	5011.5	0.8729	0.8740	0.8639	0.8845	4316.6	694.9	576.4	4412.1

PIPELINE 3: LEMATIZAÇÃO + BAG-OF-WORDS											
SEMENTE	ATRIBUTOS	POSITIVOS	NEGATIVOS	MÉTRICAS				MATRIZ DE CONFUSÃO			
				ACURÁCIA	F1 SCORE	PRECISION	RECALL	TN	FP	FN	TP
0	26818	4965	5035	0.8353	0.8344	0.8317	0.8375	4194	841	806	4159
1	27041	5061	4939	0.8398	0.8418	0.8405	0.8432	4130	809	793	4268
2	27081	4969	5031	0.8398	0.8395	0.8362	0.8433	4209	822	780	4189
3	26851	5003	4997	0.8395	0.8403	0.8360	0.8450	4168	829	776	4227
4	26985	4954	5046	0.8365	0.8356	0.8325	0.8392	4208	838	797	4157
5	26783	5017	4983	0.8365	0.8371	0.8356	0.8388	4157	826	809	4208
6	26652	5029	4971	0.8278	0.8296	0.8258	0.8339	4086	885	837	4192
7	27173	4913	5087	0.8329	0.8299	0.8296	0.8304	4249	838	833	4080
8	26966	5008	4992	0.8373	0.8381	0.8356	0.8410	4162	830	797	4211
9	27049	4966	5034	0.8309	0.8298	0.8296	0.8302	4187	847	844	4122
	26939.9	4988.5	5011.5	0.8356	0.8356	0.8333	0.8383	4175	836.5	807.2	4181.3

PIPELINE 4: LEMATIZAÇÃO + TF-IDF											
SEMENTE	ATRIBUTOS	POSITIVOS	NEGATIVOS	MÉTRICAS				MATRIZ DE CONFUSÃO			
				ACURÁCIA	F1 SCORE	PRECISION	RECALL	TN	FP	FN	TP
0	26818	4965	5035	0.8688	0.8688	0.8614	0.8763	4336	699	613	4352
1	27041	5061	4939	0.8769	0.8793	0.8702	0.8887	4270	669	562	4499
2	27081	4969	5031	0.8718	0.8717	0.8659	0.8779	4356	675	607	4362
3	26851	5003	4997	0.8730	0.8754	0.8592	0.8925	4265	732	538	4465
4	26985	4954	5046	0.8767	0.8772	0.8660	0.8890	4364	682	551	4403
5	26783	5017	4983	0.8727	0.8753	0.8594	0.8921	4252	731	542	4475
6	26652	5029	4971	0.8712	0.8737	0.8627	0.8855	4261	710	578	4451
7	27173	4913	5087	0.8739	0.8730	0.8640	0.8824	4405	682	579	4334
8	26966	5008	4992	0.8766	0.8784	0.8677	0.8896	4312	680	554	4454
9	27049	4966	5034	0.8643	0.8653	0.8535	0.8777	4285	749	608	4358
	26939.9	4988.5	5011.5	0.8726	0.8738	0.8630	0.8852	4310.6	700.9	573.2	4415.3

Fonte: a autora.

A.2.2 Classificador não linear, com função *kernel* radial

PIPELINE 1 (STEMMING + BAG-OF-WORDS), DESCONSIDERANDO FREQUÊNCIA <= 5, USANDO KERNEL RADIAL											
SEMENTE	ATRIBUTOS	POSITIVOS	NEGATIVOS	MÉTRICAS				MATRIZ DE CONFUSÃO			
				ACURÁCIA	F1 SCORE	PRECISION	RECALL	TN	FP	FN	TP
0	12084	4965	5035	0.5385	0.1749	0.7866	0.0993	4896	139	4476	489
1	12121	5061	4939	0.5281	0.6811	0.5176	0.9966	237	4702	17	5044
2	12192	4969	5031	0.5366	0.1722	0.7755	0.0973	4884	147	4487	482
3	12047	5003	4997	0.5454	0.2561	0.7085	0.1654	4638	359	4187	816
4	12130	4954	5046	0.5365	0.1718	0.7528	0.0972	4884	162	4473	481
5	11964	5017	4983	0.5452	0.4029	0.6807	0.4250	3376	1607	2941	2076
6	11967	5029	4971	0.5262	0.5579	0.5662	0.7512	1527	3444	1294	3735
7	12166	4913	5087	0.5333	0.1171	0.8375	0.0631	5024	63	4604	309
8	12007	5008	4992	0.5304	0.4142	0.6473	0.4876	2925	2067	2629	2379
9	12153	4966	5034	0.5355	0.1460	0.8544	0.0815	4954	80	4565	401
	12083.1	4988.5	5011.5	0.5356	0.3094	0.7127	0.3264	3734.5	1277	3367.3	1621.2

PIPELINE 1 (STEMMING + BAG-OF-WORDS), DESCONSIDERANDO FREQUÊNCIA <= 4, USANDO KERNEL RADIAL											
SEMENTE	ATRIBUTOS	POSITIVOS	NEGATIVOS	MÉTRICAS				MATRIZ DE CONFUSÃO			
				ACURÁCIA	F1 SCORE	PRECISION	RECALL	TN	FP	FN	TP
0	13460	4965	5035	0.5387	0.1732	0.7972	0.0980	4904	131	4482	483
1	13467	5061	4939	0.5280	0.6810	0.5176	0.9966	236	4703	17	5044
2	13516	4969	5031	0.5363	0.1719	0.7720	0.0972	4882	149	4488	481
3	13370	5003	4997	0.5409	0.2723	0.6939	0.2212	4329	668	3923	1080
4	13464	4954	5046	0.5363	0.1709	0.7526	0.0966	4885	161	4476	478
5	13322	5017	4983	0.5433	0.4011	0.6809	0.4244	3360	1623	2944	2073
6	13295	5029	4971	0.5264	0.5592	0.5684	0.7524	1523	3448	1288	3741
7	13493	4913	5087	0.5323	0.1136	0.8327	0.0611	5024	63	4614	299
8	13329	5008	4992	0.5302	0.4136	0.6482	0.4874	2924	2068	2630	2378
9	13486	4966	5034	0.5349	0.1429	0.8583	0.0797	4957	77	4574	392
	13420.2	4988.5	5011.5	0.5347	0.3100	0.7122	0.3315	3702.4	1309.1	3343.6	1644.9

Fonte: a autora.

A.3 Árvore de Decisão

PIPELINE 1: STEMMING + BAG-OF-WORDS											
SEMENTE	ATRIBUTOS	POSITIVOS	NEGATIVOS	MÉTRICAS				MATRIZ DE CONFUSÃO			
				ACURÁCIA	F1 SCORE	PRECISION	RECALL	TN	FP	FN	TP
0	36078	4965	5035	0.7091	0.7057	0.7081	0.7036	3597	1438	1471	3494
1	36380	5061	4939	0.7078	0.7129	0.7084	0.7178	3445	1494	1428	3633
2	36369	4969	5031	0.7047	0.7021	0.7036	0.7010	3564	1467	1486	3483
3	36263	5003	4997	0.7112	0.7127	0.7092	0.7166	3528	1469	1419	3584
4	36420	4954	5046	0.7152	0.7121	0.7128	0.7115	3627	1419	1429	3525
5	35886	5017	4983	0.7191	0.7197	0.7200	0.7200	3579	1404	1405	3612
6	35976	5029	4971	0.7110	0.7127	0.7122	0.7136	3522	1449	1441	3588
7	36313	4913	5087	0.7094	0.7054	0.7029	0.7084	3615	1472	1434	3479
8	35958	5008	4992	0.7155	0.7173	0.7140	0.7211	3545	1447	1398	3610
9	36545	4966	5034	0.7071	0.7065	0.7034	0.7107	3544	1490	1439	3527
	36218.8	4988.5	5011.5	0.7110	0.7107	0.7095	0.7124	3556.6	1454.9	1435	3553.5

