Ernanny Figueiredo

Classificação de sentimentos em textos de *e-commerce* utilizando redes neurais artificiais.

Cascavel-PR

2022

Ernanny Figueiredo

Classificação de sentimentos em textos de *e-commerce* utilizando redes neurais artificiais.

Dissertação apresentada como requisito parcial para a obtenção do grau de Mestre pelo Programa de Pós-Graduação em Ciência da Computação (PPGComp) da Universidade Estadual do Oeste do Paraná – Unioeste, campus de Cascavel.

Universidade Estadual do Oeste do Paraná – Unioeste – Cascavel

Centro de Ciências Exatas e Tecnológicas – CCET

Programa de Pós-Graduação em Ciência da Computação – PPGComp

Orientador: Profª Drª Simone Aparecida Miloca

Cascavel-PR

2022

Ernanny Figueiredo
Classificação de sentimentos em textos de *e-commerce* utilizando redes neurais artificiais./ Ernanny Figueiredo. – Cascavel-PR, 2022-
71p. : il. (algumas color.) ; 30 cm.

Orientador: Profª Drª Simone Aparecida Miloca

Dissertação (Mestrado)– Universidade Estadual do Oeste do Paraná – Unioeste – Cascavel
Centro de Ciências Exatas e Tecnológicas – CCET
Programa de Pós-Graduação em Ciência da Computação – PPGComp, 2022.

1. Classificação de textos. 2. Redes neurais. 2. Aprendizagem de máquina.

Ernanny Figueiredo

Classificação de sentimentos em textos de *e-commerce* utilizando redes neurais artificiais.

Dissertação apresentada como requisito parcial para a obtenção do grau de Mestre pelo Programa de Pós-Graduação em Ciência da Computação (PPGComp) da Universidade Estadual do Oeste do Paraná – Unioeste, campus de Cascavel.

Trabalho aprovado. Cascavel-PR, 04 de fevereiro de 2022.

Profª Drª Simone Aparecida Miloca
Orientador(a)

Profª Drª Rosangela Villwock
Universidade Estadual do Oeste do Paraná,
UNIOESTE

Profª Drª Giani Carla Ito
Universidade Tecnológica Federal do Paraná,
UTFPR

Cascavel-PR
2022

Agradecimentos

Meus agradecimentos não são direcionados apenas às pessoas que participaram desta fase de mestrado, é mister lembrar de todas as pessoas que me ajudaram a chegar até aqui, pois antes de um cientista, antes de um acadêmico, existe um ser humano sendo educado e preparado para seguir seu sonho, portanto, além de agradecer a Deus, faço menção de meus pais: Geonil Figueiredo e Nelci Maria Figueiredo, que desde cedo lutaram com o objetivo de me dar a melhor educação e mostrar os melhores caminhos, meu irmão Giovanny Figueiredo, que sempre me incentivou a buscar algo a mais. Sou grato também por minha esposa, Daiane de Souza Figueiredo, que mesmo quando eu achava que não era possível chegar ao fim dessa jornada, soube usar as palavras certas para me motivar e me fazer prosseguir. Ainda em tempo, fica registrado aqui também minha gratidão aos professores da instituição, à minha orientadora, Dra. Simone Aparecida Miloca e ao meu empregador, Eloi da Silva Carvalho, que acreditou no potencial do projeto e viabilizou a pesquisa.

"Não vos amoldeis às estruturas deste mundo,
mas transformai-vos pela renovação da mente,
a fim de distinguir qual é a vontade de Deus:
o que é bom, o que Lhe é agradável, o que é perfeito.
(Bíblia Sagrada, Romanos 12, 2)

Resumo

Todos os dias, milhões de pessoas compartilham abertamente, nas redes sociais e páginas de comentários, suas opiniões sobre determinados assuntos, produtos, serviços, etc. Diversos segmentos do mercado empresarial têm interesse em extrair informações desse meio, que sejam relevantes para seu negócio. Um tipo de informação desejada é a identificação de sentimentos expressos pelos usuários registrados na forma de opiniões, já que isso demonstra a aceitação ou rejeição com relação ao assunto. A obtenção de tais informações de forma manual muitas vezes é inviável devido a grande quantidade de textos e ai entram as técnicas de aprendizado de máquina permitindo organizar, gerenciar e extrair conhecimento, possibilitando ao utilizador da solução melhorar sua estratégia de negócio. Este trabalho propõe uma abordagem para o problema de classificação de textos aplicado à análise de sentimentos, para identificar a polaridade do texto, ou seja, saber se a opinião é positiva ou negativa. A literatura indica diversas ferramentas com classificadores diferentes, sendo as utilizadas neste trabalho aquelas cujos modelos construídos incorporam classificadores baseados em redes neurais artificiais. Modelos foram construídos e seu desempenho avaliado para um grupo particular de dados que contém opiniões de consumidores que adquiriram produtos da área da saúde, com textos escritos na língua portuguesa. Também investigou-se o impacto das fases de pré-processamento do texto nos modelos. Os resultados mostraram que as soluções de redes neurais artificiais, tanto as multi camadas quanto as recorrentes, implementadas em Python, atingem um nível de eficiência próximo das melhores e mais difundidas ferramentas comerciais destinadas à esta tarefa.

Palavras-chave: classificação de textos. redes neurais. aprendizagem de máquina.

Abstract

Every day, millions of people openly share their opinions on social media and comment sites about specific topics, products, services, etc. Several segments of the business market are interested in gaining information from this medium that is relevant to their business. One type of desired information is the identification of sentiments expressed by registered users in the form of opinions, as this shows agreement or disagreement related to the topic. Manual collection of such information is often not feasible due to the large amount of text. This is where machine learning techniques come into play, allowing you to organize, manage and extract knowledge so that the user of the solution can improve their business strategy. This work proposes an approach to the text classification problem applied to sentiment analysis to identify the polarity of the text, i.e., to know whether the opinion is positive or negative. The literature indicates several tools with different classifiers can be found in the ones used in this work are those whose built models incorporate classifiers based on artificial neural networks. The models were created and their performance was evaluated for a specific set of data containing the opinions of consumers who purchased health care products, with texts written in Portuguese. The effect of the preprocessing stages of the texts on the models was also studied. The results showed that artificial neural network solutions, both multilayer and recurrent, implemented in Python, reach an efficiency level close to the best and most widespread commercial tools for this task.

Keywords: text classification. neural networks. machine learning.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

ABNT	Associação Brasileira de Normas Técnicas
RNA	Redes Neurais Artificiais
NB	Naïve Bayes
NN	Neural Network
MLP	Multi-Layer Perceptron
RNN	Recurrent Neural Network
LLSF	Linear Squares Fit
kNN	k-Nearest-Neighbor
SVM	Support Vector Machines
LGPD	Lei Geral de Proteção de Dados
KDT	Knowledge Discovery in Text
BOW	Bag of words
VSM	Vector Space Model
TF	Term frequency
TFIDF	Term frequency - Inverse document frequency
NLP	Natural Language Process
RBF	Redes de função de base radial
RBM	Máquinas de Boltzmann restritas
CNN	Redes neurais convolucionais
LSTM	Long Short Term Memory
API	Application programming interface
DAN2	Dynamic Architeture for Artificial Neural Networks
UCI	Universidade da Califórnia Irvine

Sumário

1 Introdução

Cada vez mais as empresas e organizações estão sendo imersas em um mundo tecnológico que disponibiliza grande quantidade de informações e assim sentem a necessidade, em algum momento, de organizar e extrair informações relevantes que possam auxiliar principalmente nas tomadas de decisões.

Com a popularização das redes sociais, as postagens, também denominadas microtextos, viraram uma fonte vasta de informação relevante, visto que é a opinião direta do consumidor final, isso abre um leque de possibilidades para o cientista de dados, que usando os mais diversos meios, consegue extrair e classificar os dados úteis para cada necessidade.

Por exemplo, um fabricante lança um determinado produto, onde é feito um vídeo de *review* e postado na plataforma YouTube, que possui atualmente cerca de 2 bilhões de usuários, (YOUTUBE, 2021), em pouco tempo os consumidores irão assistir esse vídeo e comentar a postagem deixando a sua opinião, se é de interesse ou não, a respeito do assunto. Para entender um caso isolado, como neste exemplo, bastaria ler os comentários e classificar, quais são bons, e direcionar para o time de marketing atuar com prospecção de possíveis clientes, e quais são ruins, ou que apresentaram críticas a respeito, possibilitando assim à equipe responsável melhorar o produto. Agora imaginemos fazer isso como um processo de trabalho, com diversos produtos ou serviços, o volume é tanto que se fosse feito apenas por uma pessoa ou várias, seria inviável.

Segundo Ghiassi et al. (2012), a maioria dos problemas de classificação na vida real, incluindo classificação de texto, classificação de imagens e outros, são de natureza complexa e caracterizada por alta dimensionalidade. Para resolver estes problemas é preciso estratificar a informação e processá-la em baixo nível. Tais procedimentos fazem parte de uma área denominada mineração de textos.

Existem várias visões e definições de mineração de textos, para Moura (2004) é uma área de pesquisa tecnológica cujo objetivo é a busca por padrões, tendências e regularidades em textos escritos em linguagem natural. Já Loh, Wives e Frainer (2004) menciona que a mineração de textos é entendida como uma das aplicações de técnicas de KDD (*Knowledge Discovery From Data*), sobre dados extraídos de textos.

Dentre as tarefas presentes na mineração de textos há uma que vem despertando o interesse de muitos pesquisadores e usuários que é a mineração de opinião, também chamada de análise de sentimento, devido a evolução conceitual presente em ambos os termos. Segundo (KUMAR, 2012), a análise de sentimento é uma tarefa interdisciplinar, desafiadora e complexa que inclui processamento de linguagem natural, mineração na web

e aprendizado de máquina. Existem muitas aplicações práticas e potenciais da análise de sentimento, sendo uma delas o foco deste trabalho, que esta ligada a área de negócios. A análise de sentimento pode ajudar as organizações e prestadores de serviços a conhecer melhor seus clientes e a adaptar seus produtos e serviços às necessidades dos clientes e usuários. Kumar (2012) apresenta o estado da arte em análise de sentimento e os desafios de pesquisa e pelos trabalhos analisados nota-se que diferentes tipos de textos podem requer métodos especializados para análise por exemplo, sentimentos não são expressos da mesma maneira em textos jornalísticos, blogs, comentários, fóruns, mensagens em redes sociais entre outros.

Assim, a proposta deste trabalho envolve a abordagem para classificação de textos no contexto empresarial com o objetivo de extrair informações quanto a opinião do consumidor com relação ao produto adquirido. Sendo a informação principal o sentimento expresso na opinião, classificado como positivo ou negativo.

1.1 Contribuição e Organização do Texto

Neste trabalho buscamos uma solução e investigamos ferramentas para o problema de classificação de sentimentos expressos em textos contendo opiniões de consumidores que adquiriram produtos na área da saúde. A análise de sentimento foi efetuada considerando dois níveis de sentimento (positivo e negativo).

Algoritmos classificadores baseados em redes neurais artificiais fazem parte da solução proposta. As arquiteturas de redes utilizadas foram a *Multi-Layer Perceptron* e Recorrentes. Análises comparativas utilizando-se métricas propostas na literatura foram efetuadas. Analisamos também a influência de técnicas de pré-processamento na acurácia dos classificadores.

A proposta é obter um modelo com boa acurácia para o problema de classificação mas que seja viável economicamente, ou seja, que consiga extrair informação útil para o usuário sem altos custos. Neste sentido a linguagem Python foi utilizada nas implementações, incorporando suas bibliotecas de aprendizado de máquina. Com esta implementação pretende-se também avaliar fatores relacionados aos parâmetros exigidos pelos modelos bem como a importância de pré-processamento. Também buscamos na literatura ferramentas, na versão não comercial, que forneçam solução para o problema.

No Capítulo 2 apresentamos a fundamentação teórica, no Capítulo 3 a revisão de literatura contendo alguns trabalhos relacionados ao tema. No Capítulo 4, os experimentos assim como a metodologia e os dados que foram utilizados. No Capítulo 5 temos os resultados numéricos e por fim, no Capítulo 6 as considerações finais.

2 Fundamentação Teórica

2.1 E-commerce

E-commerce é a atividade mercantil que, em última análise, vai fazer a conexão eletrônica entre a empresa e o cliente para a venda de produtos ou serviços (SEBRAE, 2013). Conforme definição mencionada, esta área não é nova, mas ganhou notoriedade nos últimos anos, facilitando o comercio de produtos e serviços, difundindo ainda mais o efeito da globalização.

Um *e-commerce*, ou comércio eletrônico, refere-se aos negócios que estruturam seu processo de compra e venda na Internet. Todas as transações comerciais são realizadas por meio de ferramentas online. Assim, fica fácil entender que o conceito de *e-commerce* envolve muito mais do que apenas a criação de um site. Trata-se de um tipo de empreendimento que se diferencia pela e sua estrutura de funcionamento, totalmente digital.

Quando afirmamos isso, vale a pena destacar que o *e-commerce* digitaliza integralmente dois processos básicos: venda e atendimento ao cliente. A partir desse trabalho, ele também abre as portas para outras automações, como marketing, controle de finanças e estoque. Dessa maneira, ele facilita e agiliza o trabalho de gestão em muitas frentes. Por outro lado, também tem como efeito o maior peso estratégico da gestão da informação.

Segundo o portal E-commerce Brasil, (BRASIL, 2021), o e-commerce no Brasil bateu recorde de vendas no primeiro semestre de 2021, atingindo R$ 53,4 bilhões, crescimento de 31% em relação ao mesmo período do ano anterior, mostrando o acentuado crescimento deste mercado virtual.

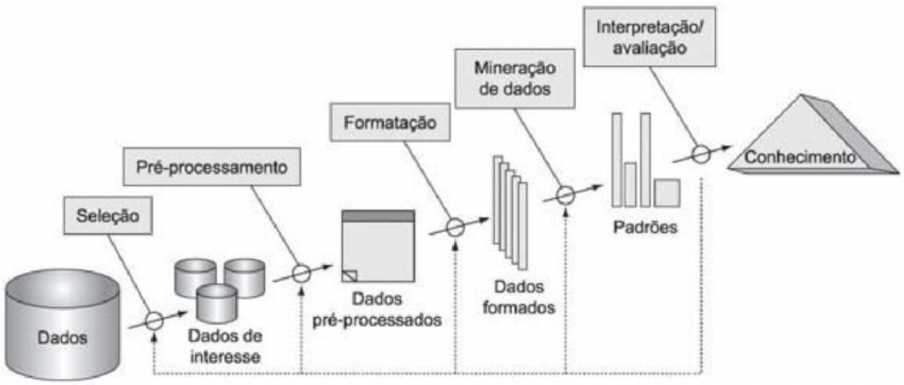
A pandemia da covid-19 mudou as dinâmicas de consumo e fez o varejo, em especial o *e-commerce*, passar por um processo de aceleração. Esse processo movimenta também outras áreas do negócio e é necessário se adaptar. Hoje vivemos um mundo de comunidades, uma vida nas mídias sociais, e o usuários estão lá. Eles é que estão avaliando os serviços e produtos.

Dados da ABComm (Associação Brasileira de Comércio Eletrônico), (4ALL, 2020), mostram que no ano de 2020 mais de 135 mil lojas brasileiras aderiram às vendas por *e-commerce*, na tentativa de se manterem ativas no mercado. As restrições da pandemia também fizeram com que até mesmo usuários que não tinham o hábito fazer compras pela internet aderissem a este novo método.

2.2 Mineração de textos

Mineração de textos é a descoberta por computador de novas informações anteriormente desconhecidas, extraindo automaticamente informações de diferentes recursos escritos em linguagem natural que possuem algum padrão, tendência ou regularidade (MOURA, 2004). Um elemento-chave é a vinculação das informações extraídas para formar novos fatos ou novas hipóteses a serem exploradas posteriormente por meios mais convencionais de experimentação. A mineração de textos busca extrair informações em base de dados textuais, denominada de KDT *(Knowledge Discovery in Text)*, uma das aplicações de técnicas de KDD *(Knowledge Discovery from Data)* que Loh, Wives e Frainer (2004) descreve em sua pesquisa. Este processo visa identificar fatos, relacionamentos e afirmações que, de outra forma, permaneceriam enterrados na massa de dados textuais. Os conjuntos de dados de documentos de texto têm uma série de características específicas que requerem explicação separada. Fazem parte do KDT as etapas: Pré-processamento, formatação, mineração, avaliação e descoberta do conhecimento, conforme observado na Figura 1.

Figura 1 – Representação KDD



Fonte: Adaptado pelo autor

A mineração de textos, como análise de dados exploratória, é um método para apoiar pesquisadores a buscar novas e relevantes informações de uma grande coleção de textos. É um processo parcialmente automatizado onde o pesquisador ainda está envolvido, interagindo com o sistema (HEARST, 1999)

O uso da mineração de textos depende de outra área de estudo, denominada de processamento de linguagem natural (NLP). Este estudo é necessário para entender o sentimento expresso nas palavras, assim como transformar esta informação para algo que o algoritmo possa trabalhar, assunto explorado na Seção 2.7. Gartner (2021) define que a tecnologia de processamento de linguagem natural envolve a capacidade de transformar texto ou voz em áudio em informações codificadas e estruturadas, com base em uma ontologia apropriada. Os dados estruturados podem ser usados para adquirir alguma informação ou conhecimento.

Na Seção 2.3 será explanado o pré-processamento de texto, assunto que merece um atenção à parte, pois é a base para aquisição do conhecimento. As diversas fases de transformação que o texto não estruturado passa nesse processo é o que faz com que o dado possa ser interpretado pelo modelo e assim transformado em informação de valor.

2.3 Pré-processamento de texto

Todos os algoritmos especializados em classificação de texto têm sua eficiência, mas para chegar ao seu resultado, eles necessitam de uma fase em comum, o pré-processamento do texto, conhecida como *Data Preparation* ou preparação dos dados. O principal objetivo da etapa de pré-processamento do texto é aumentar a qualidade inicial dos dados, para isso diversas etapas podem ser aplicadas e até mesmo combinadas para atingir um nível adequado. São exemplos a remoção de palavras irrelevantes, como preposições ou artigos, a formatação dos textos, a representação por radical e o cálculo de relevância dos termos para identificar os mais significativos.

Em conjuntos de dados do mundo real, valores errôneos podem ser registrados por uma variedade de razões, incluindo erros de edição, julgamentos subjetivos e mau funcionamento, ou uso indevido do equipamento de correção ortográfica automática. Essas questões são levantadas e podem ser tratadas também na fase de pré-processamento.

Por fim, os textos são representados em um formato estruturado que preserve as principais características dos dados (REZENDE; MARCACINI; MOURA, 2011).

2.3.1 Representação de documentos textuais

Existem várias maneiras de realizar a conversão de documentos de texto simples para instâncias com um número fixo de atributos. Por exemplo, pode-se contar o número de vezes que as frases especificadas ocorrem, ou tomar uma combinação de duas palavras consecutivas, ou ainda contar a ocorrência de duas ou três combinações de caracteres (conhecidas como bigramas e trigramas, respectivamente). Uma representação simples baseada em palavras é a mais comumente utilizada, conhecida como representação de saco de palavras (BoW, do inglês *bag-of-words*),(BRAMER, 2016).

