UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Jéssica Feliciano Coutinho

Avaliação de técnicas de Combinação de *Embeddings* para a Análise de Sentimentos de Produtos escritos em Português-BR

Recife

2022

Jéssica Feliciano Coutinho

Avaliação de técnicas de Combinação de *Embeddings* para a Análise de Sentimentos de Produtos escritos em Português-BR

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, como requisito parcial para obtenção do título de Mestre em Ciência da Computação.

Área de Concentração: Inteligência Computacional

Orientador (a): Leandro Maciel Almeida

Recife

2022

Catalogação na fonte
Bibliotecária Nataly Soares Leite Moro, CRB4-1722

C871a Coutinho, Jéssica Feliciano
Avaliação de técnicas de combinação de *embeddings* para a análise de sentimentos de produtos escritos em português-BR / Jéssica Feliciano Coutinho. – 2022.
117 f.: il., fig., tab.

Orientador: Leandro Maciel Almeida.
Dissertação (Mestrado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, Recife, 2022.
Inclui referências.

1. Inteligência computacional. 2. Análise de sentimentos. 3. Word embeddings. 4. Aprendizagem de máquina. 5. Aprendizagem profunda. I. Almeida, Leandro Maciel (orientador). II. Título

006.31 CDD (23. ed.) UFPE - CCEN 2022 – 204

Jéssica Feliciano Coutinho

“Avaliação de técnicas de Combinação de Embeddings para a Análise de Sentimentos de Produtos escritos em Português-BR”

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação. Área de Concentração: Inteligência Computacional

Aprovado em: 26/05/2022.

BANCA EXAMINADORA

_____

Prof. Dr. Luciano de Andrade Barbosa
Centro de Informática/UFPE

_____

Prof. Dr. João Fausto Lorenzato de Oliveira
Escola Politécnica de Pernambuco / UPE

_____

Prof. Dr. Leandro Maciel Almeida
Centro de Informática / UFPE
(**Orientador**)

Dedico à pessoa que acreditou em mim mais do que eu mesma, e me ajudou a desistir de desistir.

AGRADECIMENTOS

Este trabalho foi resultado de muita dedicação e esforço, de muita luta interna e até mesmo de muitas lágrimas, de modo que não seria possível concluí-lo sem o suporte fora de série de algumas pessoas incríveis que tenho na minha vida. Gostaria de começar agradecendo à minha mãe, Izabel Feliciano, e minha tia, Zenilda Feliciano, por acreditarem em mim mesmo sem fazerem ideia do que eu estava fazendo e por se orgulharem de quem sou, sem elas eu não teria ido tão longe. Agradeço também aos meu incríveis amigos da "Quinta Série": Érika, Gabi, Hugo, Márcio, Mika e Nenata, vocês tornaram esses longos anos pandêmicos e cheios de altos e baixos extremamente leves, obrigada por existirem na minha vida. Agradeço à Eliza Thaís, minha amiga e quase irmã de longa data, simplesmente por fazer parte de toda minha trajetória acadêmica, por compartilharmos de sentimentos tão parecidos e por me ensinar demais sobre tudo em cada conversa cheia de risos. Agradeço imensamente a Tarciso Lucena, psicólogo e amigo, que me guia de todas as formas possíveis e me fez aprender que, mesmo com todos os vales, eu estou sim numa curva ascendente repleta de picos. Agradeço também à Thaysa Barros, por sua amizade maravilhosa e por ter sido minha dupla no projeto da graduação que me fez querer iniciar na área do meu tema de pesquisa. Agradeço à Vitória Maria, por todas as pitangas choradas juntas e o suporte que só alguém que está passando pela mesma situação consegue fornecer. Agradeço a Washi por ainda me amar e ser meu amigo mesmo eu estando sumida devido à roda-viva do mestrado/trabalho. Agradeço demais ao meu time de Analytics da Neurotech, por estarem comigo no dia a dia me ouvindo reclamar constantemente e tornando o ambiente de trabalho mais leve. Por fim, não tenho nem palavras para agradecer ao meu amor, Jorge Delgado, por todo o suporte (emocional e técnico), acolhimento, conselhos, por tudo. Por acreditar em mim muito mais do que eu mesma, por não me deixar desistir, por ser meu parceiro e melhor amigo, por estar ao meu lado em todos os momentos, bons e ruins: sem você eu não teria conseguido. No mais, agradeço a todos que passaram pelo meu caminho nos últimos três anos e contribuíram para este projeto de alguma forma. Agradeço também a mim mesma, por não ter desistido, por ter dado o máximo de mim, mesmo quando achava que não mais conseguiria.