PIPELINE 2: STEMMING + TF-IDF											
SEMENTE	ATRIBUTOS	POSITIVOS	NEGATIVOS	MÉTRICAS				MATRIZ DE CONFUSÃO			
				ACURÁCIA	F1 SCORE	PRECISION	RECALL	TN	FP	FN	TP
0	36078	4965	5035	0.7030	0.7016	0.6993	0.7041	3533	1502	1468	3497
1	36380	5061	4939	0.6982	0.7012	0.7018	0.7010	3433	1506	1512	3549
2	36369	4969	5031	0.6960	0.6928	0.6957	0.6907	3530	1501	1539	3430
3	36263	5003	4997	0.7060	0.7036	0.7095	0.6983	3567	1430	1510	3493
4	36420	4954	5046	0.7119	0.7104	0.7074	0.7136	3584	1462	1419	3535
5	35886	5017	4983	0.7162	0.7156	0.7189	0.7129	3585	1398	1440	3577
6	35976	5029	4971	0.6985	0.6999	0.7007	0.6995	3469	1502	1513	3516
7	36313	4913	5087	0.7016	0.6945	0.6985	0.6911	3622	1465	1519	3394
8	35958	5008	4992	0.7104	0.7128	0.7082	0.7183	3509	1483	1413	3595
9	36545	4966	5034	0.7083	0.7051	0.7082	0.7030	3594	1440	1477	3489
	36218.8	4988.5	5011.5	0.7050	0.7038	0.7048	0.7033	3542.6	1468.9	1481	3507.5

PIPELINE 3: LEMATIZAÇÃO + BAG-OF-WORDS											
SEMENTE	ATRIBUTOS	POSITIVOS	NEGATIVOS	MÉTRICAS				MATRIZ DE CONFUSÃO			
				ACURÁCIA	F1 SCORE	PRECISION	RECALL	TN	FP	FN	TP
0	45139	4965	5035	0.7081	0.7056	0.7066	0.7053	3579	1456	1463	3502
1	45514	5061	4939	0.7002	0.7045	0.7020	0.7075	3421	1518	1480	3581
2	45641	4969	5031	0.7030	0.7024	0.6993	0.7060	3523	1508	1462	3507
3	45216	5003	4997	0.7114	0.7091	0.7143	0.7042	3589	1408	1478	3525
4	45585	4954	5046	0.7111	0.7088	0.7073	0.7107	3590	1456	1433	3521
5	44936	5017	4983	0.7159	0.7170	0.7162	0.7184	3556	1427	1414	3603
6	45153	5029	4971	0.7092	0.7122	0.7086	0.7162	3490	1481	1427	3602
7	45489	4913	5087	0.7098	0.7040	0.7055	0.7027	3646	1441	1461	3452
8	45048	5008	4992	0.7108	0.7094	0.7140	0.7056	3577	1415	1477	3531
9	45679	4966	5034	0.7019	0.7009	0.6984	0.7039	3524	1510	1471	3495
	45340	4988.5	5011.5	0.7081	0.7074	0.7072	0.7080	3549.5	1462	1456.6	3531.9

PIPELINE 4: LEMATIZAÇÃO + TF-IDF											
SEMENTE	ATRIBUTOS	POSITIVOS	NEGATIVOS	MÉTRICAS				MATRIZ DE CONFUSÃO			
				ACURÁCIA	F1 SCORE	PRECISION	RECALL	TN	FP	FN	TP
0	45139	4965	5035	0.7039	0.7030	0.6995	0.7071	3527	1508	1453	3512
1	45514	5061	4939	0.7029	0.7078	0.7038	0.7121	3424	1515	1456	3605
2	45641	4969	5031	0.7010	0.6977	0.7010	0.6950	3558	1473	1517	3452
3	45216	5003	4997	0.7058	0.7040	0.7089	0.6999	3557	1440	1502	3501
4	45585	4954	5046	0.7107	0.7085	0.7066	0.7106	3585	1461	1432	3522
5	44936	5017	4983	0.7132	0.7133	0.7156	0.7116	3563	1420	1448	3569
6	45153	5029	4971	0.7040	0.7054	0.7059	0.7054	3493	1478	1482	3547
7	45489	4913	5087	0.6994	0.6928	0.6959	0.6900	3605	1482	1524	3389
8	45048	5008	4992	0.7132	0.7130	0.7143	0.7122	3566	1426	1442	3566
9	45679	4966	5034	0.6979	0.6960	0.6953	0.6973	3518	1516	1505	3461
	45340	4988.5	5011.5	0.7052	0.7041	0.7047	0.7041	3539.6	1471.9	1476.1	3512.4

Fonte: a autora.

A.4 Random Forest

PIPELINE 1: STEMMING + BAG-OF-WORDS											
SEMENTE	ATRIBUTOS	POSITIVOS	NEGATIVOS	MÉTRICAS				MATRIZ DE CONFUSÃO			
				ACURÁCIA	F1 SCORE	PRECISION	RECALL	TN	FP	FN	TP
0	36078	4965	5035	0.7498	0.7272	0.7925	0.6722	4161	874	1628	3337
1	36380	5061	4939	0.7536	0.7392	0.7949	0.6913	4037	902	1562	3499
2	36369	4969	5031	0.7430	0.7195	0.7861	0.6641	4131	900	1670	3299
3	36263	5003	4997	0.7416	0.7206	0.7844	0.6668	4080	917	1667	3336
4	36420	4954	5046	0.7473	0.7228	0.7917	0.6655	4178	868	1659	3295
5	35886	5017	4983	0.7523	0.7357	0.7915	0.6878	4075	908	1569	3448
6	35976	5029	4971	0.7546	0.7380	0.7974	0.6875	4091	880	1574	3455
7	36313	4913	5087	0.7498	0.7235	0.7916	0.6671	4222	865	1637	3276
8	35958	5008	4992	0.7510	0.7315	0.7952	0.6775	4118	874	1616	3392
9	36545	4966	5034	0.7376	0.7131	0.7804	0.6568	4115	919	1705	3261
	36218.8	4988.5	5011.5	0.7481	0.7271	0.7905	0.6737	4120.8	890.7	1628.7	3359.8

PIPELINE 2: STEMMING + TF-IDF											
SEMENTE	ATRIBUTOS	POSITIVOS	NEGATIVOS	MÉTRICAS				MATRIZ DE CONFUSÃO			
				ACURÁCIA	F1 SCORE	PRECISION	RECALL	TN	FP	FN	TP
0	36078	4965	5035	0.7420	0.7170	0.7868	0.6594	4148	887	1693	3272
1	36380	5061	4939	0.7439	0.7276	0.7868	0.6770	4012	927	1634	3427
2	36369	4969	5031	0.7418	0.7155	0.7902	0.6542	4168	863	1719	3250
3	36263	5003	4997	0.7343	0.7121	0.7777	0.6572	4058	939	1718	3285
4	36420	4954	5046	0.7472	0.7255	0.7847	0.6755	4128	918	1610	3344
5	35886	5017	4983	0.7402	0.7207	0.7823	0.6687	4050	933	1665	3352
6	35976	5029	4971	0.7492	0.7294	0.7979	0.6725	4112	859	1649	3380
7	36313	4913	5087	0.7520	0.7261	0.7932	0.6697	4230	857	1623	3290
8	35958	5008	4992	0.7377	0.7135	0.7879	0.6522	4113	879	1744	3264
9	36545	4966	5034	0.7423	0.7177	0.7862	0.6607	4141	893	1684	3282
	36218.8	4988.5	5011.5	0.7431	0.7205	0.7874	0.6647	4116	895.5	1673.9	3314.6

PIPELINE 3: LEMATIZAÇÃO + BAG-OF-WORDS											
SEMENTE	ATRIBUTOS	POSITIVOS	NEGATIVOS	MÉTRICAS				MATRIZ DE CONFUSÃO			
				ACURÁCIA	F1 SCORE	PRECISION	RECALL	TN	FP	FN	TP
0	45139	4965	5035	0.7400	0.7146	0.7849	0.6564	4142	893	1707	3258
1	45514	5061	4939	0.7509	0.7349	0.7949	0.6836	4049	890	1601	3460
2	45641	4969	5031	0.7483	0.7245	0.7938	0.6667	4171	860	1657	3312
3	45216	5003	4997	0.7446	0.7232	0.7901	0.6673	4110	887	1667	3336
4	45585	4954	5046	0.7497	0.7277	0.7896	0.6752	4154	892	1611	3343
5	44936	5017	4983	0.7483	0.7277	0.7952	0.6718	4116	867	1650	3367
6	45153	5029	4971	0.7580	0.7432	0.7965	0.6968	4076	895	1525	3504
7	45489	4913	5087	0.7479	0.7188	0.7944	0.6568	4252	835	1686	3227
8	45048	5008	4992	0.7565	0.7380	0.8003	0.6851	4136	856	1579	3429
9	45679	4966	5034	0.7429	0.7181	0.7874	0.6603	4149	885	1686	3280
	45340	4988.5	5011.5	0.7487	0.7271	0.7927	0.6720	4135.5	876	1636.9	3351.6