Em processamento de linguagem natural, denominamos corpus (ou documentos) toda uma coleção de textos que temos. Um corpus pode ser decomposto em fragmentos. Os fragmentos podem ser frases simples, parágrafos ou documentos de várias páginas, sendo algo que desejamos tratar como uma amostra. Por exemplo, se estivermos analisando documentos clínicos, cada documento de admissão de paciente pode ser um fragmento; se estivermos analisando o sentimento nas mídias sociais, cada comentário do usuário é um fragmento; e assim por diante.

Um modelo de saco de palavras é feito transformando cada palavra do corpus em um recurso e em cada linha, sob aquela palavra, conta-se quantas vezes a palavra ocorre naquele fragmento. Nesta abordagem, a ordem das palavras é perdida, porém é uma das principais formas de converter a linguagem em recursos a serem alimentados em um algoritmo de aprendizado de máquina. A literatura apresenta geralmente duas possibilidades de se trabalhar, ambas envolvem a construção de um dicionário de palavras para uma coleção de documentos. A primeira é a abordagem do dicionário local, usando apenas as palavras que aparecem nos documentos classificados como pertencentes a determinada categoria. Isso permite que cada dicionário seja relativamente pequeno ao custo de precisar construir N deles, onde há N categorias. A segunda abordagem é construir um dicionário global, que inclui todas as palavras que ocorrem pelo menos uma vez em qualquer um dos documentos. Detalhes em (SKANSI, 2018).

A seguir, na Tabela 1, apresentamos um exemplo de saco de palavras para um conjunto de mídia social simples.

Tabela 1 – Exemplo de entrada de saco de palavras

Usuário	Comentário
S.A	produto bom
A.V	prazo de entrega ruim
E. F	produto bom, prazo de entrega ruim
P. M	prazo de entrega bom, produto ruim

Produzido pelo autor

Para criar um saco de palavras a partir dos comentários, precisamos fazer duas passagens. A primeira apenas coleta todas as palavras que ocorrem e as transforma em recursos, ou seja, coleta as palavras e cria colunas a partir delas, e a segunda, escreve nos valores reais, conforme a Tabela 2:

Tabela 2 – Exemplo de saco de palavras

Usuário	produto	bom	prazo	de	entrega	ruim
S.A	1	1	0	0	0	0
A.V	0	0	1	1	1	1
E. F	1	1	1	1	1	1
P. M	1	1	1	1	1	1

Produzido pelo autor

De qualquer forma, é importante considerar também a redução do tamanho do espaço de recursos (ou seja, o conjunto de palavras incluídas no dicionário) (BRAMER, 2016). Neste sentido, deve-se remover a pontuação que também facilita o caminho para uma segmentação de palavras adequada.

O processo em que um texto é segmentado em sentenças e palavras, formando pedaços é chamado de *t*okens (Tokenização), normalmente é o primeiro passo da etapa de

pré-processamento, reservando unidades mínimas de texto para análise.

2.3.2 Tokenização

A primeira etapa do pré-processamento normalmente consiste em buscar uma atomização do texto livre, formando assim unidades mínimas de informação, o *token*. Este *token* é formado por uma única palavra, ou em alguns outros casos por uma pontuação ou um caractere especial (HEARST, 1999). Para compreender melhor, acompanhe o exemplo:

Texto livre: "produto bom, prazo de entrega ruim"

Tokens: [produto] [bom] [,] [prazo] [de] [entrega] [ruim]

2.3.3 *Stopwords e stemming*

Uma abordagem amplamente usada para redução do número de palavras é usar uma lista de palavras comuns que provavelmente não são úteis para classificação, conhecida como palavras de parada (*stop words*), e remover todas as ocorrências dessas palavras antes de criar a representação do saco de palavras. Deve-se observar que para cada idioma, o conjunto de palavras de parada deve ser independente, pois deve-se levar em consideração o impacto no entendimento da mensagem e se aplicado indevidamente, perde o sentido da expressão. Essas palavras, normalmente repetidas nas frases, acabam dificultando o processo de identificação do conteúdo. Podemos citar como exemplo: as, e, os, de, para, com, sem, foi. (SANTOS, 2018)

Outro processo para redução do número de palavras na representação é o *stemming*. Em linguística computacional consiste em eliminar variações morfológicas de uma palavra através da identificação do radical. Por exemplo, podemos usar as palavras computação, computador, computar, computacional, computável e computabilidade no mesmo documento. Essas palavras têm a mesma raiz linguística (“comput”). Colocá-las juntas como se fossem ocorrências de uma única palavra provavelmente daria uma forte indicação do conteúdo. Muitas vezes o *stemming* é confundido com o processo de lematização, Nunes e Lucca (2002) esclarece que é necessário compreender que a lematização existe puramente no contexto lexicográfico, *stemming* não. Lematização é a representação da palavra através de seu masculino singular, adjetivos e substantivos e infinitivo, apenas no contexto da lexicologia. Já o *stemming* é a retirada de sufixos do radical. Assim, as estruturas são distintas, embora eventualmente possam ser graficamente semelhantes.

2.3.4 Modelo espaço vetorial

Supondo que um dicionário local ou global seja utilizado, que uma representação que substitui cada documento por uma série de N características seja efetuada e considerando uma representação de saco de palavras, tem-se o primeiro documento como um conjunto

ordenado de N valores, chamado de vetor N-dimensional. Deste modo é possível representar geometricamente os documentos por pontos ou vetores contidos em um espaço Euclidiano N-dimensional de modo que cada dimensão corresponde a uma palavra do dicionário. O conjunto completo de vetores para todos os documentos em consideração é chamado modelo de espaço vetorial (VSM - *Vector Space Model*). A cada palavra ou termo do vetor associa-se um peso que indica a importância da palavra em relação ao texto. Existem várias fórmulas para cálculo do peso, dentre elas destacam-se a frequência absoluta, frequência relativa e frequência inversa de documentos ($TFIDF$) (BRAMER, 2016).

O valor $TFIDF$ é definido pela Equação 2.1.

$$TFIDF_{(i,j)} = TF(j,i) * IDF(j) \tag{2.1}$$

onde $TF(j,i)$ indica a frequência do termo j no documento i que é multiplicada pelo IDF (frequência inversa do documento) que é calculada pela Equação 2.2.

$$IDF(j) = log\frac{n}{n_j} \tag{2.2}$$

onde n_j é o número de documentos contendo o termo j e n o número total de documentos. Esta fórmula tende a tornar os termos raros na coleção de documentos mais importantes que outros. Se um termo ocorrer em todos os documentos, o valor inverso da frequência do documento é 1.

Normalmente estas estratégias são utilizadas em conjunto e objetivam chegar a um texto estruturado e normatizado, dentro de um padrão pré-definido, para que um procedimento de classificação possa ser utilizado. Após o pré-processamento dos textos, o conjunto de dados é dividido em dados de treinamento e de teste e o processo de classificação é iniciado.

Conforme veremos mais a frente, conceito do $TFIDF$ é a base também para outras soluções, pois com ele é possível trazer uma informação textual para uma base matemática. Com a evolução das técnicas de classificação, outras soluções foram sendo construídas utilizando-se da mesma ideia, mas com outras abordagens para montar o vetor.

2.3.5 *Embedding*

Um *Embedding* é uma forma compacta de representar dados textuais. Representa um vetor de alta dimensão em um espaço de baixa dimensionalidade, (CARVALHO, 2018).

A literatura apresenta diversos modelos, incluindo modelos de linguagem de redes neurais (NNLM, na sigla em inglês), vetores globais para representação de palavras (GloVe, na sigla em inglês), representações profundas de palavras contextualizadas (ELMo, na sigla

em inglês) e Word2vec, que podem ser construídos para aprender *embeddings* de palavras, que são vetores de atributos com valores reais para cada palavra.

Como exemplo, tomando uma frase em um documento, podemos representá-la como um vetor onde cada entrada representa o número de vezes que a palavra aparece na frase. Considerando frases curtas, é bem pouco provável que uma única frase tenha mais de 50 palavras, de modo que quase todas as entradas do vetor terão valor igual a zero, de modo que o vetor seja esparso e com alta dimensionalidade. Na representação por *embedding*, a palavra é codificada semanticamente usando a mesma quantidade de atributos que o vetor de modo a deixá-lo comparativamente pequeno, mas denso, com apenas algumas centenas de elementos. Aplicações de *embeddings* em problemas envolvendo classificação de textos podem ser vistas em Souza, Gonçalves e Souza (2020).

Neste trabalho adotamos o GloVe, *Global Vectors for Word Representation*, (GLOVE, 2014), que é um método alternativo para criar *embeddings* de palavras. Ele é baseado em técnicas de fatoração de matrizes na matriz de contexto de palavras, ou seja, uma grande matriz de informações de co-ocorrência é construída e é contada cada "palavra" (as linhas) e com que frequência ocorre essa palavra em algum "contexto" (as colunas) em um grande corpus. Para cada termo, buscamos termos de contexto dentro de alguma área definida por um tamanho de janela antes do termo e um tamanho de janela depois do termo. Além disso, damos menos peso para palavras mais distantes.

O número de "contextos" é, obviamente, grande, uma vez que é essencialmente combinatório em tamanho. Então, é fatorada essa matriz para produzir uma matriz de dimensão inferior, onde cada linha agora produz uma representação vetorial para cada palavra. Em geral, isso é feito minimizando uma "perda de reconstrução". Esta perda tenta encontrar as representações dimensionais inferiores que podem explicar a maior parte da variação nos dados dimensionais elevados.

O artigo publicado por GloVe (2014) traz uma explicação mais detalhada do algoritmo. Neste mesmo trabalho, os autores mostram que o modelo GloVe apresenta uma melhor performance sobre os modelos comparados CBOW[1], SVD[2] e *Skip-Gram*[3] em todas as tarefas definidas de processamento de linguagem natural (NLP) nos experimentos, alcançando uma acurácia de 75% na tarefa de analogia das palavras (CARVALHO, 2018).

[1] *Contextual Bag-of-Words*, rede neural linear que é treinada para prever uma palavra a partir do contexto em que ela ocorre, (FACURE, 2017)

[2] *Singular Value Decomposition*, é a fatoração de uma matriz real ou complexa em valores singulares

[3] Modelo utilizado para prever a palavra de contexto para uma determinada palavra-alvo, em um processo inverso ao CBOW, (TOWARDS, 2019)

2.4 Classificadores

A informação desejada no contexto deste trabalho será obtida utilizando-se um tipo específico de tarefa que é a classificação. No processo de classificação, o conteúdo é submetido a um modelo pré-definido que atribui a informação contida à uma classe específica. Segundo Bramer (2016), é possível utilizar qualquer um dos métodos padrão de classificação presentes na literatura: *Naïve Bayes (NB)*, Rede Neural (NN), *Linear Least Squares Fit (LLSF)*, *k-Nearest-Neighbor (kNN)* e *Support Vector Machines (SVM)*. Esta escolha não é fixa, mas livre para selecionar o modelo que melhor resolva o problema.

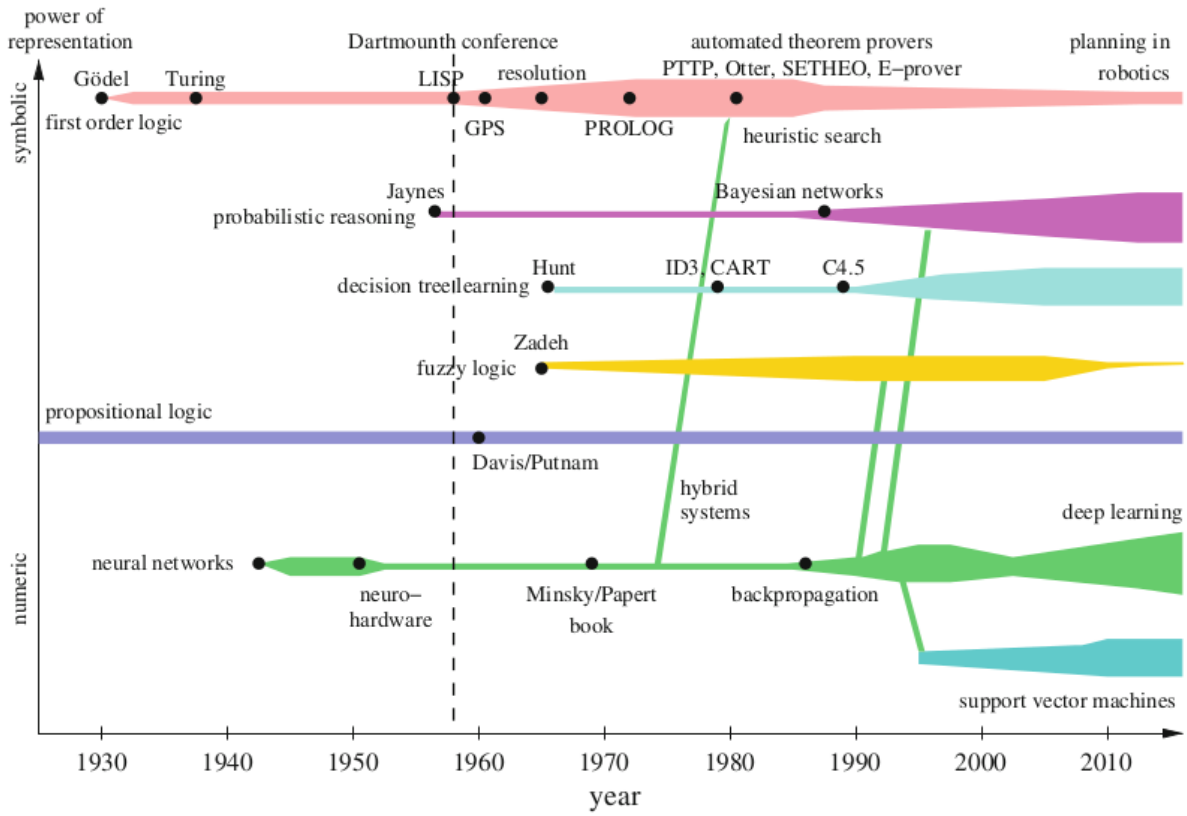
No contexto de classificação, estratégias de aprendizado profundo são atualmente as ferramentas mais utilizadas sendo baseadas em arquiteturas de rede neurais (IGUAL, 2017), que foi a escolhida para utilização neste trabalho.

Ultimamente muito se fala sobre inteligência artificial, aprendizado profundo, redes neurais artificiais (RNAs) e inúmeros são os termos associados à classificação e predição. Embora este assunto não seja novo, a evolução de modelos e aplicações nos últimos anos tem aumentado significativamente. Os registros de estudos na área e publicações datam de 1943, que incluem nomes como McCulloch e Pitts, Wiener, Hebb, Rosenblatt, Windrow e Hoff, dentre outros que, com o passar dos anos foram aprimorando o conceito e ampliando sua aplicabilidade, tornando o campo de estudo vasto, com possibilidades de aprimoramento visíveis. Na virada do século, em virtude da maior disponibilidade de dados e o aumento do poder computacional levaram a um maior sucesso das redes neurais, e essa área renasceu sob o novo rótulo de *deep learning* (aprendizado profundo), sendo a principal diferença o número de camadas ocultas, pois em modelos *deep learning* tem-se mais de uma camada oculta, especializando o modelo de aprendizado de características.

A Figura 2 mostra o histórico das linhas de estudo e destaca pontos de tal evolução na espessura das linhas. Em verde, vemos o histórico das redes neurais, que aparecem nos anos 40, com uma boa expressividade, percebe-se que próximo de 1975 as redes neurais começam a se ramificar para outras áreas tais como sistemas híbridos e buscas heurísticas. Mas a grande expansão ocorre mesmo próximo aos anos 90, quando David Rumelhart, Geoffrey Hinton e Ronald Williams publicaram o famoso artigo *Learning representations by backpropagating errors*, nele os autores descrevem várias redes neurais em que o algoritmo *backpropagation* funciona muito mais rapidamente do que as abordagens anteriores de aprendizado, possibilitando o uso de redes neurais para resolver problemas que antes eram insolúveis (SCIENCE, 2021). A partir dessa implementação, diversas outras soluções na área de redes neurais começaram a surgir, conforme pode ser visto pelas ramificações.

As redes neurais artificiais podem ser muito versáteis, pois a construção do algoritmo é feita com foco no melhor resultado para a solução de cada problema. Existem vários tipos de arquiteturas de redes neurais que são comumente usadas em diferentes aplicações.

Figura 2 – Evolução das áreas de Inteligência artificial



Fonte: Extraído de Ertel (2017)

Segundo Aggarwal (2018), a maioria dos modelos básicos de aprendizado de máquina, como regressão linear, classificação, máquinas de vetores de suporte, regressão logística, decomposição de valor singular e fatoração de matriz, podem ser simulados com redes neurais rasas contendo no máximo uma ou duas camadas. Este autor relata que arquiteturas mais profundas costumam ser criadas "empilhando" modelos mais simples de maneira criativa. Dentre as arquiteturas mais utilizadas, destacam-se:

- Redes de função de base radial (RBF) que são redes não profundas e normalmente usam apenas duas camadas. Não são comumente usadas atualmente, embora tenham um potencial significativo para tipos específicos de problemas complexos, como a predição de tráfego em redes de computadores.

- Máquinas de Boltzmann restritas (RBMs) usam a noção de minimização de energia para criar arquiteturas de rede neural para modelar dados de uma forma não supervisionada. Teve sua origem ao uso de redes de Hopfield, que podem ser usadas para armazenar memórias. Utilizam modelos probabilísticos e seu ponto forte é a redução de dimensionalidade. Atualmente são utilizadas para extrair características com objetivo de alimentar outro algoritmo supervisionado.

- As redes neurais recorrentes (RNN) são projetadas para dados sequenciais, como

frases de texto, séries temporais e outras sequências discretas, como sequências biológicas. Produzem modelos dinâmicos, ou seja, mudam ao longo do tempo de forma a produzir classificações precisas dependentes do contexto dos exemplos que estão expostos.

- Redes neurais convolucionais (CNN) são redes inspiradas biologicamente, são comumente utilizadas em classificações de imagens e detecção de objetos. A primeira arquitetura básica baseada nesta inspiração biológica foi o Neocognitron, criada por Fukushima em 1979, mas vem ganhando destaque em pesquisas para detecção de câncer no diagnóstico por imagens.

Essas redes também são consideradas redes-base para a construção de novos modelos e outras redes neurais artificiais, normalmente para aplicações específicas, tais como: Rede deconvolucional, que parte de uma convolucional invertida e é usada para geração de novas imagens obtidas a partir de convoluções. Rede de estado de eco, que é uma rede recorrente, mas com conexões aleatórias entre as camadas, usadas em microchips óticos. Maiores detalhes podem ser consultados em Aggarwal (2018) e Ertel (2017).

A próxima seção fornece uma introdução, conceitos e ideias principais que envolvem redes neurais.

2.4.1 Redes Neurais

As Redes Neurais Artificiais (RNAs), fazem parte de uma área de estudo conhecida como Inteligência Artificial, embora também estejam presentes em outras áreas como Computação Evolucionária, Metaheurísticas, *machine learning*, entre outras. Tais redes trazem soluções que podem ser aplicadas aos mais diversos propósitos, são baseadas no funcionamento dos neurônios biológicos e possui uma grande capacidade de auto-incremento ou de aprendizado conseguindo atingir resultados que em uma solução convencional seria mais demorado e difícil.

Devido ao caráter multidisciplinar, está presente em diversas áreas de atuação, como por exemplo na avaliação de imagens para exames médicos (SOUZA; SOUZA, 2004), reconhecimento facial (CHAVES, 2018), controle de tráfego (CASTRO, 2017), astronomia (MELLO, 2014), análise gráfica do mercado financeiro (THOMAZ; VELLASCO, 2016) e classificação de padrões textuais e de fala (DUTRA, 2011).