PIPELINE 4: LEMATIZAÇÃO + TF-IDF											
SEMENTE	ATRIBUTOS	POSITIVOS	NEGATIVOS	MÉTRICAS				MATRIZ DE CONFUSÃO			
				ACURÁCIA	F1 SCORE	PRECISION	RECALL	TN	FP	FN	TP
0	45139	4965	5035	0.7425	0.7179	0.7869	0.6604	4147	888	1687	3278
1	45514	5061	4939	0.7432	0.7264	0.7869	0.6752	4013	926	1642	3419
2	45641	4969	5031	0.7441	0.7179	0.7943	0.6559	4185	846	1713	3256
3	45216	5003	4997	0.7449	0.7253	0.7861	0.6736	4080	917	1634	3369
4	45585	4954	5046	0.7466	0.7199	0.7956	0.6576	4209	837	1697	3257
5	44936	5017	4983	0.7417	0.7230	0.7828	0.6725	4046	937	1646	3371
6	45153	5029	4971	0.7433	0.7247	0.7873	0.6716	4057	914	1653	3376
7	45489	4913	5087	0.7413	0.7122	0.7856	0.6520	4211	876	1711	3202
8	45048	5008	4992	0.7472	0.7277	0.7902	0.6748	4095	897	1631	3377
9	45679	4966	5034	0.7413	0.7179	0.7825	0.6634	4119	915	1672	3294
	45340	4988.5	5011.5	0.7436	0.7213	0.7878	0.6657	4116.2	895.3	1668.6	3319.9

Fonte: a autora.

A.5 *k*-Nearest Neighbours

PIPELINE 1: STEMMING + BAG-OF-WORDS											
SEMENTE	ATRIBUTOS	POSITIVOS	NEGATIVOS	MÉTRICAS				MATRIZ DE CONFUSÃO			
				ACURÁCIA	F1 SCORE	PRECISION	RECALL	TN	FP	FN	TP
0	36078	4965	5035	0.6363	0.6282	0.6384	0.6207	3284	1751	1886	3079
1	36380	5061	4939	0.6123	0.6253	0.6120	0.6437	2867	2072	1805	3256
2	36369	4969	5031	0.6126	0.5811	0.6270	0.5423	3429	1602	2272	2697
3	36263	5003	4997	0.6242	0.6571	0.6045	0.7208	2634	2363	1395	3608
4	36420	4954	5046	0.6275	0.6236	0.6241	0.6249	3177	1869	1856	3098
5	35886	5017	4983	0.6324	0.6637	0.6143	0.7247	2693	2290	1386	3631
6	35976	5029	4971	0.6339	0.6311	0.6409	0.6238	3205	1766	1895	3134
7	36313	4913	5087	0.6256	0.6049	0.6274	0.5853	3379	1708	2036	2877
8	35958	5008	4992	0.6388	0.6709	0.6177	0.7366	2698	2294	1318	3690
9	36545	4966	5034	0.6004	0.5809	0.6052	0.5614	3216	1818	2178	2788
	36218.8	4988.5	5011.5	0.6244	0.6267	0.6211	0.6384	3058.2	1953.3	1802.7	3185.8

PIPELINE 2: STEMMING + TF-IDF											
SEMENTE	ATRIBUTOS	POSITIVOS	NEGATIVOS	MÉTRICAS				MATRIZ DE CONFUSÃO			
				ACURÁCIA	F1 SCORE	PRECISION	RECALL	TN	FP	FN	TP
0	36078	4965	5035	0.7300	0.7403	0.7081	0.7764	3446	1589	1111	3854
1	36380	5061	4939	0.7379	0.7525	0.7204	0.7878	3392	1547	1074	3987
2	36369	4969	5031	0.7427	0.7499	0.7252	0.7766	3568	1463	1110	3859
3	36263	5003	4997	0.7286	0.7391	0.7114	0.7692	3437	1560	1154	3849
4	36420	4954	5046	0.7404	0.7461	0.7233	0.7705	3587	1459	1137	3817
5	35886	5017	4983	0.7435	0.7544	0.7256	0.7861	3492	1491	1074	3943
6	35976	5029	4971	0.7385	0.7511	0.7202	0.7851	3437	1534	1081	3948
7	36313	4913	5087	0.7294	0.7359	0.7066	0.7680	3521	1566	1140	3773
8	35958	5008	4992	0.7430	0.7544	0.7234	0.7887	3482	1510	1060	3948
9	36545	4966	5034	0.7294	0.7370	0.7122	0.7641	3500	1534	1172	3794
	36218.8	4988.5	5011.5	0.7363	0.7461	0.7176	0.7773	3486.2	1525.3	1111.3	3877.2

PIPELINE 3: LEMATIZAÇÃO + BAG-OF-WORDS											
SEMENTE	ATRIBUTOS	POSITIVOS	NEGATIVOS	MÉTRICAS				MATRIZ DE CONFUSÃO			
				ACURÁCIA	F1 SCORE	PRECISION	RECALL	TN	FP	FN	TP
0	45139	4965	5035	0.6367	0.6256	0.6406	0.6141	3320	1715	1918	3047
1	45514	5061	4939	0.6167	0.6217	0.6212	0.6263	2999	1940	1893	3168
2	45641	4969	5031	0.6208	0.5740	0.6481	0.5158	3642	1389	2403	2566
3	45216	5003	4997	0.6304	0.6551	0.6141	0.7033	2782	2215	1481	3522
4	45585	4954	5046	0.6227	0.6031	0.6293	0.5811	3346	1700	2073	2881
5	44936	5017	4983	0.6385	0.6734	0.6161	0.7445	2653	2330	1285	3732
6	45153	5029	4971	0.6343	0.6206	0.6499	0.5958	3350	1621	2036	2993
7	45489	4913	5087	0.6358	0.6158	0.6384	0.5962	3428	1659	1983	2930
8	45048	5008	4992	0.6408	0.6629	0.6259	0.7072	2867	2125	1467	3541
9	45679	4966	5034	0.6043	0.5621	0.6234	0.5149	3486	1548	2409	2557
	45340	4988.5	5011.5	0.6281	0.6214	0.6307	0.6199	3187.3	1824.2	1894.8	3093.7

PIPELINE 4: LEMATIZAÇÃO + TF-IDF											
SEMENTE	ATRIBUTOS	POSITIVOS	NEGATIVOS	MÉTRICAS				MATRIZ DE CONFUSÃO			
				ACURÁCIA	F1 SCORE	PRECISION	RECALL	TN	FP	FN	TP
0	45139	4965	5035	0.7325	0.7416	0.7122	0.7745	3480	1555	1120	3845
1	45514	5061	4939	0.7373	0.7485	0.7253	0.7735	3458	1481	1146	3915
2	45641	4969	5031	0.7399	0.7455	0.7252	0.7676	3584	1447	1154	3815
3	45216	5003	4997	0.7349	0.7415	0.7232	0.7610	3541	1456	1195	3808
4	45585	4954	5046	0.7440	0.7480	0.7293	0.7679	3635	1411	1149	3805
5	44936	5017	4983	0.7316	0.7404	0.7190	0.7639	3485	1498	1186	3831
6	45153	5029	4971	0.7404	0.7496	0.7276	0.7731	3516	1455	1141	3888
7	45489	4913	5087	0.7227	0.7281	0.7022	0.7561	3513	1574	1199	3714
8	45048	5008	4992	0.7344	0.7441	0.7188	0.7718	3481	1511	1145	3863
9	45679	4966	5034	0.7313	0.7380	0.7153	0.7626	3527	1507	1180	3786
	45340	4988.5	5011.5	0.7349	0.7425	0.7198	0.7672	3522	1489.5	1161.5	3827

Fonte: a autora.