2.4.1.1 Modelo Matemático

As redes neurais consistem em um modelo que tem inspiração no cérebro, devido a sua habilidade de adquirir e armazenar conhecimento necessário para realização de tarefas. Por conta desta motivação biológica, os elementos de processamento de uma rede neural

são denominados neurônios ou nós, sendo uma unidade de processamento de informações fundamental para o funcionamento da rede.

Podemos identificar três elementos básicos na composição de uma rede:

- Um conjunto de pesos sinápticos (ou conexões sinápticas) e uma operação binária que combina a entrada com a respectiva conexão sináptica.
- Uma regra de agregação que combina as entradas dos neurônios ponderados com as respectivas conexões sinápticas.
- Uma função de ativação com o objetivo de introduzir não linearidade no modelo ou confinar a saída do neurônio num determinado intervalo.

A Figura 3 exemplifica o modelo de um neurônio apresentado por Haykin (2001), onde $X_1, X_2, ..., X_m$ são os sinais de entrada conectados ao neurônio k que são multiplicados pelos pesos sinápticos, $W_{k1}, W_{k2}, ..., W_{km}$, incluindo uma variável polarização (bias) b_k que é adicionada ao somatório da função de ativação φ, com o intuito de aumentar o grau de liberdade desta função e, consequentemente, a capacidade de aproximação da rede.

Matematicamente, um neurônio k é escrito como na Equação 2.3.

$$y_k = \varphi(u_k + b_k) \tag{2.3}$$

onde,

$$u_k = \sum_{j=1}^{m} W_{kj} X_j \tag{2.4}$$

sendo, $X_1, X_2, ..., X_m$ sinais de entrada, $W_{k1}, W_{k2}, ..., W_{km}$ são os pesos sinápticos do neurônio k, u_k é a saída do combinador linear devido aos sinais de entrada, b_k é o bias, $\varphi(.)$ é a função de ativação e y_k são os sinais de saída do neurônio.

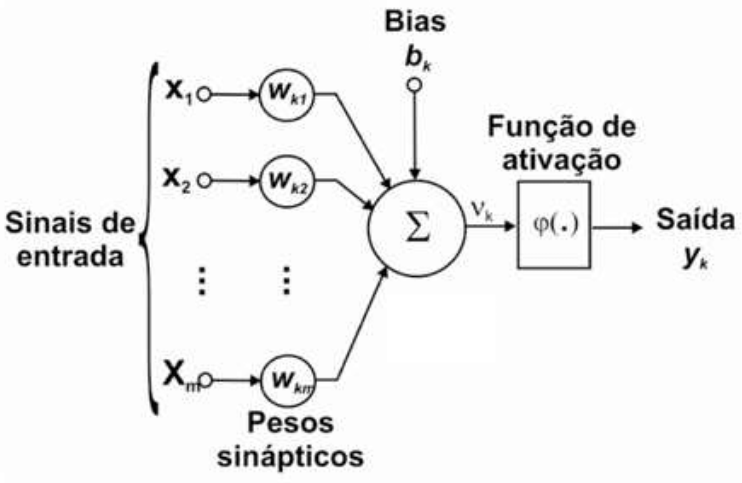
Para função de ativação a literatura sugere vários tipos. Apresentamos aqui algumas delas. A mais simples é a função identidade $\varphi(x) = x$, neste caso o neurônio calcula apenas a soma ponderada dos valores de entrada, no entanto, isso pode causar problemas de convergência com a dinâmica neural porque esta função é ilimitada.

Uma função muito utilizada é a função limiar (*threshold function*), definida pela Equação 2.5.

$$\varphi(x) = \begin{cases} 1 & se \quad x \geq 0 \\ 0 & se \quad x < 0 \end{cases} \tag{2.5}$$

A saída do neurônio k empregando esta função fica como na Equação 2.6

Figura 3 – Modelo de um Neurônio Artificial



Fonte: Extraído de Haykin (2001)

$$y_k = \begin{cases} 1 \quad se \quad \varphi(\sum_{j=1}^{m} W_{kj} X_j + b_k) \geq 0 \\ 0 \qquad\quad caso \quad contrário \end{cases} \tag{2.6}$$

A função definida em 2.5 é bastante sensível para neurônios binários porque a ativação de um neurônio só pode assumir os valores zero ou um de qualquer maneira. Em contraste, para neurônios contínuos com ativações entre 0 e 1, a função degrau cria uma descontinuidade que pode ser suavizada utilizando-se a função denominada Sigmóide (ou Logística), definida em 2.7 onde β é um parâmetro de inclinação.

$$\varphi(x) = \frac{1}{1 + e^{-\beta x}} \tag{2.7}$$

A Figura 4 mostra algumas das principais funções de ativação encontradas na literatura. As funções Tangente hiperbólica ($Tanh$), Hard-Tangente Hiperbólica ($HTanh$) e ReLU (*Retified Linear Unit-ReLU*) são dadas respectivamente por 2.8, 2.9 e 2.10.

$$Tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{2.8}$$

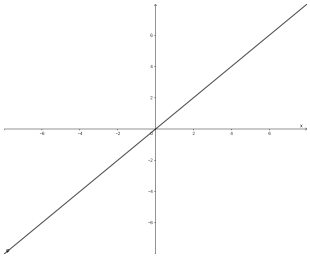
$$HTanh = \begin{cases} 1 \quad se \quad x > 1 \\ -1 \quad se \quad x < -1 \\ 0 \qquad -1 \leq x \leq 1 \end{cases} \tag{2.9}$$

$$ReLU(x) = \begin{cases} x \quad se \quad x \geq 0 \\ 0 \qquad x < 0 \end{cases}, ReLU(x) = Máx(0, x) \tag{2.10}$$

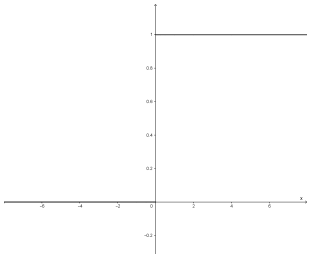
Segundo Ertel (2017), a modelagem de aprendizagem é central para a teoria das redes neurais. Uma das possibilidades de aprendizado consiste em fortalecer uma sinapse

Figura 4 – Algumas funções de ativação

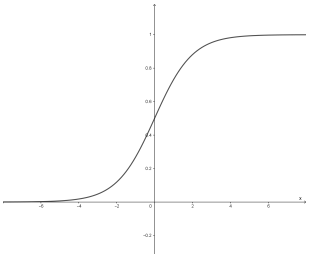
(a) Identidade



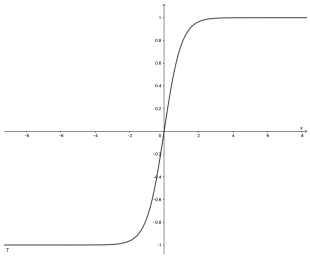
(b) Limiar



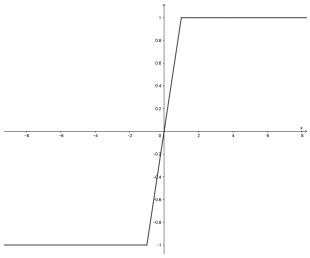
(c) Sigmóide (Logística)



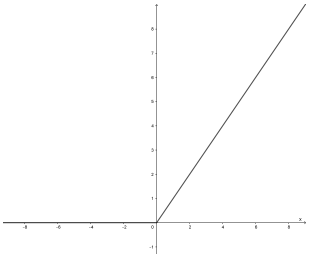
(d) Tangente Hiperbólica



(e) Hard-Tangente Hiperbólica



(f) ReLU



Fonte: Extraído de Ertel (2017)

de acordo com quantos impulsos elétricos ela deve transmitir. Este princípio foi postulado por D. Hebb em 1949 e é conhecido como regra de Hebb.

> Se existir uma conexão W_{ij} entre o neurônio j e o neurônio i e sinais repetidos forem enviados do neurônio j para o neurônio i, o que resulta em ambos os neurônios estarem simultaneamente ativos, então o peso W_{ij} é reforçado. Uma possível fórmula para a mudança de peso ΔW_{ij} é
>
> $$\Delta W_{ij} = \eta x_i x_j$$
>
> com constante η (taxa de aprendizagem), que determina o tamanho das etapas individuais de aprendizagem (ERTEL, 2017, p. 249).

Existem muitas modificações desta regra, que resultam em diferentes tipos de redes ou algoritmos de aprendizagem.

Depois de montado o modelo, é iniciado o processo de aprendizagem da rede, inicialmente os parâmetros são livres e adaptados através do processo de estimulação pelo contexto o qual a rede está. Conforme já explicado por Haykin (2001), trata-se da seguinte sequência:

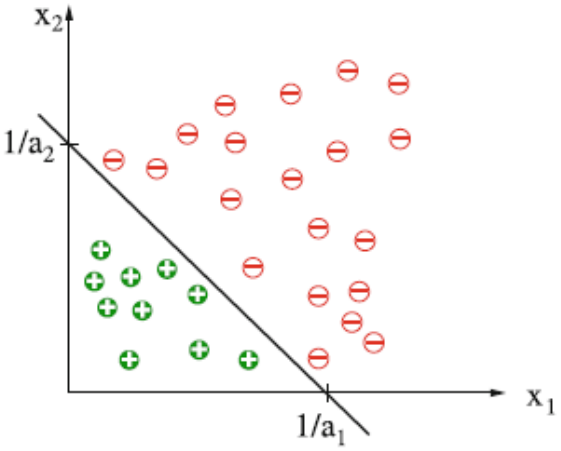
- A rede neural é estimulada por um ambiente.
- A rede sofre modificações nos seus parâmetros por conta da estimulação.
- A rede responde de forma diferente por conta da estimulação sempre diferenciada.

O processo de aprendizado pode ser supervisionado, quando alguém ou algo fornece à rede a saída desejada conforme um padrão de entrada, ou não supervisionado, quando ela mesma toma as decisões baseadas nos parâmetros. O mais comum é o processo supervisionado, visando a integridade e agilidade nesta fase da implementação (ERTEL, 2017).

2.4.1.2 Rede Perceptron

O algoritmo *Perceptron* de Rosenblatt foi visto como uma pedra angular fundamental das redes neurais, causando entusiasmo sobre as perspectivas da inteligência artificial. Foi construído com o foco de solucionar classificações binárias e é uma rede considerada linear, de uma camada apenas. Como ilustração, considere um conjunto de dados bidimensional como apresentado na Figura 5. A equação que separa o conjunto de dados é dada por $a_1x_1 + a_2x_2 = 1$.

Figura 5 – Conjunto linearmente separável



Fonte: Extraído de Ertel (2017)

Se considerarmos o espaço n-dimensional, a separação é dada por um hiperplano[4] cuja equação é dada por (2.11).

$$\sum_{i=1}^{n} a_i x_i = b \tag{2.11}$$

A Definição 1 traz o conceito de conjuntos linearmente separáveis.

Definição 1 *Dois conjuntos $S_1 \subset \mathbb{R}^n$ e $S_2 \subset \mathbb{R}^n$ são chamados linearmente separáveis se existirem números reais $a_1, a_2, ..., a_n, b$ tais que*

$$\sum_{i=1}^{n} a_i x_i > b, \quad \forall x \in S_1 \quad e \quad \sum_{i=1}^{n} a_i x_i \leq b, \quad \forall x \in S_2$$

[4] Um hiperplano do $\mathbb{R}^n$ é o conjunto $H = \{(x_1, x_2, ..., x_n) \in \mathbb{R}^n | a_1x_1 + a_2x_2 + ... + a_nx_n = b \ ; \ a_1, ..., a_n \in \mathbb{R}\}$

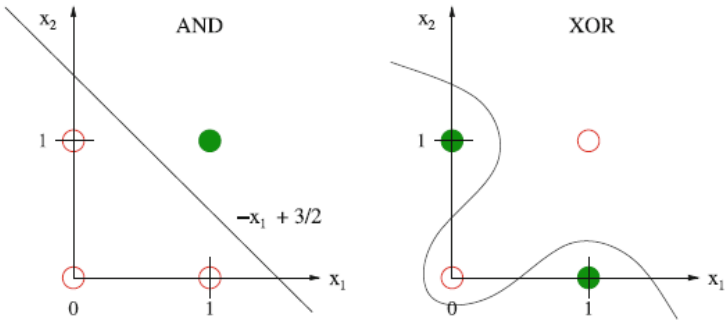
O valor b é denominado limiar (threshold).

Por vezes, o somatório $\sum_{i=1}^{n} a_i x_i$ será escrito como um produto interno[5] de dois vetores $a = a_1, a_2, ..., a_n$ e $x = x_1, x_2, ..., x_n$, ou seja,

$$\sum_{i=1}^{n} a_i x_i = a^t \dot{x}$$

Dois exemplos interessantes de conjuntos linearmente separáveis e não linearmente separáveis são as funções AND e XOR respectivamente, exemplificadas na Figura 6. A função XOR tem uma estrutura mais complexa do que a função AND. O algoritmo *Perceptron* é destinado a conjuntos linearmente separáveis.

Figura 6 – Funções Booleanas AND e XOR



Fonte: Extraído de Ertel (2017)

A Definição 2 define um *Perceptron* e o Teorema 2.4.1 garante a convergência do *Perceptron* para dados linearmente separáveis. O código para o caso binário é descrito no Algoritmo 1. Detalhes em (ERTEL, 2017).

Definição 2 *Seja* $w = (w_1, w_2, ..., w_n) \in \mathbb{R}^n$ *um vetor de pesos e* $x \in \mathbb{R}^n$ *um vetor de entrada. Um Perceptron representa uma função* $P : \mathbb{R}^n \to \{0, 1\}$ *que corresponde a seguinte regra:*

$$P(x) = \begin{cases} 1, & se \quad x = \sum_{i=1}^{n} w_i x_i > 0 \\ 0, & caso\ contrário \end{cases}$$

Teorema 2.4.1 *Sejam as classes* S_1 *e* S_2 *linearmente separáveis por um hiperplano* $wx = 0$. *Então o algoritmo Perceptron converge para qualquer inicialização do vetor de pesos* w. *O Perceptron* P *(Definição 2) com vetor de peso* w *divide as classes* S_1 *e* S_2, *ou seja:*

$$P(x) = 1 \Leftrightarrow x \in S_1$$

$$P(x) = 0 \Leftrightarrow x \in S_2$$

[5] Para conceito de produto interno consultar Leon (1999)

Entrada: Inicializar o vetor de pesos w e o bias b;
Inicie a taxa de aprendizado η $(0 < \eta \leq 1)$;
while *Critério de Parada não satisfeito* **do**
  for *Para cada par de dados de treinamento* (x_i, d_i) **do**
    Execute ;
    $y^* = b + \sum_{i=1}^{n} w_i x_i$;
    if $y^* > 0$ **then**
      $y = 1$
    else
      $y = 0$
    end
    Atualize os pesos e a tendência ;
    if $y \neq d$ **then**
      $w_i(novo) = w_i(atual) + \eta d x_i$;
      $b(novo) = b(atual) + \eta d$;
    else
      $w_i(novo) = w_i(atual)$;
      $b(novo) = b(atual)$;
    end
  end
end
Teste a condição de parada.

Algoritmo 1: Algoritmo *Perceptron*

Se um conjunto de dados de treinamento d é não separável, então não existirá um vetor de pesos w que classifique corretamente todos os exemplos de treinamento em d utilizando o algoritmo de aprendizagem do *Perceptron*. Nas seções seguinte falaremos sobre outras redes que são significativamente mais poderosas, porém como afirma Ertel (2017), vale salientar que a fórmula do *Perceptron* é equivalente ao Método do *Score* e também ao método mais simples de *Naive Bayes* ingênuo, o tipo mais simples de rede *Bayesiana* indicando que vários algoritmos de classificação muito diferentes têm uma origem comum.

Na seção seguinte apresenta-se uma generalização do *Perceptron* na forma de algoritmo de retropropagação, que permite dividir conjuntos não linearmente separáveis através do uso de múltiplas camadas e que possui uma melhor regra de aprendizagem.

2.4.1.3 Rede neural multicamadas

Foi visto na seção anterior que o algoritmo *Perceptron* é uma rede considerada linear, de uma camada apenas, o que acaba limitando as operações que é capaz de suportar. Para melhorar seu dinamismo, foi criado a *Multilayer Perceptron* (MLP), uma rede *Perceptron* com várias camadas, possibilitando assim operações como o operador XOR (HAYKIN, 2001).

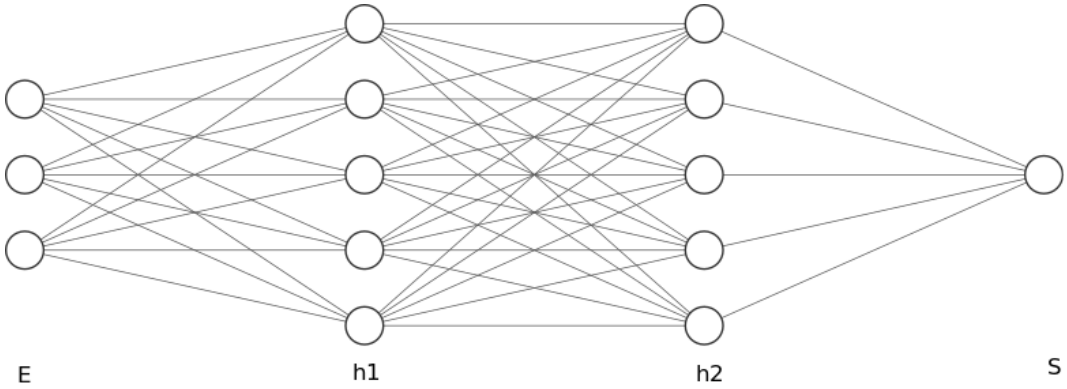
As redes neurais multicamadas possuem várias camadas computacionais, denomina-

das camadas intermediárias adicionais (entre entrada e saída) normalmente são camadas ocultas porque os cálculos realizados não são visíveis para o usuário (AGGARWAL, 2018).

Segundo Menotti (2018) essas redes são frequentemente aplicadas a problemas de aprendizagem supervisionados, treinam em um conjunto de pares entrada-saída aprendendo a modelar a correlação (ou dependências) entre essas entrada-saídas. Esse treinamento envolve os ajustes dos parâmetros do modelo para minimizar o erro.

Para compreender a arquitetura do processamento da informação em uma rede neural, é possível expressar graficamente conforme o modelo da Figura 7 sendo lidos da esquerda para a direita. Onde E são as entradas, h1 é a primeira camada oculta, h2 é a segunda camada oculta e S é a saída resultante. O modelo apresentado possui todas as conexões, ou seja, cada neurônio, em cada camada, está conectado aos outros neurônios da camada anterior.

Figura 7 – Rede MLP com duas camadas



Fonte: Adaptada de Ghoshal (2020)

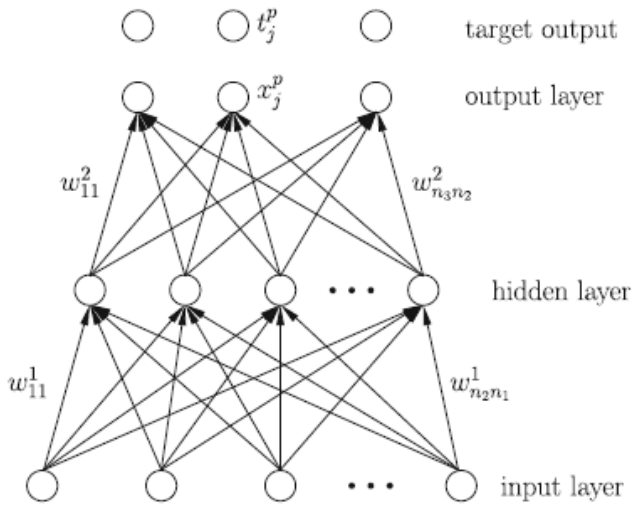
Em uma rede neural de camada única, o processo de treinamento é relativamente simples porque o erro (ou função de perda) pode ser calculado como uma função direta dos pesos. No caso de redes multicamadas, a perda é uma função de composição compilada dos pesos nas camadas anteriores. O gradiente de uma função de composição é calculado usando o algoritmo de retropropagação do erro, o *Backprogation* (AGGARWAL, 2018). O algoritmo contém duas fases principais, referidas como fases para a frente (propagação para frente) e para trás (propagação para trás).

A Figura 8 exibe uma rede de retropropagação típica com uma camada de entrada, uma camada oculta e uma camada de saída. Todos os neurônios de entrada x_j são calculados pela Regra 2.12.

$$x_j = \varphi\left(\sum_{i=1}^{n} w_{ij} x_i\right) \tag{2.12}$$

onde n é o número de neurônios na camada anterior e tomando a função φ como a sigmóide. Analogamente à regra delta incremental, os pesos são alterados proporcionalmente

Figura 8 – Rede de retropropagação com três camadas, com n_1 neurônios na primeira, n_2 neurônios na segunda e n_3 na terceira



Fonte: Extraído de Ertel (2017)

ao gradiente negativo da função de erro quadrático somado sobre os neurônios de saída, dados pela expressão 2.13.

$$E_p(w) = \frac{1}{2} \sum_{k \in output} (t_k^p - x_k^p)^2 \tag{2.13}$$

E para o padrão de treinamento p, vem

$$\Delta_p w_{ij} = -\eta \frac{\partial E_p}{\partial w_{ji}} \tag{2.14}$$

Para escrever a regra de aprendizagem, substitui-se na expressão 2.14 E_p e x_k (definida na Equação 2.12). Dentro da equação, as saídas x_i dos neurônios da próxima camada mais profunda ocorrem recursivamente, etc. Aplicando a regra da cadeia, obtemos a regra de aprendizagem *backpropagation* 2.15.

$$\Delta_p w_{ij} = \eta \delta_j^p x_i^p, \tag{2.15}$$

onde,

$$\delta_j^p = \begin{cases} x_j^p \left(1 - x_j^p\right) \left(t_j^p - x_j^p\right) & \text{se } j \text{ é um neurônio de saída} \\ x_j^p \left(1 - x_j^p\right) \sum_k \delta_k^p w_{kj} & \text{se } j \text{ é um neurônio escondido} \end{cases}$$

A execução do processo de aprendizagem é mostrado no Algoritmo 2. Após calcular a saída da rede (propagação direta) para um exemplo de treinamento, o erro de aproximação é calculado, que é então usado durante a propagação para trás para alterar os pesos de uma camada para outra. Todo o processo é aplicado a todos os exemplos de treinamento e repetido até que os pesos não mudem mais ou um limite de tempo seja atingido.

Entrada: Inicializar o vetor de pesos w_j com valores aleatórios;
repeat
  for (q^p, t^p) **do**
    1. Aplique o vetor de consulta q^p à camada de entrada;
    2. Propagação para frente:
      Para todas as camadas, da primeira camada oculta para cima
        Para cada neurônio da camada
          Calcule a ativação $x_j = f(\sum_{i=1}^{n} w_{ji}x_i)$;
    3. Calcule o erro quadrado $E_p(w)$;
    4. Propagação para trás:
      Para todos os níveis de pesos, do último para baixo
        Para cada peso $w_(ji)$
          $w(ji) = w(ji) + \eta\delta_j^p x_i^p$;
  end
until *w convergir ou atingir o tempo limite*;

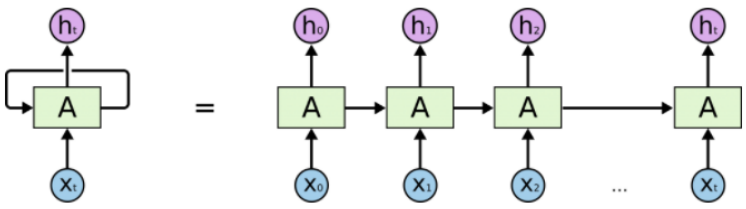
Algoritmo 2: Algoritmo *Backprogation*

A principal vantagem do MLP é a sua capacidade de aprender e quantificar modelos não lineares. Mas, também existem desvantagens, o MLP com camadas ocultas tem uma função de perda não convexa onde existe mais de um mínimo local e, portanto, diferentes inicializações de peso aleatório podem levar a uma precisão de validação diferente. Além disso, o MLP requer o ajuste de vários parâmetros.

2.4.1.4 Rede neural recorrente

As redes neurais recorrentes (*Recurrent Neural Network / RNN*) são um tipo de rede neural artificial que usa dados sequenciais ou dados de série temporal. Esse algoritmos de aprendizado profundo são comumente usados para problemas ordinais ou temporais, como na tradução de linguagem e processamento de linguagem natural. As redes neurais recorrentes utilizam dados de treinamento para aprender, possuem uma espécie de memória, pois recebem informações de entradas anteriores para influenciar as entradas e saídas atuais. As redes neurais recorrentes alavancam o algoritmo de retropropagação através do tempo para determinar os gradientes, que é ligeiramente diferente da retropropagação tradicional, pois é específico para dados de sequência (SKANSI, 2018).

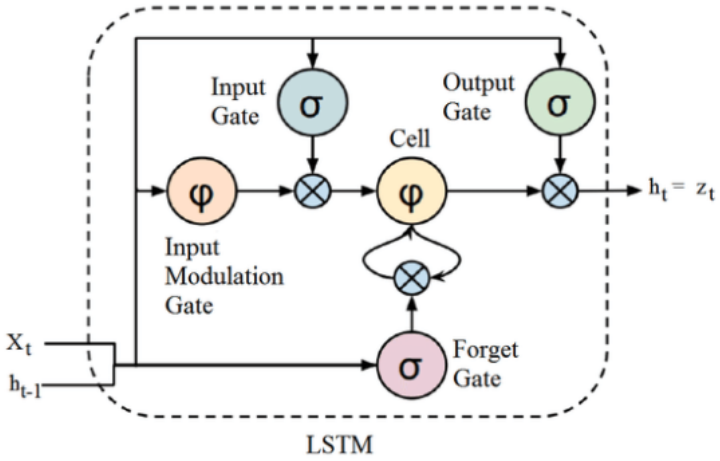
Figura 9 – Representação RNN (SCIENCE, 2021)



Fonte: Extraído de Science (2021)

A adição da recorrência neste modelo de rede, cria um efeito temporal, permitindo que certas conexões da rede possam ser utilizadas em momentos de tempo diferente. Uma rede neural recorrente pode ser interpretada como múltiplas cópias da mesma rede, cada uma passando uma mensagem a um sucessor, como exemplifica a Figura 9. Nos últimos anos, tem havido um incrível sucesso ao aplicar as RNNs a uma variedade de problemas: reconhecimento de fala, modelagem de idiomas, tradução, classificação de texto, etc (KARPATHY, 2015). Este sucesso se deve a uma de suas variações, as *Long Short Term Memory (LSTM)*, que é uma arquitetura de rede neural recorrente (RNN) que "lembra" valores em intervalos arbitrários e é exemplificada na Figura 10, observe que ela possui uma estrutura em cadeia que contém quatro redes e diferentes blocos de memória chamados células, a informação é retida pelas células e as manipulações de memória são feitas pelos portões ou *gates*.

Figura 10 – Representação cadeia LSTM



Fonte: Extraído de Science (2021)

As informações que não são mais úteis no estado da célula são removidas com o *forget gate*. Duas entradas: x_t (entrada no momento específico) e h_{t-1} (saída de célula anterior) são alimentadas ao *gate* e multiplicadas por matrizes de peso, seguidas pela adição do *bias*. O resultante é passado por uma função de ativação que fornece uma saída binária. Se para um determinado estado de célula a saída for 0, a informação é esquecida e para a saída 1, a informação é retida para uso futuro.

A adição de informações úteis ao estado da célula é feita pelo *input gate*. Primeiro, a informação é regulada usando a função sigmoide que filtra os valores a serem lembrados de forma similar ao *forget gate* usando as entradas h_{t-1} e x_t. Então, um vetor é criado usando a função *Tanh* que dá saída de -1 a +1, que contém todos os valores possíveis de h_{t-1} e x_t. Os valores do vetor e os valores regulados são multiplicados para obter as informações úteis.

A tarefa de extrair informações úteis do estado da célula atual para serem apresentadas como uma saída é feita pelo *output gate*. Primeiro, um vetor é gerado aplicando a função *Tanh* na célula. Então, a informação é regulada usando a função sigmoide que filtra os valores a serem lembrados usando as entradas h_{t-1} e x_t. Os valores do vetor e os valores regulados são multiplicados para serem enviados como uma saída e entrada para a próxima célula.

2.5 Análise de desempenho dos classificadores

Após a escolha de um algoritmo de classificação, é usual fazer uma análise do desempenho do classificador, ou seja, identificar com que frequência as instâncias de uma classe C_k foram classificadas corretamente como classe C_k ou classificadas incorretamente como alguma outra classe. Essas informações são mostradas em uma matriz denominada Matriz de Confusão, exemplificada na Tabela 3. Para um classificador perfeito, FP e FN deveriam ser nulos.

Tabela 3 – Matriz de Confusão para Categoria C_k

		Classe Predita	
		C_k	Não C_k
Classe Atual	C_k	*VP*	*FN*
	Não C_k	*FP*	*VN*

Fonte: Extraído de Bramer (2016)

Na Tabela 3 temos que, verdadeiro positivo(VP) indica a quantidade de registros que foram classificados como positivos corretamente, ou seja, a resposta do classificador foi que o comentário era positivo e o comentário realmente era positivo. Falso negativo(FN) indica a quantidade de registros que foram classificados como comentários negativos de maneira incorreta, ou seja, a resposta do classificador foi que o comentário era negativo, mas o comentário era positivo. Falso positivo(FP) indica a quantidade de registros que foram classificados como comentários positivos de maneira incorreta, ou seja, a resposta do classificador foi que o comentário era positivo, mas o comentário era negativo. Verdadeiro negativo(VN) indica a quantidade de registros que foram classificados como negativos de maneira correta, ou seja, a resposta do classificador foi que o comentário era negativo e o comentário realmente era negativo.

Mas para um entendimento completo é necessário agrupar essa informação em cálculos que exibam a informação de maneira compilada e objetiva, ou seja, um número que expresse a efetividade conforme determinada ótica. Assim, as métricas mais utilizadas são:

- **Acurácia:** É uma das métricas mais utilizadas, pois mede a frequência com que o classificador efetuou a previsão correta. É o indicador mais simples de se calcular,

dado pela divisão entre todos os acertos pelo total. É definida pela Equação (2.16).

$$Acurácia = \frac{VP + VN}{VP + VN + FN + FP} \tag{2.16}$$

- **Precisão:** A precisão representa a exatidão do método e é calculada como a proporção de instâncias que foram previstas como positivas e foram de fato positivas divididas pelo número total de instâncias que foram previstas como positivas. É definida pela Equação (2.17).

$$Precisão = \frac{VP}{VP + FP} \tag{2.17}$$

- ***Recall:*** É utilizada para indicar a relação entre as previsões positivas realizadas corretamente e todas as previsões que realmente são positivas. Essa métrica é capaz de responder a questão: De todos os comentários que realmente são positivos, qual percentual é identificado corretamente pelo modelo. É definida pela Equação (2.18).

$$Recall = \frac{VP}{VP + FN} \tag{2.18}$$

- ***F-measure (F1_Score):*** Muitas vezes, calcular o recall e a precisão ainda não é suficiente. Uma combinação dos dois é mais apropriada para avaliar o desempenho dos métodos. O *F-measure* também conhecida como *F1_score*, é uma média harmônica entre *recall* e precisão. Um valor baixo para *F1_score* indica que ou a precisão ou o *recall* está baixo. Esta métrica é calculada conforme a Equação (2.19).

$$F - Measure = 2.Precisão.\frac{Recall}{Precisão + Recall} \tag{2.19}$$

Que é equivalente à equação

$$F - Measure = \frac{2.VP}{(VP + FN) + (VP + FP)} \tag{2.20}$$

2.6 Treinamento, validação e testes

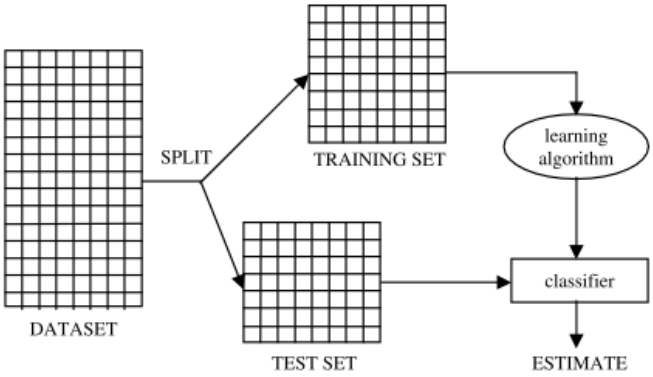
Ao construir um modelo estatístico ou de rede neural, é necessário organizar os dados que serão explorados, de forma que atenda à expectativa do propósito do modelo e também seja eficiente no seu processamento. Igual (2017) descreve como conjuntos de treinamento, validação e teste.

Os dados de teste são usados exclusivamente para avaliar o desempenho no final do processo e não devem ser usados no processo de aprendizagem. Os dados de validação são usados para selecionar os parâmetros com o melhor desempenho de acordo com uma estimativa do erro de generalização. Já os dados de treinamento são usados para ensinar a instância do modelo.

É importante compreender que não existe apenas uma forma de fazer o processo de organização dos conjuntos de dados, sendo os mais utilizados descritos a seguir.

- *Hold-out validation* Esta é conhecida como a forma mais simples e difundida nos modelos de predição. Define-se um percentual para cada conjunto de dados de treino e de teste, e a partir dessa divisão criam-se as amostras. Normalmente usa-se este método quando o conjunto de dados é maior e é possível comprovar que as amostras criadas têm significância estatística para representar a população. Bramer (2016) descreve este método como "treinar e testar", seu fluxo pode ser verificado na Figura 11.

Figura 11 – Treino e teste *Hold-out validation*



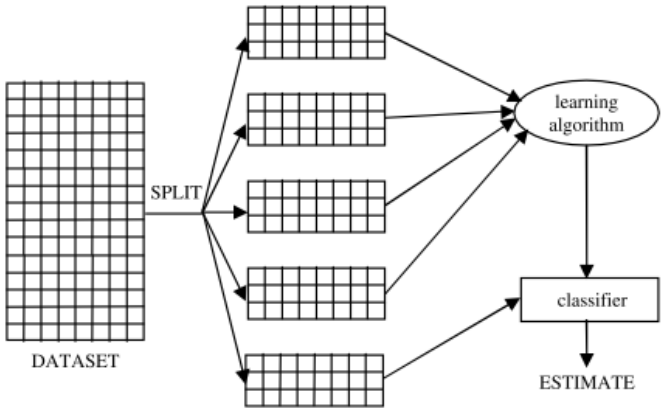
Fonte: Extraído de Bramer (2016)

- *Cross validation* Apesar de existir mais de um *cross validation*, o mais comum é utilizar o *k-fold cross validation*. Neste método, demonstrado na Figura 12, criamos um número k de amostras, sendo que cada amostra é deixada de lado enquanto o modelo treina com o restante delas. O processo se repete até que seja possível determinar a qualidade de cada observação, chegando também em uma média geral. Os valores comuns para o número k de amostras são entre 5 e 10. O *k-fold cross validation* geralmente é usado quando não se tem dados suficientes para seguir com a estratégia mais simples de *hold-out validation*.

- *Bootstrapping:* Se baseia em uma amostragem aleatória com substituição. Este método pode ajudar em muitas situações, como na validação do desempenho preditivo de um modelo, métodos de *ensemble*[6], estimativa de viés e variação do parâmetro de um modelo, etc. Selecionando amostragem com substituição do conjunto de dados original e assumindo ao mesmo tempo que os pontos de dados que não foram escolhidos são o conjunto de dados de teste. Pode-se repetir esse procedimento várias vezes para calcular a pontuação média como estimativa do desempenho do modelo. Além disso,

[6] Técnica de aprendizado de máquina que combina o resultado de múltiplos modelos em busca de produzir um melhor modelo preditivo.

Figura 12 – Treino e teste *Cross validation*- De Bramer (2016)



Fonte: Extraído de Bramer (2016)

o *bootstrapping* está relacionado aos métodos de treinamento do conjunto, porque podemos construir um modelo usando cada conjunto de dados de *bootstrap* e agrupar esses modelos em um conjunto usando a votação majoritária (para classificação) ou calculando a média (para previsões numéricas) para todos os esses modelos como resultado final. O *bootstrapping* não requer cálculos teóricos e está disponível não importa o quão matematicamente complicado se trata o conjunto para validação (EFRON; J.TIBSHIRANI, 1993).

2.7 Análise de sentimento

Nessa nova era, denominada era do *Big Data*, evidenciou-se a importância de analisar o grande volume de dados para as empresas que desejam entender melhor o comportamento do seu consumidor. Neste cenário, os profissionais de *Data Science* desenvolveram ferramentas para extrair informações e *insights* a partir destes dados. Um dos métodos para isto é a análise de sentimentos. Esta análise é realizada por meio de processamento de linguagem natural, análise de texto e linguística computacional tendo a finalidade de criar conhecimento a partir destes dados.

A rápida ascensão das redes e mídias sociais proporcionou um crescente interesse pela área de análise de sentimentos. A multiplicação de opiniões, avaliações, recomendações e outras formas de expressão *on-line* transformou o reconhecimento e interpretação de dados em valor para as empresas. Afinal, com essas informações é possível gerenciar reputações e identificar novas oportunidades.