A.6 Multilayer Perceptron

PIPELINE 1: STEMMING + BAG-OF-WORDS											
SEMENTE	ATRIBUTOS	POSITIVOS	NEGATIVOS	MÉTRICAS				MATRIZ DE CONFUSÃO			
				ACURÁCIA	F1 SCORE	PRECISION	RECALL	TN	FP	FN	TP
0	36078	4965	5035	0.8593	0.8596	0.8510	0.8685	4280	755	652	4313
1	36380	5061	4939	0.8627	0.8653	0.8584	0.8725	4212	727	646	4415
2	36369	4969	5031	0.8608	0.8606	0.8556	0.8659	4305	726	666	4303
3	36263	5003	4997	0.8586	0.8599	0.8521	0.8680	4244	753	661	4342
4	36420	4954	5046	0.8611	0.8611	0.8528	0.8698	4302	744	645	4309
5	35886	5017	4983	0.8652	0.8670	0.8571	0.8775	4250	733	615	4402
6	35976	5029	4971	0.8560	0.8582	0.8510	0.8658	4207	764	676	4353
7	36313	4913	5087	0.8581	0.8565	0.8510	0.8622	4345	742	677	4236
8	35958	5008	4992	0.8629	0.8639	0.8594	0.8685	4280	712	659	4349
9	36545	4966	5034	0.8552	0.8556	0.8475	0.8642	4262	772	676	4290
	36218.8	4988.5	5011.5	0.8600	0.8608	0.8536	0.8683	4268.7	742.8	657.3	4331.2

PIPELINE 2: STEMMING + TF-IDF											
SEMENTE	ATRIBUTOS	POSITIVOS	NEGATIVOS	MÉTRICAS				MATRIZ DE CONFUSÃO			
				ACURÁCIA	F1 SCORE	PRECISION	RECALL	TN	FP	FN	TP
0	36078	4965	5035	0.8581	0.8581	0.8516	0.8651	4286	749	670	4295
1	36380	5061	4939	0.8627	0.8644	0.8626	0.8663	4242	697	676	4385
2	36369	4969	5031	0.8622	0.8619	0.8581	0.8660	4319	712	666	4303
3	36263	5003	4997	0.8602	0.8615	0.8534	0.8700	4250	747	651	4352
4	36420	4954	5046	0.8620	0.8615	0.8563	0.8671	4324	722	658	4296
5	35886	5017	4983	0.8660	0.8669	0.8625	0.8716	4286	697	643	4374
6	35976	5029	4971	0.8559	0.8585	0.8497	0.8677	4197	774	667	4362
7	36313	4913	5087	0.8637	0.8626	0.8548	0.8708	4360	727	636	4277
8	35958	5008	4992	0.8640	0.8646	0.8620	0.8674	4296	696	664	4344
9	36545	4966	5034	0.8549	0.8547	0.8499	0.8598	4280	754	697	4269
	36218.8	4988.5	5011.5	0.8610	0.8615	0.8561	0.8672	4284	727.5	662.8	4325.7

PIPELINE 3: LEMATIZAÇÃO + BAG-OF-WORDS											
SEMENTE	ATRIBUTOS	POSITIVOS	NEGATIVOS	MÉTRICAS				MATRIZ DE CONFUSÃO			
				ACURÁCIA	F1 SCORE	PRECISION	RECALL	TN	FP	FN	TP
0	45139	4965	5035	0.8653	0.8652	0.8576	0.8731	4316	719	628	4337
1	45514	5061	4939	0.8666	0.8680	0.8672	0.8692	4267	672	662	4399
2	45641	4969	5031	0.8611	0.8609	0.8560	0.8663	4307	724	665	4304
3	45216	5003	4997	0.8637	0.8651	0.8561	0.8745	4262	735	628	4375
4	45585	4954	5046	0.8652	0.8652	0.8576	0.8731	4327	719	629	4325
5	44936	5017	4983	0.8652	0.8664	0.8600	0.8732	4270	713	635	4382
6	45153	5029	4971	0.8566	0.8588	0.8516	0.8670	4208	763	671	4358
7	45489	4913	5087	0.8652	0.8639	0.8570	0.8712	4372	715	633	4280
8	45048	5008	4992	0.8639	0.8646	0.8626	0.8667	4299	693	668	4340
9	45679	4966	5034	0.8558	0.8559	0.8493	0.8630	4273	761	681	4285
	45340	4988.5	5011.5	0.8629	0.8634	0.8575	0.8697	4290.1	721.4	650	4338.5

PIPELINE 4: LEMATIZAÇÃO + TF-IDF											
SEMENTE	ATRIBUTOS	POSITIVOS	NEGATIVOS	MÉTRICAS				MATRIZ DE CONFUSÃO			
				ACURÁCIA	F1 SCORE	PRECISION	RECALL	TN	FP	FN	TP
0	45139	4965	5035	0.8637	0.8631	0.8599	0.8666	4334	701	662	4303
1	45514	5061	4939	0.8665	0.8685	0.8646	0.8726	4248	691	644	4417
2	45641	4969	5031	0.8666	0.8662	0.8625	0.8703	4342	689	645	4324
3	45216	5003	4997	0.8628	0.8635	0.8593	0.8680	4286	711	661	4342
4	45585	4954	5046	0.8595	0.8599	0.8509	0.8696	4288	758	647	4307
5	44936	5017	4983	0.8657	0.8676	0.8564	0.8794	4244	739	604	4413
6	45153	5029	4971	0.8564	0.8582	0.8527	0.8644	4218	753	683	4346
7	45489	4913	5087	0.8681	0.8667	0.8608	0.8729	4393	694	625	4288
8	45048	5008	4992	0.8638	0.8640	0.8643	0.8645	4308	684	678	4330
9	45679	4966	5034	0.8588	0.8586	0.8540	0.8634	4301	733	679	4287
	45340	4988.5	5011.5	0.8632	0.8636	0.8585	0.8692	4296.2	715.3	652.8	4335.7

Fonte: a autora.

APÊNDICE B – Experimentos completos com *Rotten Tomatoes*

B.1 *Naïve Bayes*

PIPELINE 1: STEMMING + BAG-OF-WORDS											
SEMENTE	ATRIBUTOS	POSITIVOS	NEGATIVOS	MÉTRICAS				MATRIZ DE CONFUSÃO			
				ACURÁCIA	F1 SCORE	PRECISION	RECALL	TN	FP	FN	TP
0	12090	6118	3882	0.7493	0.8040	0.7706	0.8407	2350	1532	975	5143
1	12118	6084	3916	0.7526	0.8047	0.7734	0.8388	2421	1495	979	5105
2	12022	6071	3929	0.7537	0.8053	0.7739	0.8396	2440	1489	974	5097
3	12078	6124	3876	0.7543	0.8076	0.7755	0.8427	2382	1494	963	5161
4	12030	6066	3934	0.7531	0.8053	0.7716	0.8423	2422	1512	957	5109
5	12136	6134	3866	0.7444	0.7995	0.7702	0.8314	2344	1522	1034	5100
6	12137	6083	3917	0.7482	0.8009	0.7715	0.8328	2416	1501	1017	5066
7	11880	6148	3852	0.748	0.8032	0.7720	0.8373	2333	1519	1001	5147
8	12043	6136	3864	0.7448	0.8006	0.7688	0.8352	2323	1541	1011	5125
9	12068	5985	4015	0.7407	0.7907	0.7648	0.8186	2508	1507	1086	4899
	12060.2	6094.9	3905.1	0.7489	0.8022	0.7712	0.8359	2393.9	1511.2	999.7	5095.2

PIPELINE 2: STEMMING + TF-IDF											
SEMENTE	ATRIBUTOS	POSITIVOS	NEGATIVOS	MÉTRICAS				MATRIZ DE CONFUSÃO			
				ACURÁCIA	F1 SCORE	PRECISION	RECALL	TN	FP	FN	TP
0	12090	6118	3882	0.7063	0.8004	0.6849	0.9634	1170	2712	225	5893
1	12118	6084	3916	0.7095	0.8006	0.6871	0.9591	1259	2657	248	5836
2	12022	6071	3929	0.7064	0.7984	0.6845	0.9582	1247	2682	254	5817
3	12078	6124	3876	0.7095	0.8024	0.6875	0.9637	1193	2683	222	5902
4	12030	6066	3934	0.7077	0.7989	0.6854	0.9577	1268	2666	257	5809
5	12136	6134	3866	0.7013	0.7980	0.6818	0.9622	1111	2755	232	5902
6	12137	6083	3917	0.7058	0.7978	0.6854	0.9548	1250	2667	275	5808
7	11880	6148	3852	0.7064	0.8010	0.6864	0.9622	1149	2703	233	5915
8	12043	6136	3864	0.703	0.7990	0.6831	0.9627	1123	2741	229	5907
9	12068	5985	4015	0.7084	0.7965	0.6838	0.9541	1374	2641	275	5710
	12060.2	6094.9	3905.1	0.7064	0.7993	0.6850	0.9598	1214.4	2690.7	245	5849.9