Conforme Aggarwal (2018), o termo análise de sentimento (ou mineração de opinião) refere-se à análise de dados da atitude do sujeito em relação a um determinado tópico. Essa atitude pode ser um julgamento (teoria da avaliação), um estado afetivo ou a comunicação emocional pretendida. Geralmente, a análise de sentimento é realizada com base no

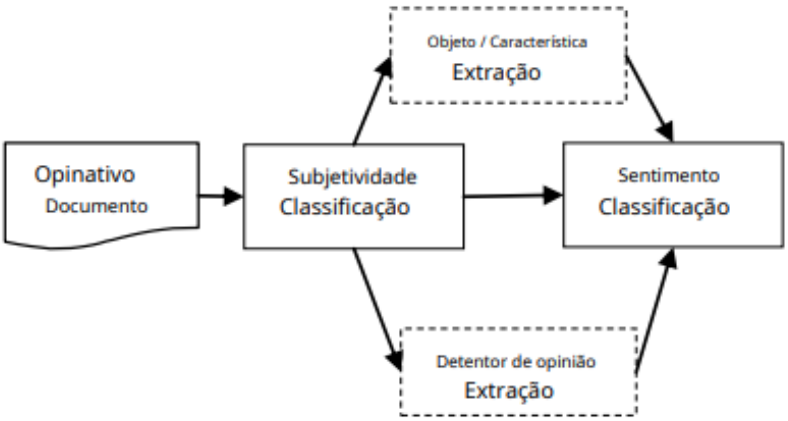
processamento da linguagem natural, na análise do texto e na linguística computacional. Embora o escopo da análise de sentimento possa apresentar muitos aspectos a serem observados, analisaremos problemas de categorização da análise de sentimento binária. Assim, basicamente abordaremos a classificação de opiniões positivas e negativas a partir de dados de texto. O escopo da análise de sentimentos é amplo e inclui muitos aspectos que tornam a análise de sentimentos uma tarefa desafiadora, como por exemplo

- Identificação do sarcasmo: às vezes sem conhecer a personalidade da pessoa, você não sabe se "mau" significa mau ou bom.
- Falta de estrutura de texto: no caso do comentário, por exemplo, pode conter abreviaturas, e pode haver falta de maiúsculas, grafia incorreta, pontuação e gramática inadequadas, dificultando a análise do texto.
- Muitas categorias e graus de sentimento possíveis: positivo e negativo é uma análise simples, mas imagine identificar a quantidade de ódio que há dentro da opinião, quanta felicidade, quanta tristeza, etc.
- Identificação do objeto de análise: muitos conceitos podem aparecer no texto, como detectar o objeto para o qual a opinião é positiva e o objeto para o qual a opinião é negativa é uma questão em aberto. Por exemplo, se você disser "Ela o venceu!", Isso significa um sentimento positivo para ela e um sentimento negativo para ele, ao mesmo tempo.
- Texto subjetivo: outro desafio em aberto é como analisar frases ou parágrafos muito subjetivos. Às vezes, mesmo para os humanos, é muito difícil concordar quanto ao sentimento desses textos altamente subjetivos.

Kumar (2012) apresenta em seu estudo quatro pontos a serem observados na tarefa de classificação de sentimento, conforme ilustra a Figura 13

A classificação da subjetividade onde é identificado o que pode ser considerado opinião e o que não pode ser, ou seja, essa tarefa visa identificar as opiniões dentro de sentenças. Se for possível identificar diretamente esta subjetividade, a próxima etapa é a classificação do sentimento, onde a polaridade é elucidada, seja positivo ou negativo. Caso a subjetividade não identifique a opinião, duas novas etapas podem compor a solução, a primeira é a extração de detentores de opinião, que é o reconhecimento das fontes diretas ou indiretas da opinião. E o segundo é a extração do objeto ou característica, que é a descoberta da entidade de destino. As soluções para vencer estes desafios estão baseadas em estratégias que detectam o sentimento por meio de análise léxicas e aprendizado de máquina, sendo este último o foco do estudo desta pesquisa.

Figura 13 – Análise do sentimento



Fonte: Extraído de Kumar (2012)

2.8 Ferramentas para classificação de textos

Quando se trata de opiniões sobre produtos ou serviços, elas podem ser positivas, negativas ou neutras. Mas julgar cada bloco de texto contendo este comentário é um trabalho repetitivo e demorado. Algumas ferramentas computacionais apresentam soluções combinadas para este fim.

Nas subseções a seguir descrevemos algumas delas. A primeira é formada por um conjunto de bibliotecas disponível na linguagem Python. A segunda ferramenta é o RapidMiner, uma plataforma de ciência de dados, onde é possível modelar o processo de pré-processamento do texto, separando as fases (*tokenização, stop words, steaming,* etc) conforme a necessidade. A terceira ferramenta é o IBM-Watson, um conjunto de serviços cognitivos, elaborado para processar a informação no mesmo formato que a mente humana. No IBM-Watson, o processamento e classificação do texto, utiliza diversos algoritmos combinados.

2.8.1 Bibliotecas Python

A linguagem Python vem se destacando em soluções ligadas à redes neurais, estatística e demais segmentos envolvendo cálculos complexos, e para auxiliar nas soluções, algumas bibliotecas foram desenvolvidas e constantemente atualizadas, visando produtividade e facilidade ao implementar algoritmos tais como os de redes neurais artificiais. Um grande exemplo é a biblioteca *Scikit Learn*, desenvolvida inicialmente no *Google Summer of Code*, hoje é uma das principais ferramentas para análise e classificação de dados e está aberta à comunidade (SCIKIT-LEARN, 2021).

A biblioteca *Scikit Learn* contém o algoritmo *MLPClassifier* com funções para rede neural multicamada para o problema de classificação. Os parâmetros requeridos são:

1. Ativação - Escolha da função de ativação para a camada oculta, que pode ser Identidade, Logística, Tanh e ReLU.

2. Solucionador - É o parâmetro para definir qual método será utilizado para otimização de peso, podendo ser 'lbfgs' que é um otimizador da família de métodos quase Newton, 'sgd' refere-se à descida do gradiente estocástico e 'adam' refere-se a um otimizador baseado em gradiente estocástico proposto por Kingma (2015). O solucionador padrão 'adam' funciona muito bem em conjuntos de dados relativamente grandes (com milhares de amostras de treinamento ou mais) em termos de tempo de treinamento e pontuação de validação. Para pequenos conjuntos de dados, no entanto, 'lbfgs' pode convergir mais rápido e ter um melhor desempenho.

3. Taxa de aprendizado - É a escolha do parâmetro da taxa de aprendizado para atualizações de peso, pode ser 'constante', 'invscaling' ou 'adaptativo', conforme documentação em Scikit-Learn (2021).

4. Taxa de aprendizado inicial - Controla o tamanho da etapa na atualização dos pesos, normalmente utilizada como padrão o valor 0,001.

5. Tolerância - É a taxa de tolerância aceitável utilizada dentro de outros parâmetros.

6. Número máximo de iterações - O solucionador itera até a convergência (determinada por 'tol') ou até esse número de iterações escolhido.

O *MLPClassifier* treina iterativamente, já que a cada etapa de tempo as derivadas parciais da função de perda em relação aos parâmetros do modelo são calculadas para atualizar os parâmetros. Também pode ter um termo de regularização adicionado à função de perda que encolhe os parâmetros do modelo para evitar o *overfitting*, ou sobre-ajuste.

Já para os algoritmos de redes recorrentes podemos falar da classe *LSTM - Long Short-Term Memory*, essa classe utilizada um laço para iterar ao longo dos passos de tempo de uma sequência, enquanto mantém um estado interno que codifica informações sobre os passos de tempo que viu até agora, ou seja da informação já conhecida, este processo pode ser melhor compreendido na Seção 2.4.1.4.

Alguns parâmetros da classe LSTM podem ser customizados, tais como o número de neurônios utilizados, definidos por "hidden_nodes", as dimensões dos recursos utilizados, definidos por "input_dim"ou ainda o "dropout_value"que desconsidera aleatoriamente uma quantidade de conexões para conter o sobreajuste.

2.8.2 Rapidminer

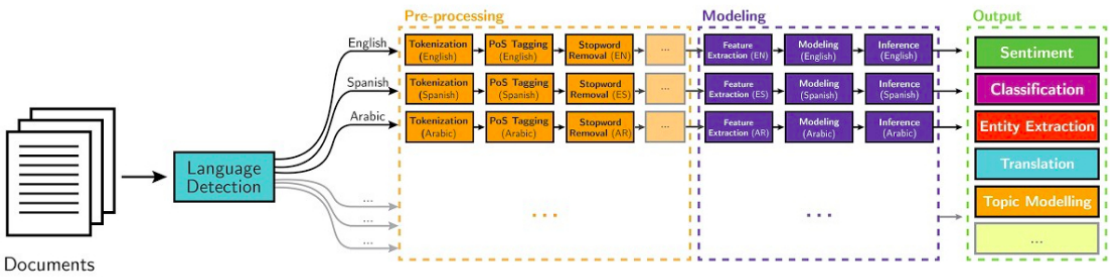
Desenvolvido no centro de Inteligência Artificial da Universidade de Dortmund em 2001, atualmente é mantido pela Rapid-I que mantém versões beta para uso acadêmico

e proprietárias do software. O conceito básico do Rapidminer é o chamado "árvore de operadores", na qual algoritmos podem ser inseridos em forma de cadeia. Inclui uma biblioteca de algoritmos extensa, podendo inserir mais algoritmos através de download de plugins. É compatível com a importação e uso de arquivo nos formatos .xls, arff, bibtex, csv, mdb entre outros. O pacote para processamento de textos possui algoritmos para: Inserção de dados, Filtros ("stopwords" para língua germânica e inglesa e algoritmo para inserir uma lista criada pelo usuário), Tokenizadores, Stemmizadores e Miscelâneos (algoritmos usados para *webmining* e agrupamento). Para todos os algoritmos existe uma visão sobre os parâmetros que podem ser customizados, numa visão de usuário ou de especialista. A interface foi construída no sentido de facilitar a chamada dos procedimentos de aprendizagem de máquina. Mais informações em RAPIDMINER (2014).

2.8.3 Plataforma IBM-Watson

É uma plataforma de programação cognitiva, dividido em diversas APIs, com aplicações específicas. Com essas aplicações é possível entender emoções, interpretar textos, imagens, distinguir sons e fornecer respostas à perguntas. O Watson é o resultado de trabalho de pesquisa e desenvolvimento da IBM, onde seus desenvolvedores pensaram em uma soluções que atendesse as requisições de uma forma próxima ao pensamento humano, daí o nome solução cognitiva. Apesar de mais antigo, ele veio a público por volta de 2011 onde a popularização se iniciou. Um de seus core está no tratamento e validação de dados e de reconhecimento da linguagem natural. O Watson utiliza o software IBM DeepQA e também o framework Apache UIMA (*Unstructure Information Management Architeture*). O sistema foi escrito em diversas linguagens de programação, C++, Java e Prolog, ele roda no sistema operacional SUSE Linux Enterprise Server 11 utilizando o framework Apache Hadoop para a DPC (*Distributed/Parallel Computing*). Esta tecnologia é capaz de processar cerca de 500 giba bytes por segundo, exemplificando, esta capacidade seria aproximadamente um milhão de livros por segundo. Sua capacidade já é utilizada em diversos setores, tais como: medicina, finanças, marketing, etc. Mais informações em (IBM, 2021).

Figura 14 – Fluxo NLP



Fonte: Extraído de IBM (2021)

A NLP *(Natural language processing)* é o campo de interesse neste trabalho, e é ela também um dos pontos fortes do Watson, é uma área da Inteligência Artificial que dá às máquinas a capacidade de ler, entender e extrair significado das linguagens humanas. Conforme a Figura 14, (IBM, 2021), o processo é dividido em fases, desde a identificação do idioma, pré-processamento, aplicação do modelo, para daí sim a saída classificada. Para tal entendimento, existem alguns algoritmos ou técnicas utilizadas, conforme conteúdo da seção 2.2.

3 Revisão de literatura

As empresas e organizações cada vez mais estão sendo imergidas em um mundo tecnológico que disponibiliza grande quantidade de informações e necessitam em algum momento, do uso de tais informações. Para isso precisam de meios e ferramentas para tomarem decisões estratégicas de gestão de modo a se destacarem no mercado, considerando principalmente o aumento da competitividade global. Portanto, é desejável organizar a informação disponível das diversas bases de dados existentes, porém a tarefa se torna cada vez mais difícil devido a quantidade e formato dos dados.

Este trabalho tem interesse particular no problema de extração de informação tomando como base de dados documentos textuais. Para a extração da informação, a ideia é procurar reconhecer padrões utilizando-se de técnicas envolvendo aprendizado de máquina.

Segundo Ghiassi et al. (2012), a maioria dos problemas de classificação na vida real, incluindo classificação de texto, classificação de imagens (genética, médica, empresarial,...) e outros, são de natureza complexa e caracterizada por alta dimensionalidade. As estratégias de solução incluem Naïve Bayes (NB), Rede Neural (NN), Linear Least Squares Fit (LLSF), k-Nearest-Neighbor (kNN) e Support Vector Machines (SVM). São inúmeras técnicas que tem como raiz as pesquisas da década de 60, mas que até o final dos anos 80, utilizando engenharia de conhecimento, eram construídas manualmente. Essas abordagens se baseavam em um conhecimento especializado com um conjunto de regras lógicas, categorias e tópicos para o reconhecimento dos padrões. Em seu artigo, a alternativa apresentada para a classificação de textos é a rede neural artificial dinâmica (DAN2) que pode ser utilizada como aprendizado de máquina com bons resultados, esta técnica também pode ser usada na redução de dimensão, tornando o processamento mais eficiente. Foi desenvolvido um modelo de rede neural, DAN2 (A Dynamic Architecture for Artificial Neural Networks), que emprega uma arquitetura diferente da rede neural tradicional. A filosofia geral do modelo DAN2 é baseada no princípio de aprender e acumular conhecimentos em cada camada propagando e ajustando esse conhecimento para a frente da próxima camada repetindo essas etapas até que os critérios de desempenho de rede desejados sejam atingidos. Como nas redes neurais clássicas, a arquitetura DAN2 é composta por uma camada de entrada, camadas ocultas e uma camada de saída. A camada de entrada aceita dados externos para o modelo. No DAN2, ao contrário redes neurais clássicas, o número de camadas ocultas não é fixado a priori. Eles são gerados sequencial e dinamicamente até que um nível precisão de desempenho é atingido. Além disso, a abordagem proposta usa um número fixo de nós ocultos (quatro) em cada camada. Essa estrutura não é arbitrária, mas justificada pela estimativa aproximação. Em cada camada oculta, a rede é treinada usando

todas as observações no conjunto de treinamento simultaneamente, para minimizar uma medida de precisão de treinamento declarada, como erro médio quadrático (MSE) ou outras medidas de precisão. Em contraste com o método kNN (K-Nearest Neighbors), o DAN2 não requer armazenamento de dados de treinamento. O algoritmo é altamente escalável e composto de uma série de camadas que são geradas automática e dinamicamente, sem exigir nenhuma entrada dos analistas. A escalabilidade do DAN2 é uma força distintiva da abordagem das redes neurais artificiais tradicionais. Para avaliação de desempenho do DAN2 em classificação, o modelo foi aplicado em um conjunto de dados de referência da Reuters (Reuters-21578) que é composto por 21578 documentos, baseados em notícias. O computador usado para o experimento tinha seguinte configuração: Processador Intel Quad Core 3.2Ghz, 16Gb de memória ram, com um conjunto de 4 discos SAS em Raid 1/0. Rodando Linux SuSE Enterprise Server 11, as ferramentas foram executas em uma máquina virtual, dada a compatibilidade.

No experimento, mesmo após o pré-processamento, o conjunto de dados ficou muito grande. A convergência dos algoritmos internos de otimização não linear empregados no DAN2 aprimoraram com dimensionalidade reduzida. Reduzindo o tamanho do recurso a ser processado, pode melhorar muito o desempenho. Foi experimentado vários tamanhos de recursos começando em 1000 e incrementando em 250. Especificamente, foi notada uma queda de 5% na precisão da previsão ao usar mais de 2500 recursos. Houve também a mesma queda de 5% para a métrica de ponto de equilíbrio de precisão/recuperação para o tamanho do recurso definido além da faixa de 2000 a 2250. A região foi explorada incrementando o conjunto de recursos em apenas 50 e encontrou o melhor número de características em torno de 2200. Claramente, não existe uma abordagem estabelecida para selecionar as melhores conjunto de recursos para cada modelo. No entanto, os pesquisadores reconhecem que um pequeno conjunto de recursos mascara a eficácia do classificador e inclui um conjunto muito grande além do nível "crítico"que prejudica o desempenho dos classificadores.

Os resultados desde estudo de Ghiassi et al. (2012), mostram que DAN2 tem um desempenho superior que os outros modelos de classificação comparados, tanto na eficácia quanto no tempo de execução. Sua uniformidade indica robustez na abordagem do tema.

Christen et al. (2015) expõe em seu artigo que: microtextos extraídos de mídias sociais se tornaram uma fonte importante para pesquisa, porém a classificação e interpretação automatizada destes textos são desafiadoras. Em seu estudo ele apresenta um sistema de aprendizado semi-supervisionado para classificação de microtexto que combina técnicas de aprendizado de máquina com a capacidade humana incomparável de tomar decisões exigentes, ou seja, não lineares com base em dados esparsos. Um passo fundamental na mineração de dados sociais é classificação de texto, que pode envolver textos de identificação que correspondem a classes predefinidas, bem como encontrar um esquema

de classificação desconhecido que se encaixe nos dados.

A natureza desses microtextos adiciona uma complicação substancial à classificação do texto, devido à limitada extensão de caracteres, a violação de convenções gramaticais ou estilísticas. As abordagens apresentadas neste trabalho são promissoras, mas geralmente depende de um grande número de textos para permitir um aprendizado de máquina eficiente. Foram investigadas quais abordagens de classificação de texto são mais adequadas para conjuntos de dados médios de microtextos. Comparada uma abordagem de baixo para cima e de cima para baixo, que envolve vários tipos de pré-processamento e enriquecimento de texto, bem como um procedimento de aprendizado semi-supervisionado. Propondo um novo procedimento semi-supervisionado que combina uma etapa de aprendizado não supervisionado para identificar os melhores agrupamentos com uma etapa de controle supervisionado para definir um classificador.

Para este experimento, foram usados microtextos de três concursos de ideias do site Atizo.com. No primeiro, os participantes foram convidados a fornecer ideias sobre como um museu poderia se tornar mais atraente e atrairia mais visitantes (394 textos de 154 solucionadores). No segundo concurso, os participantes foram instruídos à fornecer ideias sobre como um produtor de alimentos de conveniência poderia aumentar sua gama de produtos (314 textos de 129 solucionadores). No terceiro concurso, os participantes foram convidados a dar ideias sobre como uma nova bebida alcoólica poderia ser anunciada (396 textos de 117 solucionadores). Os textos eram principalmente em alemão, alguns eram em inglês ou francês. Foram selecionados 100 microtextos, cada conjunto de dados foi agrupado manualmente por um autor para identificar grupos claros de ideias dentro de cada projeto. Cada texto da ideia consistia em um título, algumas palavras-chave escolhidas pelos solucionadores e o texto que descreve a ideia. Frequentemente, títulos e palavras-chave eram mal descritos e não refletiam o real conteúdo da ideia.

O primeiro objetivo do estudo foi avaliar o efeito do pré-processamento e enriquecimento de texto do resultado de agrupamento, para a técnica de enriquecimento, os textos foram submetidos ao serviço de tradução do Google, algumas palavras foram substituídas pelos substantivos, adjetivos ou verbos, adicionando sinônimos para enriquecer. Após um agrupamento, foi gerado uma matriz-referência, uma matriz de frequência e um fator de normalização, que reduz a importância de ocorrências muito frequentes. O objetivo geral de tais métodos é descobrir a estrutura semântica subjacente de uma coleção de textos encontrando tópicos latentes, nos quais - nas versões estatísticas - um tópico é entendido como uma distribuição por termos e cada texto é associado a diferentes tópicos em diferentes graus.

A ideia básica do algoritmo pode ser resumida da seguinte forma: 1. Realizar uma decomposição de valor singular da matriz de referencia centralizada V, que é equivalente a um PCA e envolve uma decomposição de autovalor da matriz V^tV. 2. Criar uma

representação dimensional inferior dos dados com base nos k maiores valores próprios. 3. Atribuir a cada microtexto o componente principal que possui a maior coordenada absoluta valor (escores) para o vetor correspondente v_d. Isso produz os k conjuntos tópicos básicos. 4. Cada tópico pode ser caracterizado pelo vetor próprio correspondente, que, por sua vez, é caracterizado escolhendo os vetores base (cargas) mais predominantes, ou seja, termos que determinam o vetor próprio. Esse algoritmo básico tem a desvantagem de não permitir uma integração direta de novos dados - nem de novas dimensões (termos) nem de novos itens (microtextos) e a decomposição deve ser recalculada para se adaptar a novos dados. Para cada termo dos 100 textos de um único projeto, é analisada sua frequência relativa em cada grupo de membros do grupo referência. As palavras que são específicas apenas para um grupo, formam o classificador de cada grupo.

Foi descoberto que o efeito do pré-processamento e enriquecimento é bastante variável e fortemente dependente do tipo de texto, ou seja, relacionado ao projeto. Devido a essa variabilidade, apenas alguns tipos de pré-processamento alcançam melhorias significativas. O resultado mostra que a tradução, que foi introduzida como uma alternativa aos sinônimos de enriquecimento de texto, é uma técnica de enriquecimento bastante poderosa. Enriquecer textos por sinônimos geralmente aumenta a semelhança de todos os textos e, portanto, piora o discriminação. O resultado da técnica aplicada, depende do tipo de dados de entrada, sendo assim para alcançar um melhor resultado é necessário a intervenção humana no processo.

Os autores demonstraram que para a classificação de conjuntos de microtextos de tamanho intermediário, uma "sugestão de máquina" e uma intervenção humana que introduz um tipo de correção não linear podem ser a estratégia ideal. Nos casos em que a quantidade de texto é muito pequena para revelar o contexto e uma estatística de texto apropriada, o fator humano permanece importante para resultados práticos.

Hashemi (2020) apresenta em seu artigo uma pesquisa de perspectivas, lacunas e direções futuras das classificações de páginas da Web, que tem seu foco também no uso de pequenos textos, muitas vezes metadados, para entender do que se trata o conteúdo ali publicado. Devido ao recente crescimento impressionante de desempenho e espaço de memória em máquinas de computação, juntamente com a especialização de modelos de aprendizado de máquina para classificação de texto e imagem, muitos pesquisadores começaram a direcionar o problema de classificação de páginas da Web. No entanto, a classificação automática de páginas da Web permanece nos estágios iniciais devido à sua complexidade, diversidade de conteúdo das páginas da Web (imagens de diferentes tamanhos, texto, hiperlinks etc.) e seu custo computacional. O estudo investiga:
(a) Os metadados e as informações contextuais que cercam os termos são geralmente ignorados na classificação do conteúdo textual;
(b) A estrutura e distribuição do texto nas tags HTML e nos hiperlinks são subestudados

na classificação do conteúdo textual;
(c) A eficácia de recursos na distinção entre classes de páginas da Web, a mensuração da contribuição de cada recurso na precisão da classificação é uma lacuna importante na pesquisa;
(d) Os métodos de classificação de imagens dependem fortemente de análises computacionalmente intensivas e específicas de problemas para a extração de recursos;
(e) O aprendizado é pouco estudado, apesar de sua importância na classificação de páginas da Web, devido à enorme quantidade de páginas não rotuladas e ao alto custo de rotulagem.
(f) Aprendizado profundo, redes convolucionais e recorrentes e aprendizado por reforço permanecem pouco explorados, mas intrigantes para a classificação de páginas da Web, e, por último, mas não menos importante.
(g) Desenvolvimento de um teste detalhado junto com a avaliação das métricas e estabelecimento de padrões para os *benchmarks* permanecem uma lacuna na avaliação dos classificadores de páginas da Web.

Habitualmente os métodos utilizados para classificação de páginas são: Baseado em texto, onde basicamente o foco é contar as frequências em que os termos ocorrem, formar um vetor e assim treinar o classificador, para isso pode-se utilizar de algumas técnicas tais como redes neurais artificiais, Naïve Bayesian, SVM, etc. Baseado em imagem, no qual é escaneado o conteúdo de imagem e analisados como estão agrupados os pixels da imagem, utilizando-se do histograma, é possível detectar combinações que levam a identificação em base de conhecimento de possível formações (técnica largamente utilizada para reconhecimento de pornografia), neste tipo de classificação as CNN (Convolutional neural network) tem ganhado destaque, mostrando resultados altamente relevantes no que tange a eficácia do processo de classificação. E a classificação baseada em combinação de texto com imagens.

Em um olhar comparativo, os autores demonstram as melhores estratégias de classificação são atingidas utilizando algoritmos baseados em KNN ou Naïve Bayes, já quando o conteúdo é exclusivamente de imagens, as CNN são mais eficientes. Mas a conclusão do estudo mostra que na área textual as tags e hiperlinks ainda são pouco utilizados na classificação. Na classificação visual, a aplicação de técnicas como as CNN dependem de recursos grandes e isso pode tornar a disseminação mais seletiva.

Agarap (2020) descreve em seu artigo, que compreender os sentimentos do cliente é de suma importância nas estratégias de marketing hoje. Isso não só dará às empresas *insights* sobre como os clientes percebem seus produtos e/ou serviços, mas também lhes dará uma ideia de como melhorar seu produto e/ou serviço. Neste artigo ele tenta entender a correlação de diferentes variáveis em avaliações de clientes em um *e-commerce* de roupas femininas, para classificar a recomendação dada pelo cliente e também o sentimento, seja positivo, negativo ou neutro. Para atingir esses objetivos, foram empregadas variáveis

univariadas e análises multivariadas sobre os recursos do conjunto de dados, implementando um sistema utilizando neural recorrente bidirecional com unidade de memória de longo prazo (LSTM) para recomendação e classificação de sentimento. Os resultados mostraram que uma recomendação é um forte indicador de um sentimento positivo pontuação e vice-versa. Por outro lado, as classificações nas análises de produtos são indicadores difusos de pontuações de sentimento. Também foi constatado que o LSTM bidirecional foi capaz de atingir um F1-score de 0,88 para classificação de recomendação e 0,93 para classificação de sentimento.

Observando diversas pesquisas a respeito do assunto, pode-se inferir que para atingir uma classificação eficiente, é pode-se utilizar mais do que uma técnica isolada, ou mais do que um método, pois a classificação depende muito de qual o tipo de dado será classificado bem como o tamanho deste conjunto de dados.

4 Experimentos

Tendo como objetivo central buscar uma solução para o problema de classificação de opiniões, em texto de uma área comercial específica, esta pesquisa é caracterizada como de natureza aplicada e de caráter exploratório, sendo adotado o procedimento de pesquisa bibliográfica a respeito do campo de mineração e classificação de textos, estudo de ferramentas disponíveis na área de classificação de textos e realização de testes numéricos. Para tal é necessário entender o problema analisando o conteúdo das opiniões expressadas em páginas da internet.

Este capítulo traz informações sobre os conjuntos de dados utilizados nos experimentos bem como a metodologia adotada.

4.1 Banco de dados

Alguns conjuntos de dados foram considerados para este estudo, o primeiro é um conjunto com dados do repositório *UCI Machine Learning Repository* (DUA; GRAFF, 2017), reconhecida fonte de dados para pesquisas científicas, são opiniões oriundas do site amazon.com, contém 500 frases positivas e 500 negativas. Este conjunto de dados é em língua inglesa e foi adotado para ser possível uma comparação, já que as diferentes soluções de mercado estão preparadas preferencialmente para este idioma.

O segundo é composto por opiniões de clientes de *e-commerce* da área da saúde, conta com 1382 opiniões positivas e 1274 opiniões negativas, em língua portuguesa, coletado pelo próprio autor e nos experimentos foi denominado ESU1. A classificação manual (humana) foi realizada para ser possível uma análise completa da matriz confusão.

Já na fase final de validação, os dados utilizados nos processos de classificação, serão do site ISP Saúde, empresa referência na área de varejo para saúde, comercializando produtos para fisioterapeutas, esteticistas, educadores físicos, médicos, massagistas e profissionais de enfermagem, um portfólio de mais 8 mil itens. O ISP Saúde está presente em 15 estados e possui mais de 30 lojas espalhadas pelo Brasil. Os dados fornecidos para este estudo são oriundos de uma pesquisa de satisfação pós-vendas real, realizada com clientes de todas as regiões do Brasil, entre os meses de janeiro e julho de 2021, contando com 475 opiniões positivas e 34 negativas. Os dados são anonimizados e respeitam todas as normas da Lei Geral de Proteção de Dados, Nº 13.709 (PLANALTO, 2018). Neste conjunto a classificação foi realizada pelo próprio cliente, já que juntamente com sua opinião ele deu um nota no formato de estrela correspondente ao grau de satisfação, 1 estrela para muito insatisfeito, 2 estrelas para insatisfeito, 3 estrelas para indiferente (neutro) 4 estrelas

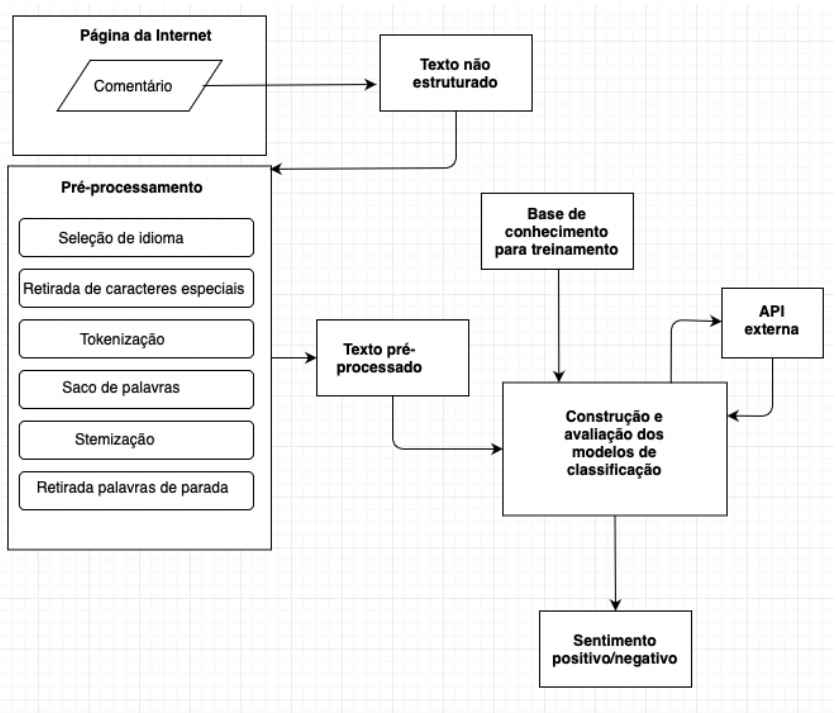
satisfeito e 5 estrelas para muito satisfeito. Em meio a tanta concorrência, as pesquisas de satisfação configuram-se como uma excelente ferramenta a fim de diagnosticar a satisfação do cliente com o objetivo de torná-lo fiel ao produto ou serviço. (KOTLER, 2003) afirma que a satisfação consiste na sensação de prazer ou desapontamento resultantes da comparação do desempenho percebido de um produto ou serviço em relação às expectativas, ou seja essas pesquisas são meios de entender a satisfação do cliente.

Na fase final, o segundo conjunto de dados, ESU1, foi utilizado como base de conhecimento para a classificação dos dados da pesquisa real do ISP Saúde, demonstrando assim o impacto das fases de treinamento da rede com uma base de conhecimento maior do que o conjunto analisado.

4.2 Metodologia

Segundo a revisão de literatura, a proposta para o problema de classificação de textos, seja ele multi categorizado ou binário como é o caso do sentimento positivo/negativo, tem sempre a fase de pré-processamento como um pilar para o sucesso do procedimento, fase descrita na Seção 2.3. O fluxo do processo planejado é mostrado na Figura 15.

Figura 15 – Fluxo de classificação



Fonte: Produzido pelo autor

Diferentes experimentos foram estabelecidos e efetuados com relação a execução deste fluxo. A intenção é também investigar o impacto do pré-processamento em dois tipos de redes, a rede MLP e a Recorrente. Uma etapa comparativa também foi realizada, utilizando um conjunto de dados em língua inglesa, testado em todas as soluções e outros

dois conjuntos em língua portuguesa, testado nas ferramentas que possuem suporte para este idioma. Os testes que utilizam parâmetros de língua inglesa foi realizado no sentido de comparar com as ferramentas comerciais existentes. Os diferentes experimentos realizados foram:

Experimento I - Rede Neural MLP . Os testes foram realizados tomando banco de dados UCI, ESU1 e ISP, considerando:

- Sem pré-processamento.
- Com pré-processamento, conforme Figura 15.

Experimento II - Rede Neural Recorrente . Os testes foram realizados tomando banco de dados UCI, ESU1 e ISP, considerando:

- Sem pré-processamento.
- Com pré-processamento, conforme Figura 15.

Experimento III - Ferramentas Comerciais . Os testes foram realizados considerando:

- Dados UCI, em língua inglesa.
- Dados ISP, em língua portuguesa.

4.2.1 Experimentos com Redes Neurais

Para experimentos I e II foram implementados algoritmos para fazer a classificação de sentimentos e os modelos avaliados conforme as métricas descritas na Seção 2.5. Os classificadores utilizados foram construídos usando redes neurais MLP e redes Recorrentes.

Os experimentos I e II foram realizados tomando 10 épocas e 50 épocas, e em ambos os testes o resultado admitido para a análise foi a média simples de 10 execuções.

Será analisado o impacto da classificação sem o pré-processamento comparado ao processo com pré-processamento e também o tempo de execução.

O modelo MLP foi montado com uma camada oculta de 128 neurônios utilizando a função de ativação $ReLU$, com uma camada de saída de apenas 1 neurônio utilizando a função de ativação sigmoide, os dados foram divididos em 30% para teste e 70% para treinamento. Devido à grande variação ocorrida por muitos neurônios na camada oculta, foi adotado um *dropout* de 0.5 e assim reduzido o sobreajuste. O uso destes parâmetros se deve aos resultados de simulações em testes preliminares, onde esta configuração apresentou um melhor desempenho.

Para a RNN, foi mantido o máximo de similaridade de parâmetros da MLP, ou seja, uma camada oculta com 128 neurônios, utilizando a função de ativação $ReLU$, porém o diferencial está na camada LSTM.

A ferramenta utilizada nestes experimentos foi o Google Colabs, (GOOGLE, 2021), para implementação em Python, com bibliotecas tais como NLTK (NLTK, 2021), *Pandas data analysis* (PANDAS, 2021) e *Scikit-learn machine learning* (SCIKIT-LEARN, 2021). O desenvolvimento foi implementado em blocos afim de ser possível a análise separadamente de cada sessão.

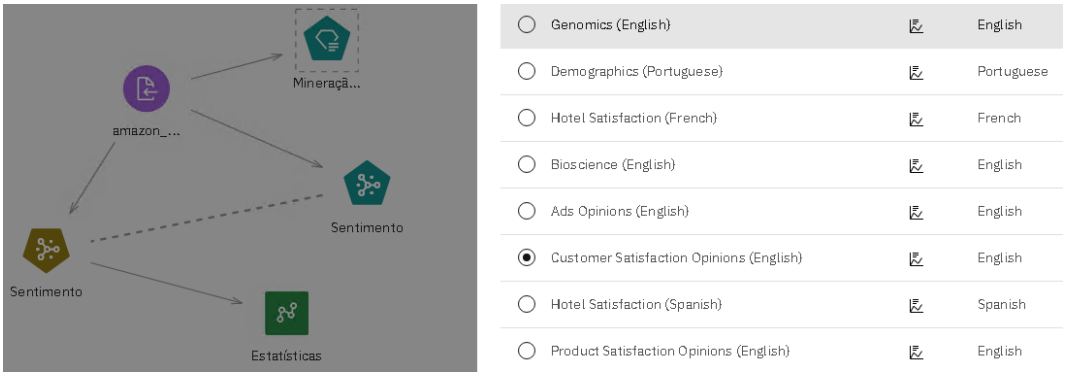
4.2.2 Experimentos com as ferramentas IBM-Watson e RapidMiner

Utilizando ferramenta IBM-Watson o processo foi um pouco mais simplificado, pois a ferramenta em sua versão gratuita é bastante limitada, tanto em parametrização quanto em quantidade de dados que ela está liberada para processar. Mas mesmo nestas condições foi possível realizar o processo de classificação e chegar a um percentual total de classificação utilizando redes multi camadas.

Como é uma ferramenta comercial, o grande core de sua inteligência está oculto, mas o conceito de trabalho nas simulações basicamente orientado a fluxos. A integração pode ser feita nas mais diferentes linguagens de programação, tais como Curl, Java, Node, Python, Go, .Net, etc.

A Figura 16 mostra a interface de comunicação com o usuário para mineração de texto. O parâmetro "Customer Satisfaction Opinions (English)", oferecido pela ferramenta, que é utilizado para base de conhecimento, identifica o tipo de dado oriundo da base de dados para realização do pré-processamento. O modelo utilizado, sugerido automaticamente pelo Watson, foi o Multi Camadas com 16 neurônios na camada oculta e 2 neurônios na camada de saída, com função de ativação tangente hiperbólica, em testes com a ferramenta Python, esta mesma configuração apresentou resultados bem inferiores.

Figura 16 – Modelo Watson



Fonte: Extraído do programa IBM-Watson Studio pelo Autor

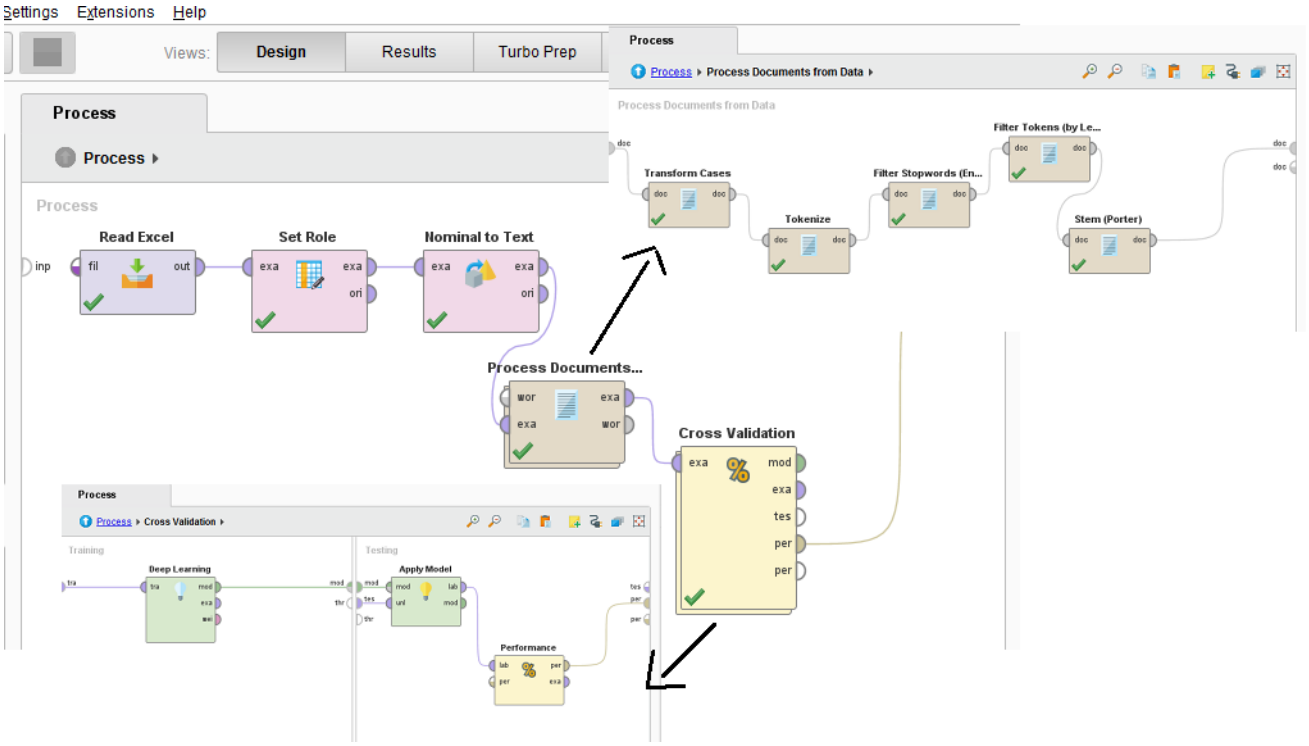
Utilizando ferramenta Rapidminer, que é plataforma de software de ciência de

dados desenvolvida pela empresa de mesmo nome e que fornece um ambiente integrado para preparação de dados, aprendizado de máquina, aprendizado profundo, mineração de texto e análise preditiva, foi realizado um experimento em blocos, separando as fases do pré-processamento e realizando a classificação do sentimento utilizando um bloco de aprendizado profundo.