PIPELINE 3: LEMATIZAÇÃO + BAG-OF-WORDS											
SEMENTE	ATRIBUTOS	POSITIVOS	NEGATIVOS	MÉTRICAS				MATRIZ DE CONFUSÃO			
				ACURÁCIA	F1 SCORE	PRECISION	RECALL	TN	FP	FN	TP
0	14706	6118	3882	0.7531	0.8065	0.7746	0.8415	2383	1499	970	5148
1	14718	6084	3916	0.7485	0.8018	0.7697	0.8367	2393	1523	992	5092
2	14620	6071	3929	0.7549	0.8064	0.7744	0.8413	2441	1488	963	5108
3	14737	6124	3876	0.7499	0.8046	0.7711	0.8413	2347	1529	972	5152
4	14641	6066	3934	0.7541	0.8055	0.7739	0.8401	2445	1489	970	5096
5	14694	6134	3866	0.7439	0.7993	0.7693	0.8319	2336	1530	1031	5103
6	14788	6083	3917	0.746	0.7992	0.7695	0.8315	2402	1515	1025	5058
7	14503	6148	3852	0.7455	0.8012	0.7707	0.8347	2325	1527	1018	5130
8	14638	6136	3864	0.7427	0.7991	0.7670	0.8341	2309	1555	1018	5118
9	14606	5985	4015	0.7394	0.7895	0.7640	0.8171	2504	1511	1095	4890
	14665.1	6094.9	3905.1	0.7478	0.8013	0.7704	0.8350	2388.5	1516.6	1005.4	5089.5

PIPELINE 4: LEMATIZAÇÃO + TF-IDF											
SEMENTE	ATRIBUTOS	POSITIVOS	NEGATIVOS	MÉTRICAS				MATRIZ DE CONFUSÃO			
				ACURÁCIA	F1 SCORE	PRECISION	RECALL	TN	FP	FN	TP
0	14706	6118	3882	0.7023	0.7991	0.6803	0.9688	1097	2785	192	5926
1	14718	6084	3916	0.7001	0.7959	0.6790	0.9618	1149	2767	232	5852
2	14620	6071	3929	0.7005	0.7957	0.6790	0.9610	1171	2758	237	5834
3	14737	6124	3876	0.7033	0.7998	0.6815	0.9680	1105	2771	196	5928
4	14641	6066	3934	0.7079	0.8002	0.6837	0.9650	1226	2708	213	5853
5	14694	6134	3866	0.701	0.7986	0.6803	0.9670	1078	2788	202	5932
6	14788	6083	3917	0.7011	0.7959	0.6806	0.9589	1178	2739	250	5833
7	14503	6148	3852	0.6975	0.7966	0.6789	0.9642	1048	2804	221	5927
8	14638	6136	3864	0.6987	0.7973	0.6787	0.9665	1056	2808	205	5931
9	14606	5985	4015	0.6997	0.7918	0.6765	0.9547	1283	2732	271	5714
	14665.1	6094.9	3905.1	0.7012	0.7971	0.6799	0.9636	1139.1	2766	221.9	5873

Fonte: a autora.

B.2 Support Vector Machine

PIPELINE 1: STEMMING + BAG-OF-WORDS											
SEMENTE	ATRIBUTOS	POSITIVOS	NEGATIVOS	MÉTRICAS				MATRIZ DE CONFUSÃO			
				ACURÁCIA	F1 SCORE	PRECISION	RECALL	TN	FP	FN	TP
0	7292	6118	3882	0.7234	0.7980	0.7208	0.8940	1765	2117	649	5469
1	7275	6084	3916	0.7312	0.8037	0.7229	0.9051	1805	2111	577	5607
2	7291	6071	3929	0.7263	0.7977	0.7235	0.8890	1866	2063	674	5397
3	7323	6124	3876	0.7329	0.8048	0.7284	0.8992	1822	2054	617	5507
4	7249	6066	3934	0.7236	0.7971	0.7184	0.8957	1803	2131	633	5433
5	7331	6134	3866	0.7212	0.7982	0.7176	0.8995	1694	2172	616	5518
6	7345	6083	3917	0.7233	0.7974	0.7188	0.8959	1784	2133	634	5449
7	7213	6148	3852	0.7164	0.7961	0.7131	0.9013	1623	2229	607	5541
8	7196	6136	3864	0.7181	0.7966	0.7144	0.9005	1655	2209	610	5526
9	7300	5985	4015	0.7211	0.7901	0.7186	0.8777	1958	2057	732	5253
	7281.5	6094.9	3905.1	0.7238	0.7980	0.7196	0.8958	1777.5	2127.6	634.9	5460

PIPELINE 2: STEMMING + TF-IDF											
SEMENTE	ATRIBUTOS	POSITIVOS	NEGATIVOS	MÉTRICAS				MATRIZ DE CONFUSÃO			
				ACURÁCIA	F1 SCORE	PRECISION	RECALL	TN	FP	FN	TP
0	7292	6118	3882	0.6595	0.7794	0.6455	0.9839	576	3306	99	6019
1	7275	6084	3916	0.6593	0.7778	0.6447	0.9804	628	3288	119	5965
2	7291	6071	3929	0.6545	0.7749	0.6411	0.9796	598	3331	124	5947
3	7323	6124	3876	0.6603	0.7806	0.6457	0.9868	560	3316	81	6043
4	7249	6066	3934	0.6619	0.7785	0.6458	0.9801	674	3260	121	5945
5	7331	6134	3866	0.6624	0.7815	0.6480	0.9845	585	3281	95	6039
6	7345	6083	3917	0.6588	0.7781	0.6436	0.9843	601	3316	96	5987
7	7213	6148	3852	0.6594	0.7804	0.6463	0.9854	536	3316	90	6058
8	7196	6136	3864	0.6576	0.7789	0.6449	0.9838	539	3325	99	6037
9	7300	5985	4015	0.6563	0.7732	0.6389	0.9793	702	3313	124	5861
	7281.5	6094.9	3905.1	0.6590	0.7783	0.6444	0.9828	599.9	3305.2	104.8	5990.1

PIPELINE 3: LEMATIZAÇÃO + BAG-OF-WORDS											
SEMENTE	ATRIBUTOS	POSITIVOS	NEGATIVOS	MÉTRICAS				MATRIZ DE CONFUSÃO			
				ACURÁCIA	F1 SCORE	PRECISION	RECALL	TN	FP	FN	TP
0	8197	6118	3882	0.7222	0.7989	0.7166	0.9030	1698	2184	594	5524
1	8146	6084	3916	0.7226	0.7982	0.7160	0.9019	1738	2178	596	5488
2	8218	6071	3929	0.7184	0.7926	0.7166	0.8869	1800	2129	687	5384
3	8261	6124	3876	0.723	0.7986	0.7196	0.8971	1736	2140	630	5494
4	8171	6066	3934	0.7206	0.7956	0.7150	0.8970	1765	2169	625	5441
5	8202	6134	3866	0.7218	0.7994	0.7166	0.9045	1670	2196	586	5548
6	8238	6083	3917	0.7139	0.7918	0.7104	0.8947	1697	2220	641	5442
7	8118	6148	3852	0.717	0.7971	0.7124	0.9053	1605	2247	583	5565
8	8113	6136	3864	0.7169	0.7967	0.7118	0.9051	1615	2249	582	5554
9	8201	5985	4015	0.7136	0.7866	0.7096	0.8827	1853	2162	702	5283
	8186.5	6094.9	3905.1	0.7190	0.7956	0.7144	0.8978	1717.7	2187.4	622.6	5472.3

PIPELINE 4: LEMATIZAÇÃO + TF-IDF											
SEMENTE	ATRIBUTOS	POSITIVOS	NEGATIVOS	MÉTRICAS				MATRIZ DE CONFUSÃO			
				ACURÁCIA	F1 SCORE	PRECISION	RECALL	TN	FP	FN	TP
0	8197	6118	3882	0.6496	0.7746	0.6385	0.9851	470	3412	92	6026
1	8146	6084	3916	0.6518	0.7747	0.6388	0.9843	529	3387	95	5989
2	8218	6071	3929	0.6458	0.7710	0.6346	0.9826	493	3436	106	5965
3	8261	6124	3876	0.6497	0.7753	0.6384	0.9871	452	3424	79	6046
4	8171	6066	3934	0.6524	0.7744	0.6385	0.9841	555	3379	97	5969
5	8202	6134	3866	0.6531	0.7768	0.6416	0.9846	491	3375	94	6040
6	8238	6083	3917	0.6507	0.7745	0.6375	0.9869	504	3413	80	6003
7	8118	6148	3852	0.6496	0.7755	0.6396	0.9854	438	3414	90	6058
8	8113	6136	3864	0.65	0.7755	0.6393	0.9858	451	3413	87	6049
9	8201	5985	4015	0.6449	0.7678	0.6307	0.9815	575	3440	111	5874
	8186.5	6094.9	3905.1	0.6498	0.7740	0.6377	0.9847	495.8	3409.3	93.1	6001.8

Fonte: a autora.