A Figura 17 mostra a árvore de operadores principais, ao lado esquerdo, bem como as secundárias. A principal contém os blocos para entrada de dados, pré-processamento, aprendizagem e aplicação do modelo. O operador *Read Excel* acessa um arquivo em planilha contendo duas colunas, a primeira contém os textos e a segunda a classificação humana dos mesmos (positiva ou negativa). O operador *Set Role* indica o objetivo de um atributo, no caso altera para *label* indicando que as colunas são rótulos. O operador *Nominal Text* seleciona o campo atributo e o converte para o tipo texto, necessário para o próximo operador que é *Process Document from Data*. Este operador é quem vai processar as informações gerando os vetores segundo o modelo espaço vetorial descrito na Seção 2.2, sendo os subprocessos internos utilizados neste operador mostrados na Figura 17.

O operador *Cross-Validation* recebe como entrada os vetores produzidos e possui dois subprocessos, um de treinamento e um de teste. O subprocesso de treinamento separa parte do conjunto de dados para treinar o modelo escolhido e a outra parte dos dados é utilizada para o teste depois do treinamento. O modelo utilizado neste teste numérico foi o *Deep Learning*, que segundo o menu ajuda do RapiMiner é baseado em uma rede neural artificial multicamadas *feed-forward*, treinada com gradiente descendente estocástico usando retropropagação.

Figura 17 – Árvore de Operadores



Fonte: Extraído do programa RapidMiner pelo Autor

5 Resultados Numéricos

Os resultados apresentados aqui, que dependem de máquina local para execução, foram realizados em um *laptop* Apple Macbook Air M1, com 8Gb de memória e 256Gb de disco. Os tempos são padronizados em segundos.

5.1 Experimento 1: Rede MLP

O conjunto de dados da UCI, (DUA; GRAFF, 2017), foi submetido ao algoritmo escrito em Python, com as devidas parametrizações de idioma em inglês. Os resultados são apresentados conforme a Tabela 4, o pré-processamento não melhorou o resultado final, pois a métrica *F1_score* teve um decréscimo de 1,14%. O tempo de execução também não teve significativa alteração já que o tempo de processamento foi praticamente o mesmo da diferença de execução da rede, alterando o tempo total de execução em apenas 0.07 segundos, 0.55% mais lento.

Tabela 4 – Comparativo dados UCI - Rede MLP

Métrica	Sem Pré-Processamento	Com Pré-Processamento	Diferença	Percentual
Acuracia	0.70666664	0.69666665	-0.0099999	-1.42%
Precisão	0.72421205	0.71610414	-0.00810790	-1.12%
Recall	0.69868797	0.68533051	-0.01335746	-1.91%
F1_score	0.70340222	0.69538086	-0.00802135	-1.14%
Tempo da rede	6.19562077	4.50176668	-1.69385409	-27.34%
Tempo do pré	0.12074661	2.07028269	1.94953608	1614.57%
Tempo total	13.81727194	13.89291000	0.07563805	0.55%

Fonte: Produzido pelo autor

Na análise de sua matriz confusão, pode-se observar que o uso da MLP sem o pré-processamento gerou um resultado de 83,17% de classificação correta, conforme a Tabela 5, onde se esperava 500 opiniões positivas, foram classificadas 427 corretamente e onde era esperado 500 negativas foram classificadas 429 corretamente.

Tabela 5 – Matriz confusão da análise da rede MLP - UCI

	Analisado positivo	Analisado negativo
Positivo real	427	73
Negativo real	71	429

Fonte: Produzido pelo autor

No uso da rede MLP para classificar os dados ESU1, com 10 épocas, conforme podemos observar os resultados na Tabela 6, a métrica *F1_score* obteve uma melhora de 4,56 % quando executado com o pré-processamento, isso é refletido principalmente porque o *recall* aumentou em 15,84 %, esta diferença acontece principalmente por conta

do processo de retirada de *stopwords*, já que ele reduz a quantidade de informações da matriz de entrada.

Tabela 6 – Comparativo em 10 épocas - Rede MLP - Dados ESU1

Métrica	Sem Pré-Processamento	Com Pré-Processamento	Diferença	Percentual
Acuracia	0.71944793	0.72321205	0.00376411	0.52%
Precisão	0.76148101	0.71914131	-0.04233970	-5.56%
Recall	0.67580730	0.78282354	0.10701625	15.84%
F1_score	0.70911576	0.74143344	0.03231767	4.56%
Tempo da rede	2.26757109	2.25085564	-0.01671546	-0.74%
Tempo do pré	0.17548707	6.25428507	6.07879801	3463.96%
Tempo total	19.73386519	25.93177927	6.19791408	31.41%

Fonte: Produzido pelo autor

Os resultados dos testes em 50 épocas, mostrado na Tabela 7, demonstraram um comportamento um pouco diferente: É possível perceber uma melhora no desempenho que pode ser relacionado à quantidade de épocas, mas a métrica *F1_score* acaba diminuindo um pouco seu incremento, ficando em uma melhora de apenas 4,05 %, fato decorrente do cálculo das médias.

Esta diminuição do *F1_score* ao longo das épocas também está ligado a precisão que vai reduzindo ao longo das execuções, isso porque o algoritmo *Backpropagation* apesar de muito eficiente, não consegue armazenar memória para usar as correções nas posições corretas.

Tabela 7 – Comparativo em 50 épocas - Rede MLP- Dados ESU1

Métrica	Sem Pré-Processamento	Com Pré-Processamento	Diferença	Percentual
Acuracia	0.70326224	0.71430364	0.01104140	1.57%
Precisão	0.74244630	0.73247136	-0.00997493	-1.34%
Recall	0.65382549	0.71261047	0.05878497	8.99%
F1_score	0.68995567	0.71788630	0.02793064	4.05%
Tempo da rede	5.63642445	5.40403891	-0.23238554	-4.12%
Tempo do pré	0.17507670	6.24182527	6.06674857	3465.19%
Tempo total	23.15600665	29.02157345	5.86556680	25.33%

Fonte: Produzido pelo autor

O tempo do pré-processamento nas redes MLP, expresso em segundos, não impactou tanto no tempo total de execução do algoritmo, pois conforme as duas Tabelas 6 e 7, o tempo total apenas foi incrementado com o tempo gasto nas etapas de pré-processamento.

Observando a Tabela 8, verifica-se a precisão do modelo, onde se esperava 1382 opiniões positivas obteve-se 1095 e onde era esperado 1274 negativa, foi possível classificar 1181 corretamente.

Após estes testes, aplicamos o modelo sendo executado em 50 épocas ao conjunto de dados do ISP Saúde, onde estava populado com opiniões reais do *e-commerce*, com 475 positivas e 34 opiniões negativas. O modelo obteve um resultado muito bom, pois como se trata de uma coleta total, sem balancear a mesma quantidade para positivos e negativos,

Tabela 8 – Matriz confusão da análise da rede MLP - Dados ESU1

	Analisado positivo	Analisado negativo
Positivo real	1095	179
Negativo real	199	1181

Fonte: Produzido pelo autor

teve um alto índice de acerto, chegando em alguns ensaios à uma precisão de 94% com um *F1_score* de 97%, conforme a Tabela 9.

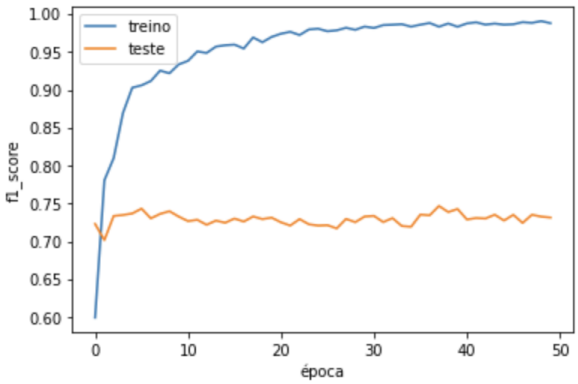
Tabela 9 – Matriz confusão da análise da rede MLP - ISP

	Analisado positivo	Analisado negativo
Positivo real	475	0
Negativo real	9	15

Fonte: Produzido pelo autor

Conforme observado nos experimentos, a fase de pré-processamento não impactou consideravelmente no resultado final, pois teve um desempenho 6,59% inferior ao modelo sem pré-processamento, isso pode-se justificar pelo uso de *Embeddings*. Quando comparado o *F1_score* neste modelo, Figura 18, observa-se que existe uma diferença grande entre a fase de treino e de teste, fato que pode ser justificado pela ineficiência da fase de correção do erro que não possui uma memória temporal, pois conforme observado na mesma figura os melhores resultados são obtidos por volta da terceira época, após isso as médias vão sendo impactadas pela grande diferença entre treino e teste.

Figura 18 – *F1_score* Rede MLP



Fonte: Extraído do Google Colabs pelo Autor

Tabela 10 – Matriz confusão da análise MLP - ISP treinado

	Analisado positivo	Analisado negativo
Positivo real	475	0
Negativo real	7	17

Fonte: Produzido pelo autor

Finalizando os testes desta solução, foi utilizado o modelo já treinado com os dados ESU1, para classificar os dados do ISP Saúde, Tabela 10 como se trata de um conjunto de

dados pequeno, o impacto não foi tão grande, porém a precisão aumentou em cerca de 1,33% atingindo 95,3%.

5.2 Experimento 2: Rede Recorrente

Os teste iniciais utilizando os dados da UCI, (DUA; GRAFF, 2017), demonstraram um diferença quase insignificativa do *F1_score* sem o pré-processamento e com o pré-processamento, apenas 0.32%, tendo seu melhor resultado com 78.30% de *F1_score* sem o pré-processamento.

Tabela 11 – Comparativo dados UCI - Rede RNN

Métrica	Sem Pré-Processamento	Com Pré-Processamento	Diferença	Percentual
Acuracia	0.78666669	0.79000000	0.00333333	0.42%
Precisão	0.77865117	0.79836595	0.01971477	2.53%
Recall	0.79857015	0.77998936	-0.01858079	-2.33%
F1_score	0.78308421	0.78060656	-0.00247764	-0.32%
Tempo da rede	47.44399142	50.72790384	3.28391242	6.92%
Tempo do pré	0.13548231	2.16037249	2.02489018	1494.58%
Tempo total	55.21782708	57.54800057	2.33017349	4.22%

Fonte: Produzido pelo autor

Quando analisada a matriz confusão desta classificação, podemos verificar conforme a Tabela 12, que onde se esperava 500 opiniões positivas, o algoritmo conseguiu classificar 415 e onde eram esperadas 500 negativas, foram classificadas 438, o que gera um índice de acerto de 82,75%.

Tabela 12 – Matriz confusão da análise da rede RNN - UCI

	Analisado positivo	Analisado negativo
Positivo real	415	85
Negativo real	62	438

Fonte: Produzido pelo autor

Já nos testes com o conjunto de dados em português, ESU1, apresentaram resultados bem estáveis ao longo de suas repetições para a composição das médias. Observando a Tabela 13, na primeira bateria de testes, com os dados referente à 10 épocas, pode se observar que a métrica *F1_score* se alterou apenas 0,61 % entre os testes com pré-processamento e sem pré-processamento. Já o tempo de execução do algoritmo correspondente a rede teve uma boa melhora, reduzindo em 18 % a duração quando aplicado o pré-processamento, impactando em um tempo total 1,41 % mais rápido na execução total do algoritmo.

Já na execução em 50 épocas, Tabela 14, observa-se uma melhora de eficiência de 1,53 % quando aplicado o pré-processamento, porém o tempo de execução da rede foi incrementado em 5,04 %, impactando em uma demora de 7,73 % no tempo total de execução quando comparado ao tempo sem o pré-processamento.

Tabela 13 – Comparativo em 10 épocas - Rede Recorrente - Dados ESU1

Métrica	Sem Pré-Processamento	Com Pré-Processamento	Diferença	Percentual
Acuracia	0.80125471	0.79209535	-0.00915936	-1.14%
Precisão	0.83585003	0.78184614	-0.05400389	-6.46%
Recall	0.79153336	0.85090089	0.05936754	7.50%
F1_score	0.80180963	0.80673777	0.00492815	0.61%
Tempo da rede	41.72537892	33.99199686	-7.73338206	-18.53%
Tempo do pré-processamento	0.19747157	6.66904001	6.47156844	3277.22%
Tempo total	59.78543386	58.94174845	-0.84368541	-1.41%

Fonte: Produzido pelo autor

Tabela 14 – Comparativo em 50 épocas - Rede Recorrente - Dados ESU1

Métrica	Sem Pré-Processamento	Com Pré-Processamento	Diferença	Percentual
Acuracia	0.83400251	0.81781681	-0.01618570	-1.94%
Precisão	0.83707503	0.80857760	-0.02849743	-3.40%
Recall	0.85232165	0.85436141	0.00203976	0.24%
F1_score	0.83897910	0.82615021	-0.01282889	-1.53%
Tempo da rede	136.20773442	143.07577102	6.86803660	5.04%
Tempo do pré-processamento	0.20097661	6.24915535	6.04817874	3009.39%
Tempo total	154.83244247	166.79771168	11.96526921	7.73%

Fonte: Produzido pelo autor

Analisando a Tabela 15, verifica-se a precisão do modelo, onde se esperava 1382 opiniões positivas o modelo obteve-se 1113 e onde era esperado 1274 negativa, foi possível classificar 1258 corretamente, gerando um índice médio total de aproximadamente 88%.

Tabela 15 – Matriz confusão da análise da rede recorrente - Dados ESU1

	Analisado positivo	Analisado negativo
Positivo real	1113	161
Negativo real	122	1258

Fonte: Produzido pelo autor

O algoritmo para classificação de sentimentos com rede neural recorrente apresentou sua melhor performance entre 10 e 20 épocas, que conforme a Figura 19, a métrica *F1_score*, atingia uma pequena diferença entre treino e testes e mesmo assim com número próximo ao seu melhor resultado.

Ao aplicar este modelo ao conjunto de dados do ISP, podemos ver a importância da escolha do modelo, pois mesmo sendo as redes recorrentes mais indicadas para este tipo de classificação, o resultado foi inferior ao resultado do modelo com redes multi camadas.

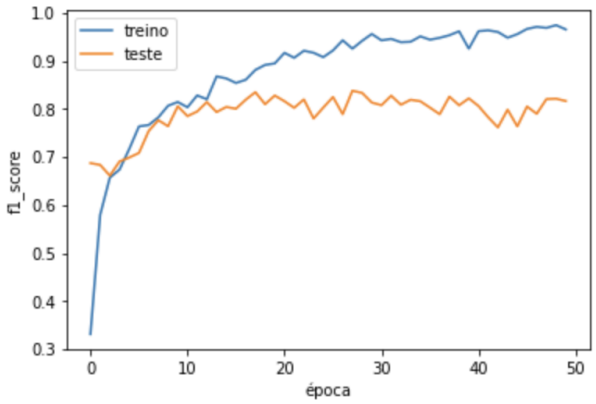
Tabela 16 – Matriz confusão da análise da rede recorrente - ISP

	Analisado positivo	Analisado negativo
Positivo real	472	0
Negativo real	18	16

Fonte: Produzido pelo autor

Conforme observado na Tabela 16, a quantidade opiniões positivas que foram classificadas corretamente foi grande, porém como são poucas opiniões negativas, as que

Figura 19 – *F1_score* Rede Recorrente



Fonte: Extraído do Google Colabs pelo Autor

foram classificadas incorretamente fazem com que o impacto no cálculo final seja mais representativo, diminuindo a eficiência total do algoritmo, com 76,82% de precisão.

Já com o modelo treinado, utilizando os dados ESU1 como base de conhecimento, a quantidade de opiniões classificadas corretamente aumenta, chegando a uma precisão de 82,59%, conforme Tabela 17.

Tabela 17 – Matriz confusão da análise RNN- ISP treinado

	Analisado positivo	Analisado negativo
Positivo real	472	0
Negativo real	13	21

Fonte: Produzido pelo autor

5.3 Experimento 3: Ferramentas IBM-WATSON e RapidMiner

A Tabela 18, mostra a matriz de confusão para os dados da UCI, analisados com o Software RapidMiner. Tem-se que dos 500 textos classificados como positivos pelos humanos (“verdadeiro positivo”), 376 foram classificados corretamente como positivos (“predito positivo”), com taxa de cobertura *recall* de 75,2%. Dos 500 textos classificados como negativos (“verdadeiro negativo”), 353 exemplos foram classificados corretamente como negativo (“predito negativo), o que resultou em uma taxa de cobertura de 70,6%. Pode-se afirmar que o classificador gerado possui uma exatidão um pouco maior para identificar textos com sentimento positivo.

Tabela 18 – Matriz confusão da análise RapidMiner - UCI

	Analisado positivo	Analisado negativo
Positivo real	376	124
Negativo real	147	353

Fonte: Produzido pelo autor

Um dos principais atrativos e vantagens do RapidMiner é sua interface, que oferece ao usuário os componentes em formato de blocos, dispensando o conhecimento em programação efetivamente. Porém estes componentes acabam limitando um pouco a análise, pois o processo de validação de conectivos é bastante complexa. Vencida a etapa de montagem da linha de processo, a execução da análise consumiu bastante recurso computacional e tempo da máquina, em alguns casos não concluindo a execução, sendo esse um dos principais pontos de dificuldade.

Nos testes utilizando o IBM-Watson, os resultados foram com maior precisão, conforme analisado a seguir, conforme mostrado na Tabela 19

Tabela 19 – Matriz confusão da análise Watson - UCI

	Analisado positivo	Analisado negativo
Positivo real	500	0
Negativo real	149	351

Fonte: Produzido pelo autor

No teste com o conjunto de dados de opiniões da UCI, conforme a matriz confusão que é mostrada na Tabela 19, tem-se que das 500 opiniões negativas, 351 foram classificadas corretamente, já as positivas chegaram à 500 classificadas corretamente. O índice total de classificação correta chegou à 85,1%. A exatidão do modelo atingiu 0.851, com uma taxa de falso positivo ponderado de 0.081.