B.3 Árvore de Decisão

PIPELINE 1: STEMMING + BAG-OF-WORDS											
SEMENTE	ATRIBUTOS	POSITIVOS	NEGATIVOS	MÉTRICAS				MATRIZ DE CONFUSÃO			
				ACURÁCIA	F1 SCORE	PRECISION	RECALL	TN	FP	FN	TP
0	12090	6118	3882	0.6383	0.7030	0.7058	0.7006	2097	1785	1832	4286
1	12118	6084	3916	0.6453	0.7018	0.7178	0.6868	2274	1642	1905	4179
2	12022	6071	3929	0.6357	0.6949	0.7069	0.6837	2207	1722	1921	4150
3	12078	6124	3876	0.6464	0.7156	0.7049	0.7271	2011	1865	1671	4453
4	12030	6066	3934	0.6423	0.7120	0.6959	0.7299	1997	1937	1640	4426
5	12136	6134	3866	0.6452	0.7161	0.7027	0.7306	1970	1896	1652	4482
6	12137	6083	3917	0.6512	0.7169	0.7082	0.7269	2092	1825	1663	4420
7	11880	6148	3852	0.6416	0.7056	0.7123	0.6993	2116	1736	1848	4300
8	12043	6136	3864	0.6397	0.7038	0.7098	0.6981	2113	1751	1852	4284
9	12068	5985	4015	0.6478	0.7032	0.7093	0.6974	2305	1710	1812	4173
	12060.2	6094.9	3905.1	0.6434	0.7073	0.7074	0.7080	2118.2	1786.9	1779.6	4315.3

PIPELINE 2: STEMMING + TF-IDF											
SEMENTE	ATRIBUTOS	POSITIVOS	NEGATIVOS	MÉTRICAS				MATRIZ DE CONFUSÃO			
				ACURÁCIA	F1 SCORE	PRECISION	RECALL	TN	FP	FN	TP
0	12090	6118	3882	0.6342	0.7005	0.7013	0.7000	2058	1824	1834	4284
1	12118	6084	3916	0.6379	0.7001	0.7052	0.6953	2148	1768	1853	4231
2	12022	6071	3929	0.6356	0.6987	0.7018	0.6958	2132	1797	1847	4224
3	12078	6124	3876	0.6474	0.7194	0.7018	0.7383	1953	1923	1603	4521
4	12030	6066	3934	0.6362	0.7067	0.6912	0.7233	1975	1959	1679	4387
5	12136	6134	3866	0.6423	0.7142	0.7001	0.7292	1950	1916	1661	4473
6	12137	6083	3917	0.6375	0.7097	0.6919	0.7287	1943	1974	1651	4432
7	11880	6148	3852	0.6428	0.7069	0.7124	0.7018	2112	1740	1832	4316
8	12043	6136	3864	0.6351	0.7017	0.7039	0.6999	2057	1807	1842	4294
9	12068	5985	4015	0.6263	0.6861	0.6896	0.6829	2176	1839	1898	4087
	12060.2	6094.9	3905.1	0.6375	0.7044	0.6999	0.7095	2050.4	1854.7	1770	4324.9

PIPELINE 3: LEMATIZAÇÃO + BAG-OF-WORDS											
SEMENTE	ATRIBUTOS	POSITIVOS	NEGATIVOS	MÉTRICAS				MATRIZ DE CONFUSÃO			
				ACURÁCIA	F1 SCORE	PRECISION	RECALL	TN	FP	FN	TP
0	14706	6118	3882	0.6353	0.7020	0.7019	0.7027	2055	1827	1820	4298
1	14718	6084	3916	0.6307	0.6887	0.7058	0.6729	2211	1705	1988	4096
2	14620	6071	3929	0.633	0.6932	0.7038	0.6832	2183	1746	1924	4147
3	14737	6124	3876	0.6459	0.7160	0.7039	0.7294	1992	1884	1657	4467
4	14641	6066	3934	0.6384	0.7064	0.6956	0.7183	2028	1906	1710	4356
5	14694	6134	3866	0.6393	0.7106	0.6995	0.7226	1960	1906	1701	4433
6	14788	6083	3917	0.6413	0.7121	0.6955	0.7297	1974	1943	1644	4439
7	14503	6148	3852	0.6422	0.7070	0.7116	0.7030	2101	1751	1827	4321
8	14638	6136	3864	0.6385	0.7038	0.7075	0.7006	2087	1777	1838	4298
9	14606	5985	4015	0.6286	0.6872	0.6926	0.6823	2202	1813	1901	4084
	14665.1	6094.9	3905.1	0.6373	0.7027	0.7018	0.7045	2079.3	1825.8	1801	4293.9

PIPELINE 4: LEMATIZAÇÃO + TF-IDF											
SEMENTE	ATRIBUTOS	POSITIVOS	NEGATIVOS	MÉTRICAS				MATRIZ DE CONFUSÃO			
				ACURÁCIA	F1 SCORE	PRECISION	RECALL	TN	FP	FN	TP
0	14706	6118	3882	0.6201	0.6863	0.6934	0.6799	2041	1841	1958	4160
1	14718	6084	3916	0.6331	0.6973	0.6997	0.6953	2101	1815	1854	4230
2	14620	6071	3929	0.6281	0.6925	0.6953	0.6900	2092	1837	1882	4189
3	14737	6124	3876	0.6395	0.7135	0.6953	0.7331	1906	1970	1635	4489
4	14641	6066	3934	0.6351	0.7075	0.6885	0.7276	1937	1997	1652	4414
5	14694	6134	3866	0.6318	0.7080	0.6889	0.7285	1849	2017	1665	4469
6	14788	6083	3917	0.6273	0.7032	0.6817	0.7265	1853	2064	1663	4420
7	14503	6148	3852	0.6296	0.6953	0.7035	0.6883	2066	1786	1918	4230
8	14638	6136	3864	0.6295	0.6978	0.6984	0.6976	2015	1849	1856	4280
9	14606	5985	4015	0.621	0.6828	0.6839	0.6822	2128	1887	1903	4082
	14665.1	6094.9	3905.1	0.6295	0.6984	0.6928	0.7049	1998.8	1906.3	1798.6	4296.3

Fonte: a autora.

B.4 Random Forest

PIPELINE 1: STEMMING + BAG-OF-WORDS											
SEMENTE	ATRIBUTOS	POSITIVOS	NEGATIVOS	MÉTRICAS				MATRIZ DE CONFUSÃO			
				ACURÁCIA	F1 SCORE	PRECISION	RECALL	TN	FP	FN	TP
0	12090	6118	3882	0.6825	0.7447	0.7329	0.7575	2191	1691	1484	4634
1	12118	6084	3916	0.684	0.7391	0.7417	0.7370	2354	1562	1598	4486
2	12022	6071	3929	0.6814	0.7346	0.7434	0.7261	2406	1523	1663	4408
3	12078	6124	3876	0.6935	0.7594	0.7307	0.7905	2093	1783	1282	4842
4	12030	6066	3934	0.6822	0.7473	0.7212	0.7756	2117	1817	1361	4705
5	12136	6134	3866	0.6828	0.7513	0.7239	0.7815	2035	1831	1341	4793
6	12137	6083	3917	0.6763	0.7434	0.7181	0.7711	2074	1843	1394	4689
7	11880	6148	3852	0.6871	0.7496	0.7373	0.7626	2183	1669	1460	4688
8	12043	6136	3864	0.6884	0.7536	0.7316	0.7772	2115	1749	1367	4769
9	12068	5985	4015	0.682	0.7328	0.7367	0.7294	2455	1560	1620	4365
	12060.2	6094.9	3905.1	0.6840	0.7456	0.7318	0.7609	2202.3	1702.8	1457	4637.9