Como o IBM-Watson apresenta suporte para língua portuguesa, foi possível realizar os testes tal como na solução escrita em Python, ressalvada a questão da parametrização que é fixa e muitos dados não são disponibilizados.

Quando os testes foram realizados com o conjunto de dados com opiniões de *e-commerce*, os resultados tiveram uma variação de precisão dependendo da quantidade de dados que eram avaliados, perdendo a eficiência quando analisado o conjunto inteiro com todas as 2656 opiniões, atingindo apenas 67,6%, com uma taxa de falso positivo de 32,3% e um *F1_score* de 67,8%. Conforme a Tabela 20, quando eram esperadas 1382 opiniões positivas, o modelo classificou corretamente apenas 962, já quando eram esperados 1274 negativas foi classificado 691 opiniões apenas. Este resultado se deve ao teste ser realizado apenas com a licença de testes, já que não é possível armazenar o treinamento para futuras analises, ou seja, cada execução o modelo precisa ser treinado novamente.

Tabela 20 – Matriz confusão da análise Watson - Dados *e-commerce*

	Analisado positivo	Analisado negativo
Positivo real	962	467
Negativo real	325	691

Fonte: Produzido pelo autor

Os testes do Watson com o conjunto de dados do ISP Saúde mostraram resultados

com índices de precisão altos, aproximadamente 99,8%, conforme pode ser observador na Tabela 21, onde das 475 opiniões positivas o modelo classificou corretamente 447, ficando 28 opiniões sem serem classificadas, já das opiniões negativas o modelo conseguiu classificar corretamente 33 das 34, com um falso positivo de apenas 0,027%.

Tabela 21 – Matriz confusão da análise Watson - ISP

	Analisado positivo	Analisado negativo
Positivo real	447	0
Negativo real	1	33

Fonte: Produzido pelo autor

Um ponto que merece destaque é a eficiência do Watson quando o conjunto de dados aumenta de tamanho, pois quando os testes eram realizados com os dados da UCI (1000 opiniões) e os dados do ISP (509 opiniões) os resultados eram com altos índices de precisão, 85,1% e 99,8% respectivamente, mas quando realizado o teste com o conjunto de dados de produtos da área da saúde, coletados de *e-commerce* (2656 opiniões coletadas) a precisão caiu para 67,6%.

O programa oferece um ambiente integrado, com ampla gama de opções de mineração de dados e processamento de informação que permitem fluxo centrado na modelagem. É uma solução focada em ciência de dados, porém totalmente dependente de licença. Os testes foram realizados dentro de um período de licença demonstrativa, com recursos limitados.

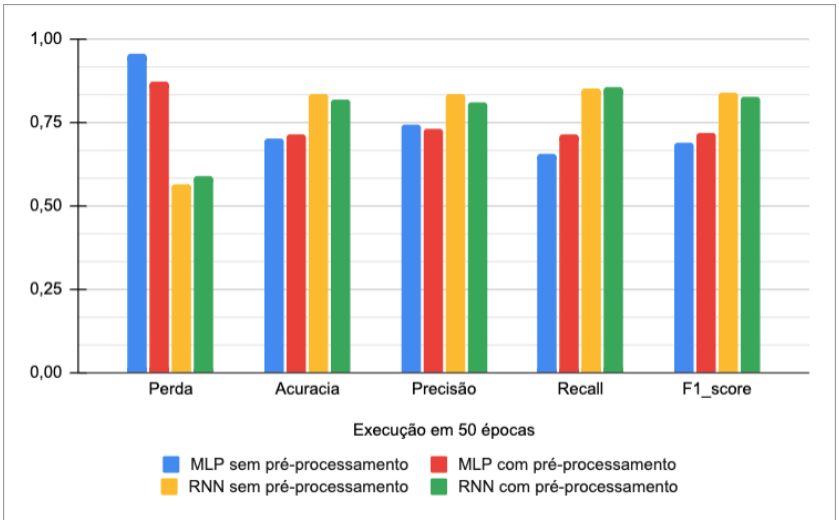
5.4 Comparativos

Após utilizar a solução com redes neurais multi camadas e rede neurais recorrentes, uma dúvida pode ser esclarecida, ambas as técnicas podem ser utilizadas para o problema abordado, já que para os padrões de mercado o índice de acerto está bem próximos das melhores soluções comerciais atualmente difundidas, conforme apresentado por Altexsoft (2021).

Observando a Figura 20, pode-se verificar que a solução em rede neural recorrente, RNN, apresenta um melhor desempenho em todas as métricas de avaliação, pois obteve uma maior acurácia e precisão, atingindo um *F1_score* geral mais elevado do que as redes MLP. Já no que tange a análise do pré-processamento, vemos que nas MLP ele impacta em um melhor resultado, já nas recorrentes isso não acontece, pois diferentemente das redes neurais tradicionais, as entradas de uma rede neural recorrente não são independentes umas das outras, e os resultados para cada elemento dependem da computação dos elementos precedentes, sendo assim o uso do pré-processamento em uma MLP reduz a dimensionalidade dos elementos, tornando mais simples a análise, já que os dados são de

"microtextos"e não exigem tanto da rede em si, mas da quantidade de iterações necessárias para classificar.

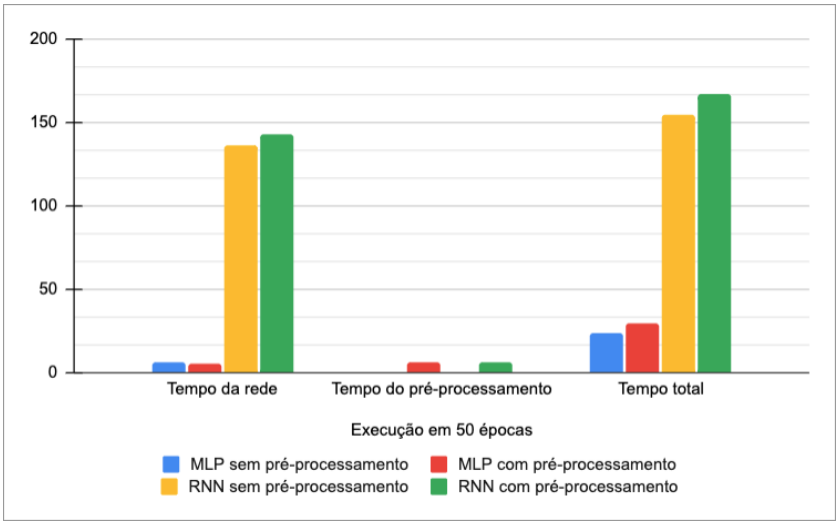
Figura 20 – Comparativos de métricas MLP vs RNN



Fonte: Produzido pelo autor

Quando analisado o tempo gasto nas soluções de redes neurais, Figura 21, observa-se que as redes neurais recorrentes necessitam de mais tempo para serem executadas, dada a sua complexidade. Como o algoritmo foi construído de forma modular para comparar as duas soluções, o tempo gasto no pré-processamento em ambas as soluções é praticamente igual.

Figura 21 – Comparativos de tempos MLP vs RNN



Fonte: Produzido pelo autor

Conclui-se que a solução aplicada tanto com redes neurais multicamadas quanto as redes neurais recorrentes, aqui implementadas em Python, podem atingir resultados próximos aos obtidos com ferramentas comerciais. Conforme os detalhes na Tabela 22, pode-se observar que em alguns testes, o modelo implementado em Python se comportou

com mais precisão do que o IBM Watson, como no caso do conjunto de dados maior, onde foi possível chegar a 83,70% de precisão com redes neurais recorrentes.

Tabela 22 – Comparativos de melhores resultados

	UCI	*e-commerce*	ISP
IBM- Watson	85.10%	67.60%	99.80%
MLP em Python	83.17%	74.44%	94.00%
RNN em Python	79.83%	83.70%	76.82%

Fonte: Produzido pelo autor

É fato que deve-se considerar que quando submetemos um conteúdo à ferramentas proprietárias, não é possível saber qual o método está sendo utilizado para a classificação, mas sabe-se que em muitos casos utiliza-se redes neurais.

6 Considerações Finais

Em um contexto tal qual o escopo descrito nesta pesquisa, onde grande parte das informações armazenadas pelas organizações está na forma textual, faz-se necessário o desenvolvimento de técnicas computacionais para a exploração do conhecimento nelas contido. A tarefa de análise de sentimentos é complexa pois é algo essencialmente subjetivo e do próprio ser humano, esta classificação sendo realizada por máquina ainda é um desafio. Os resultados demostram que as ferramentas utilizadas no trabalho permitem realizar o processo de classificação de textos com bom desempenho quando comparado à ferramentas comerciais. Este processo pode ser aprimorado com uma etapa de treinamento mais consistente (uma base conhecimento maior).

Quando comparado os tipos de arquitetura de redes, observou-se também que o uso de redes neurais recorrentes é mais indicado para este tipo de problema, a classificação de sentimentos em microtextos, porém é necessário mensurar muito bem sua aplicação, dado o consumo de recursos de máquina e de tempo para o processamento.

Já o impacto da fase de pré-processamento no resultado da classificação é mais percebido nas redes neurais multicamadas, pois a redução da dimensionalidade da matriz que entra na rede neural otimiza seu processamento. Já as redes neurais recorrentes não se aproveitam tanto deste recurso, pois sua arquitetura já atinge resultados melhores com o texto em sua forma original, utilizando-se do principio de recorrência através do tempo (entre seus neurônios).

Apesar da grande maioria das pesquisas focarem em testes na língua inglesa, os resultados obtidos no processamento em língua portuguesa se mostraram satisfatórios, demonstrando que as bibliotecas utilizadas estão preparadas para este idioma, desde que devidamente parametrizados. A precisão atingida em língua inglesa é próxima à portuguesa, evidenciando a evolução dos algoritmos para processamento do português escrito.

Observa-se ainda que, conforme Igual (2017) menciona em sua obra, em muitas pesquisas as pessoas se atentam apenas para uma linguagem ou para uma solução possível, mas como demonstrado neste trabalho, existem diversas formas de se chegar ao ponto desejado, deve-se observar então o tipo de informação, o nível de precisão esperado, o tempo disponível e os recursos de máquina empregados.

Com relação aos textos (opiniões) em *e-commerce*, destacamos que os modelos gerados se mostraram bons para a tarefa de classificação de polaridade (positiva/negativa).

Como trabalhos futuros pretende-se aplicar outros métodos para buscar melhores resultados no processo de classificação. Realizar outros experimentos alterando configura-

ções dos modelos aplicados neste trabalho principalmente para resolver o problema do *overfitting* nas redes MLP.

Referências

4ALL. *Os impactos da pandemia no ecommerce brasileiro*. 2020. Https://4all.com/blog/pandemia-e-commerce. Citado na página 15.

AGARAP, A. F. M. Statistical analysis on e-commerce reviews, with sentiment classification using bidirectional recurrent neural network. *arXvi*, 2020. Citado na página 48.

AGGARWAL, C. C. *Neural Networks and Deep Learning*. Switzerland: Springer, 2018. ISBN 9783319944630. Citado 4 vezes nas páginas 23, 24, 31 e 38.

ALTEXSOFT. *Comparing Machine Learning as a Service: Amazon, Microsoft Azure, Google Cloud AI, IBM Watsonl*. 2021. Https://www.altexsoft.com/blog/datascience/comparing-machine-learning-as-a-service-amazon-microsoft-azure-google-cloud-ai-ibm-watson/. Citado na página 63.

BRAMER, M. *Principles of Data Mining*. London: Springer, 2016. ISBN 9781447173076. Citado 8 vezes nas páginas 8, 17, 18, 20, 22, 35, 37 e 38.

BRASIL, E. *E-commerce no Brasil bate recorde*. 2021. Https://www.ecommercebrasil.com.br/noticias/e-commerce-no-brasil-bate-recorde-e-atinge-r-53-bilhoes-ebit-nielsen-webshoppers/. Citado na página 15.

CARVALHO, M. H. de. Estudo Comparativo dos Métodos de Word Embedding na Análise de Sentimentos. Universidade Federal de Pernambuco, 2018. Citado 2 vezes nas páginas 20 e 21.

CASTRO, G. B. Modelo de rede neural bioinspirada para o controle de trânsito urbano. PUC-SP, 2017. Citado na página 24.

CHAVES, R. R. Redes neurais deep learning aplicadas ao reconhecimento facial. UCS, 2018. Citado na página 24.

CHRISTEN, M. et al. Micro-text classification between small and big data. *Nonlinear Theory and Its Applications, IEICE*, v. 6, p. 556–569, 10 2015. Citado na página 45.

DUA, D.; GRAFF, C. *UCI Machine Learning Repository*. 2017. Disponível em: <http://archive.ics.uci.edu/ml>. Citado 3 vezes nas páginas 50, 56 e 59.

DUTRA, V. P. Redes neurais e o reconhecimento de padrões de texto. USF, 2011. Citado na página 24.

EFRON, B.; J.TIBSHIRANI, R. *An Introduction to the Bootstrap*. São Paulo: Springer Science, 1993. ISBN 9780412042317. Citado na página 38.

ERTEL, W. *Introduction to Artificil Intelligence*. Weingarten: Springer, 2017. ISBN 9783319584867. Citado 8 vezes nas páginas 23, 24, 26, 27, 28, 29, 30 e 32.

FACURE, M. *Aprendendo Representações de Palavras*. 2017. Https://matheusfacure.github.io/2017/03/20/word2vec/. Citado na página 21.

GARTNER. *Gartner Glossary*. 2021. Https://www.gartner.com/en/information-technology/glossary/natural-language-processing-nlp. Citado na página 16.

GHIASSI, M. et al. Automated text classification using a dynamic artificial neural network model. *Expert Syst. Appl.*, Maryland, v. 39, p. 10967–10976, 2012. Citado 3 vezes nas páginas 13, 44 e 45.

GHOSHAL, S. R. S. Thresholded ConvNet ensembles: neural networks for technical forecasting. Springer Nature journal, 2020. Citado na página 31.

GLOVE. *GloVe: Global Vectors for Word Representation*. 2014. Https://nlp.stanford.edu/projects/glove/. Citado na página 21.

GOOGLE. *Google Colaboratory*. 2021. Https://colab.research.google.com/. Citado na página 53.

HASHEMI, M. Web page classification: a survey of perspectives, gaps, and future directions. *Multimedia Tools and Applications*, 2020. Citado na página 47.

HAYKIN, S. *Redes neurais: princípios e prática*. São Paulo: Bookman, 2001. ISBN 0132733501. Citado 4 vezes nas páginas 25, 26, 27 e 30.

HEARST, M. A. Untangling Text Data Mining. ACL'99: the 37th Annual Meeting of the Association for Computational Linguistics.College Park: University of Maryland, 1999. Citado 2 vezes nas páginas 16 e 19.

IBM. *IBM Knowledge Center*. 2021. Https://www.ibm.com/br-pt/cloud/watson-natural-language-understanding. Citado 2 vezes nas páginas 42 e 43.

IGUAL, S. S. L. *Introduction to Data Science*. Barcelona: Springer, 2017. ISBN 9783319500164. Citado 3 vezes nas páginas 22, 36 e 66.

KARPATHY, A. *The Unreasonable Effectiveness of Recurrent Neural Networks*. 2015. Https://karpathy.github.io/2015/05/21/rnn-effectiveness/. Citado na página 34.

KINGMA, J. L. B. D. P. Adam: A method for stochastic optimization. *ICLR*, 2015. Citado na página 41.

KOTLER, G. A. P. *Principios de marketing*. São Paulo: Person Pretince Hall, 2003. Citado na página 51.

KUMAR, T. M. S. A. Sentiment Analysis: A Perspective on its Past, Present and Future. International Journal of Intelligent Systems and Applications, 2012. Citado 4 vezes nas páginas 13, 14, 39 e 40.

LEON, J. S. *Álgebra linear com aplicações*. Rio de Janeiro: LTC, 1999. Citado na página 29.

LOH, S.; WIVES, L. K.; FRAINER, A. S. *Recuperação semântica de documentos textuais na internet*. Porto Alegre: Universidade Federal do Rio Grande do Sul, 2004. Citado 2 vezes nas páginas 13 e 16.

MELLO, A. J. T. S. Uso de técnicas de redes neurais em instrumentação para astronomia. UFSC, 2014. Citado na página 24.

MENOTTI, D. Multiple Layer Perceptron. Ufpr, 2018. Citado na página 31.

MOURA, M. F. *Proposta de utilização de mineração de textos para seleção, classificação e qualificação de documentos*. São Paulo: Embrapa, 2004. ISSN 1677-9274. Citado 2 vezes nas páginas 13 e 16.

NLTK. *Nltk Project*. 2021. Https://www.nltk.org/. Citado na página 53.

NUNES, M. das G. V.; LUCCA, J. L. D. Lematização versus Stemming. NILC - ICMC-USP, 2002. Citado na página 19.

PANDAS. *Pandas Data Analysis*. 2021. Https://pandas.pydata.org/. Citado na página 53.

PLANALTO. Lei Geral de Proteção de Dados. Planalto, 2018. Citado na página 50.

RAPIDMINER. *RapidMiner Studio Manual*. 2014. Https://docs.rapidminer.com/downloads/RapidMiner-v6-user-manual.pdf. Citado na página 42.

REZENDE, S. O.; MARCACINI, R. M.; MOURA, M. F. O uso da Mineração de Textos para Extração e Organização Não Supervisionada de Conhecimento. Fsma, 2011. Citado na página 17.

SANTOS, V. dos. *Como remover stopowords em Python*. 2018. Https://www.computersciencemaster.com.br/como-remover-stopwords-em-python/. Citado na página 19.

SCIENCE, A. D. *Deep Learing Book*. 2021. Https://www.deeplearningbook.com.br/. Citado 4 vezes nas páginas 8, 22, 33 e 34.

SCIKIT-LEARN. *Scikit-Learn, Machine Learning in Python*. 2021. Https://scikit-learn.org/stable/. Citado 3 vezes nas páginas 40, 41 e 53.

SEBRAE, S. B. d. A. a. M. e. P. E. *Uma breve definição sobre o comércio online*. 2013. Https://www.sebrae.com.br/sites/PortalSebrae/artigos/uma-breve-definicao-sobre-o-comercio-online. Citado na página 15.

SKANSI, S. *Introduction to Deep Learning*. Zagreb, Croatia: Springer, 2018. ISBN 9783319730035. Citado 2 vezes nas páginas 18 e 33.

SOUZA, C. G. e; SOUZA, L. de Mello Tostes e. O Uso de Redes Neurais Artificiais no Diagnóstico de Doenças Reumatológicas. UFSC, 2004. Citado na página 24.

SOUZA, L. F. S. d.; GONçALVES, A. L.; SOUZA, J. A. d. UtilizaÇÃo prÁtica de word embedding aplicada À classificaÇÃo de texto. *Anais do Congresso Internacional de Conhecimento e Inovação – ciki*, v. 1, n. 1, nov. 2020. Disponível em: <https://proceeding.ciki.ufsc.br/index.php/ciki/article/view/899>. Citado na página 21.

THOMAZ, C. E.; VELLASCO, M. M. Análise de Tendências de Mercado por Redes Neurais Artificiais. PUC-RJ, 2016. Citado na página 24.

TOWARDS, D. S. *Skip-Gram: NLP context words prediction algorithm*. 2019. Https://towardsdatascience.com/skip-gram-nlp-context-words-prediction-algorithm-5bbf34f84e0c. Citado na página 21.

YOUTUBE. *You Tube*. 2021. Https://www.youtube.com/intl/pt-BR/about/. Citado na página 13.