PIPELINE 2: STEMMING + TF-IDF											
SEMENTE	ATRIBUTOS	POSITIVOS	NEGATIVOS	MÉTRICAS				MATRIZ DE CONFUSÃO			
				ACURÁCIA	F1 SCORE	PRECISION	RECALL	TN	FP	FN	TP
0	12090	6118	3882	0.6883	0.7536	0.7294	0.7800	2111	1771	1346	4772
1	12118	6084	3916	0.6857	0.7481	0.7293	0.7686	2180	1736	1407	4677
2	12022	6071	3929	0.678	0.7394	0.7272	0.7525	2212	1717	1503	4568
3	12078	6124	3876	0.6816	0.7573	0.7101	0.8115	1847	2029	1155	4969
4	12030	6066	3934	0.6873	0.7585	0.7132	0.8103	1958	1976	1151	4915
5	12136	6134	3866	0.6817	0.7567	0.7123	0.8075	1864	2002	1181	4953
6	12137	6083	3917	0.6803	0.7549	0.7070	0.8101	1875	2042	1155	4928
7	11880	6148	3852	0.6844	0.7559	0.7203	0.7957	1952	1900	1256	4892
8	12043	6136	3864	0.6805	0.7515	0.7183	0.7881	1968	1896	1299	4837
9	12068	5985	4015	0.6757	0.7384	0.7137	0.7654	2176	1839	1404	4581
	12060.2	6094.9	3905.1	0.6824	0.7514	0.7181	0.7890	2014.3	1890.8	1285.7	4809.2

PIPELINE 3: LEMATIZAÇÃO + BAG-OF-WORDS											
SEMENTE	ATRIBUTOS	POSITIVOS	NEGATIVOS	MÉTRICAS				MATRIZ DE CONFUSÃO			
				ACURÁCIA	F1 SCORE	PRECISION	RECALL	TN	FP	FN	TP
0	14706	6118	3882	0.6746	0.7373	0.7279	0.7473	2174	1708	1546	4572
1	14718	6084	3916	0.6807	0.7362	0.7396	0.7334	2343	1573	1620	4464
2	14620	6071	3929	0.671	0.7278	0.7306	0.7255	2305	1624	1666	4405
3	14737	6124	3876	0.6796	0.7503	0.7176	0.7866	1979	1897	1307	4817
4	14641	6066	3934	0.679	0.7468	0.7161	0.7811	2054	1880	1330	4736
5	14694	6134	3866	0.6779	0.7479	0.7188	0.7802	1993	1873	1348	4786
6	14788	6083	3917	0.6759	0.7415	0.7197	0.7650	2105	1812	1429	4654
7	14503	6148	3852	0.6793	0.7435	0.7308	0.7569	2139	1713	1494	4654
8	14638	6136	3864	0.671	0.7410	0.7159	0.7683	1995	1869	1421	4715
9	14606	5985	4015	0.6691	0.7225	0.7251	0.7209	2377	1638	1671	4314
	14665.1	6094.9	3905.1	0.6758	0.7395	0.7242	0.7565	2146.4	1758.7	1483.2	4611.7

PIPELINE 4: LEMATIZAÇÃO + TF-IDF											
SEMENTE	ATRIBUTOS	POSITIVOS	NEGATIVOS	MÉTRICAS				MATRIZ DE CONFUSÃO			
				ACURÁCIA	F1 SCORE	PRECISION	RECALL	TN	FP	FN	TP
0	14706	6118	3882	0.6784	0.7465	0.7207	0.7745	2046	1836	1380	4738
1	14718	6084	3916	0.6745	0.7415	0.7171	0.7683	2069	1847	1408	4676
2	14620	6071	3929	0.6723	0.7383	0.7169	0.7613	2101	1828	1449	4622
3	14737	6124	3876	0.6779	0.7555	0.7056	0.8134	1798	2078	1143	4981
4	14641	6066	3934	0.673	0.7492	0.7003	0.8058	1843	2091	1179	4887
5	14694	6134	3866	0.6779	0.7558	0.7062	0.8132	1791	2075	1146	4988
6	14788	6083	3917	0.6656	0.7441	0.6957	0.8000	1789	2128	1216	4867
7	14503	6148	3852	0.6797	0.7504	0.7198	0.7841	1976	1876	1327	4821
8	14638	6136	3864	0.6725	0.7476	0.7088	0.7914	1869	1995	1280	4856
9	14606	5985	4015	0.6716	0.7336	0.7128	0.7561	2191	1824	1460	4525
	14665.1	6094.9	3905.1	0.6743	0.7462	0.7104	0.7868	1947.3	1957.8	1298.8	4796.1

Fonte: a autora.

B.5 *k*-Nearest Neighbours

PIPELINE 1: STEMMING + BAG-OF-WORDS											
SEMENTE	ATRIBUTOS	POSITIVOS	NEGATIVOS	MÉTRICAS				MATRIZ DE CONFUSÃO			
				ACURÁCIA	F1 SCORE	PRECISION	RECALL	TN	FP	FN	TP
0	12090	6118	3882	0.5799	0.6380	0.6746	0.6069	2088	1794	2407	3711
1	12118	6084	3916	0.555	0.5721	0.6885	0.4899	2569	1347	3103	2981
2	12022	6071	3929	0.5471	0.5217	0.7275	0.4084	2992	937	3592	2479
3	12078	6124	3876	0.5906	0.6538	0.6782	0.6320	2036	1840	2254	3870
4	12030	6066	3934	0.5948	0.6993	0.6354	0.7783	1225	2709	1343	4723
5	12136	6134	3866	0.6124	0.7045	0.6612	0.7587	1470	2396	1480	4654
6	12137	6083	3917	0.5983	0.6820	0.6571	0.7098	1663	2254	1763	4320
7	11880	6148	3852	0.6172	0.7109	0.6626	0.7693	1438	2414	1414	4734
8	12043	6136	3864	0.5926	0.6656	0.6683	0.6667	1831	2033	2041	4095
9	12068	5985	4015	0.5793	0.6287	0.6666	0.5969	2222	1793	2414	3571
	12060.2	6094.9	3905.1	0.5867	0.6477	0.6720	0.6417	1953.4	1951.7	2181.1	3913.8

PIPELINE 2: STEMMING + TF-IDF											
SEMENTE	ATRIBUTOS	POSITIVOS	NEGATIVOS	MÉTRICAS				MATRIZ DE CONFUSÃO			
				ACURÁCIA	F1 SCORE	PRECISION	RECALL	TN	FP	FN	TP
0	12090	6118	3882	0.6599	0.7078	0.7471	0.6747	2471	1411	1990	4128
1	12118	6084	3916	0.6616	0.7036	0.7512	0.6637	2576	1340	2044	4040
2	12022	6071	3929	0.5902	0.5643	0.7880	0.4506	3166	763	3335	2736
3	12078	6124	3876	0.6865	0.7572	0.7204	0.7982	1977	1899	1236	4888
4	12030	6066	3934	0.6758	0.7661	0.6806	0.8769	1437	2497	745	5321
5	12136	6134	3866	0.6802	0.7729	0.6849	0.8879	1356	2510	688	5446
6	12137	6083	3917	0.6768	0.7493	0.7090	0.7947	1934	1983	1249	4834
7	11880	6148	3852	0.676	0.7479	0.7169	0.7851	1930	1922	1318	4830
8	12043	6136	3864	0.6661	0.7389	0.7090	0.7741	1908	1956	1383	4753
9	12068	5985	4015	0.6511	0.6887	0.7390	0.6471	2639	1376	2113	3872
	12060.2	6094.9	3905.1	0.6624	0.7197	0.7246	0.7353	2139.4	1765.7	1610.1	4484.8

PIPELINE 3: LEMATIZAÇÃO + BAG-OF-WORDS											
SEMENTE	ATRIBUTOS	POSITIVOS	NEGATIVOS	MÉTRICAS				MATRIZ DE CONFUSÃO			
				ACURÁCIA	F1 SCORE	PRECISION	RECALL	TN	FP	FN	TP
0	14706	6118	3882	0.5701	0.6226	0.6725	0.5819	2144	1738	2561	3557
1	14718	6084	3916	0.5464	0.5553	0.6875	0.4667	2625	1291	3245	2839
2	14620	6071	3929	0.5308	0.4902	0.7211	0.3733	3043	886	3806	2265
3	14737	6124	3876	0.5792	0.6345	0.6773	0.5977	2132	1744	2464	3660
4	14641	6066	3934	0.5911	0.6924	0.6365	0.7600	1300	2634	1455	4611
5	14694	6134	3866	0.6062	0.6954	0.6613	0.7379	1534	2332	1606	4528
6	14788	6083	3917	0.5923	0.6672	0.6616	0.6743	1818	2099	1978	4105
7	14503	6148	3852	0.6023	0.6963	0.6554	0.7454	1437	2415	1562	4586
8	14638	6136	3864	0.5883	0.6549	0.6732	0.6414	1946	1918	2199	3937
9	14606	5985	4015	0.5683	0.6079	0.6665	0.5614	2324	1691	2626	3359
	14665.1	6094.9	3905.1	0.5775	0.6317	0.6713	0.6140	2030.3	1874.8	2350.2	3744.7

PIPELINE 4: LEMATIZAÇÃO + TF-IDF											
SEMENTE	ATRIBUTOS	POSITIVOS	NEGATIVOS	MÉTRICAS				MATRIZ DE CONFUSÃO			
				ACURÁCIA	F1 SCORE	PRECISION	RECALL	TN	FP	FN	TP
0	14706	6118	3882	0.663	0.7089	0.7531	0.6720	2519	1363	2007	4111
1	14718	6084	3916	0.663	0.7045	0.7528	0.6643	2587	1329	2041	4043
2	14620	6071	3929	0.5886	0.5565	0.7955	0.4377	3227	702	3412	2659
3	14737	6124	3876	0.6867	0.7571	0.7210	0.7973	1985	1891	1242	4882
4	14641	6066	3934	0.6789	0.7676	0.6841	0.8756	1477	2457	754	5312
5	14694	6134	3866	0.6746	0.7684	0.6824	0.8804	1346	2520	734	5400
6	14788	6083	3917	0.6856	0.7542	0.7188	0.7936	2029	1888	1256	4827
7	14503	6148	3852	0.6767	0.7470	0.7193	0.7807	1963	1889	1344	4804
8	14638	6136	3864	0.6635	0.7355	0.7091	0.7660	1932	1932	1433	4703
9	14606	5985	4015	0.6531	0.6888	0.7432	0.6436	2680	1335	2134	3851
	14665.1	6094.9	3905.1	0.6634	0.7189	0.7279	0.7311	2174.5	1730.6	1635.7	4459.2

Fonte: a autora.

B.6 Multilayer Perceptron

PIPELINE 1: STEMMING + BAG-OF-WORDS											
SEMENTE	ATRIBUTOS	POSITIVOS	NEGATIVOS	MÉTRICAS				MATRIZ DE CONFUSÃO			
				ACURÁCIA	F1 SCORE	PRECISION	RECALL	TN	FP	FN	TP
0	12090	6118	3882	0.721	0.7747	0.7655	0.7845	2411	1471	131	4799
1	12118	6084	3916	0.7237	0.7762	0.7654	0.7875	2446	1470	1293	4791
2	12022	6071	3929	0.7181	0.7702	0.7618	0.7795	2447	1482	1337	4734
3	12078	6124	3876	0.7237	0.7789	0.7630	0.7957	2363	1513	1250	4874
4	12030	6066	3934	0.7238	0.7783	0.7579	0.8002	2384	1550	1212	4854
5	12136	6134	3866	0.7132	0.7710	0.7550	0.7879	2298	1568	1300	4834
6	12137	6083	3917	0.7163	0.7712	0.7572	0.7864	2381	1536	1301	4782
7	11880	6148	3852	0.7218	0.7785	0.7616	0.7970	2317	1535	1247	4901
8	12043	6136	3864	0.7174	0.7743	0.7589	0.7907	2323	1541	1285	4851
9	12068	5985	4015	0.7128	0.7614	0.7572	0.7661	2543	1472	1400	4585
	12060.2	6094.9	3905.1	0.7192	0.7735	0.7603	0.7876	2391.3	1513.8	1175.6	4800.5

PIPELINE 2: STEMMING + TF-IDF											
SEMENTE	ATRIBUTOS	POSITIVOS	NEGATIVOS	MÉTRICAS				MATRIZ DE CONFUSÃO			
				ACURÁCIA	F1 SCORE	PRECISION	RECALL	TN	FP	FN	TP
0	12090	6118	3882	0.7102	0.7678	0.7540	0.7846	2307	1575	1323	4795
1	12118	6084	3916	0.7191	0.7738	0.7581	0.7906	2380	1536	1273	4811
2	12022	6071	3929	0.7113	0.7658	0.7541	0.7784	2386	1543	1344	4727
3	12078	6124	3876	0.719	0.7758	0.7591	0.7944	2327	1549	1261	4863
4	12030	6066	3934	0.7116	0.7707	0.7445	0.7995	2268	1666	1218	4848
5	12136	6134	3866	0.7024	0.7675	0.7373	0.8015	2109	1757	1219	4915
6	12137	6083	3917	0.7039	0.7639	0.7416	0.7889	2240	1677	1284	4799
7	11880	6148	3852	0.7164	0.7764	0.7523	0.8029	2227	1625	1211	4937
8	12043	6136	3864	0.7034	0.7628	0.7485	0.7791	2251	1613	1353	4783
9	12068	5985	4015	0.7032	0.7520	0.7519	0.7531	2526	1489	1479	4506
	12060.2	6094.9	3905.1	0.7101	0.7676	0.7501	0.7873	2302.1	1603	1296.5	4798.4

PIPELINE 3: LEMATIZAÇÃO + BAG-OF-WORDS											
SEMENTE	ATRIBUTOS	POSITIVOS	NEGATIVOS	MÉTRICAS				MATRIZ DE CONFUSÃO			
				ACURÁCIA	F1 SCORE	PRECISION	RECALL	TN	FP	FN	TP
0	14706	6118	3882	0.7199	0.7744	0.7631	0.7868	2387	1495	1306	4812
1	14718	6084	3916	0.7203	0.7710	0.7672	0.7753	2484	1432	1365	4719
2	14620	6071	3929	0.7195	0.7712	0.7629	0.7801	2458	1471	1334	4737
3	14737	6124	3876	0.7174	0.7748	0.7564	0.7943	2309	1567	1259	4865
4	14641	6066	3934	0.721	0.7744	0.7602	0.7901	2419	1515	1275	4791
5	14694	6134	3866	0.7114	0.7704	0.7518	0.7906	2263	1603	1283	4851
6	14788	6083	3917	0.7137	0.7681	0.7573	0.7803	2392	1525	1338	4745
7	14503	6148	3852	0.7204	0.7765	0.7634	0.7906	2343	1509	1287	4861
8	14638	6136	3864	0.7133	0.7709	0.7558	0.7870	2305	1559	1308	4828
9	14606	5985	4015	0.7148	0.7651	0.7542	0.7767	2500	1515	1337	4648
	14665.1	6094.9	3905.1	0.7172	0.7717	0.7592	0.7852	2386	1519.1	1309.2	4785.7

PIPELINE 4: LEMATIZAÇÃO + TF-IDF											
SEMENTE	ATRIBUTOS	POSITIVOS	NEGATIVOS	MÉTRICAS				MATRIZ DE CONFUSÃO			
				ACURÁCIA	F1 SCORE	PRECISION	RECALL	TN	FP	FN	TP
0	14706	6118	3882	0.7164	0.7731	0.7571	0.7908	2325	1557	1279	4839
1	14718	6084	3916	0.7181	0.7741	0.7550	0.7946	2345	1571	1248	4836
2	14620	6071	3929	0.7088	0.7643	0.7508	0.7790	2358	1571	1341	4730
3	14737	6124	3876	0.7198	0.7797	0.7520	0.8106	2235	1641	1161	4963
4	14641	6066	3934	0.7212	0.7806	0.7463	0.8190	224	1690	1098	4968
5	14694	6134	3866	0.7006	0.7627	0.7425	0.7851	2190	1676	1318	4816
6	14788	6083	3917	0.7095	0.7663	0.7515	0.7834	2331	1586	1319	4764
7	14503	6148	3852	0.7128	0.7728	0.7512	0.7968	2228	1624	1248	4900
8	14638	6136	3864	0.7001	0.7620	0.7424	0.7834	2194	1670	1329	4807
9	14606	5985	4015	0.7058	0.7601	0.7424	0.7790	2397	1618	1324	4661
	14665.1	6094.9	3905.1	0.7113	0.7696	0.7491	0.7921	2082.7	1620.4	1266.5	4828.4

Fonte: a autora.