RESUMO

A Análise de Sentimentos é uma área de pesquisa voltada para a determinação da polaridade do sentimento presente em um texto, buscando identificar se a informação é de caráter positivo, negativo ou neutro, dentre outras formas de classificação. Com o expressivo volume de informações textuais que circulam na *web* diariamente, o processo de análises automáticas dos sentimentos torna-se ainda mais necessário. Para conteúdos relacionados à avaliação de produtos e serviços, a detecção de sentimentos é de grande relevância, uma vez que entender a mensagem que um consumidor está querendo passar sobre um produto é essencial para as empresas por diversos fatores, dentre eles campanhas de *marketing* e melhoria no relacionamento com seus clientes. Nesse cenário, o estudo das formas de melhorar a representação das informações textuais, de modo que elas sejam processadas através de modelos de aprendizagem de máquina, é de extrema importância para contribuir com o aumento de performance na classificação dos sentimentos presentes nos textos. Diante disto, o presente trabalho realiza um estudo experimental do comportamento de diferentes técnicas de vetorização de textos, com foco nos *embeddings*: vetores representativos compostos por valores reais capazes de armazenar informações sintáticas e semânticas das palavras. Para isso, são avaliados diferentes tipos de vetores de *embeddings* e três formas de combinação desses vetores, que são utilizados no processo de classificação de cinco diferentes algoritmos de aprendizagem de máquina. Além disso, também é utilizado um algoritmo de aprendizagem profundo onde a etapa de *embedding* é realizada pela própria camada da rede neural. Com o intuito de contribuir com a Análise de Sentimentos de textos em Português-BR, foram estudadas quatro bases de dados neste idioma: Buscapé, B2W, Olist e UTL *Movies*. Essas bases são compostas por avaliações de usuários reais da *web* sobre produtos e serviços. Os resultados encontrados nessa pesquisa mostraram que nem todos os algoritmos de aprendizagem de máquina sofrem impacto diante da mudança na técnica de vetorização, porém quando pelo menos duas técnicas são combinadas a partir da concatenação entre seus vetores de pesos, é possível obter melhoria na performance de algoritmos comumente utilizados na área de aprendizagem de máquina, como a MLP e o XGBoost.

Palavras-chaves: análise de sentimentos; word embeddings; aprendizagem de máquina; aprendizagem profunda; classificação de múltiplas classes; meta-embeddings.

ABSTRACT

Sentiment Sentiment Analysis is a field of research that aims to find the polarity of a sentiment in a text, in order to identify whether the information is positive, negative or neutral, among other forms of classification. With The expressive amount of textual information that is spread around the web every day, the process of automatic sentiment analysis becomes even more necessary. For contents related to products and services reviews, the detection of feelings is of great importance, since understanding the message that consumers want to pass about a product is essential for companies due to several factors, including marketin campaigns and the improviment of the relationship between companies and their customers. In this scenario, the study of ways to improve the representation of textual information in order to be processed by machine learning models is extremely important to contribute with the increasement of performance in the sentiment classification tasks. Thus, the present work carries out an experimental study of the behavior of different text vectorization techniques, with focus on embeddings: representative vectors composed by real values and capable of preserve syntactic and semantic information from words. For this, different types of embeddings vectors and three types of combination for these vectors are evaluated, which are used in the classification process of five different machine learning algorithms. Furthermore, a deep learning model is also used where the embedding step is performed by the embedding layer. In order to contribute with Sentiment Analysis of texts in Portuguese-BR, four databases in this language were studied: Buscapé, B2W, Olist and UTL Movies. These bases are composed by reviews from real web users about products and services. The results found in this research showed that not all machine learning algorithms are impacted by the change in the embedding technique, but when at least two techniques are combined by concatenation of their vectors, it brought an improvement in the performance of algorithms commonly used in machine learning problems, such as MLP and XGBoost.

Keywords: sentiment analysis; word embeddings; machine learning; deep learning; muticlass classification; meta-embeddings.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AM	Aprendizagem de Máquina
AS	Análise de Sentimentos
CGU	Conteúdo Gerado por Usuário
CNN	*Convolucional Neural Network*
IA	Inteligência Artificial
LSTM	*Long Short-Term Memory*
MLP	*Multilayer Perceptron*
PLN	Processamento de Linguagem Natural
POS	*Part of Speech Tagging*
RNA	Rede Neural Artificial
RNR	Rede Neural Recorrente
SVM	*Supported Vector Machine*

SUMÁRIO

1 INTRODUÇÃO

1.1 MOTIVAÇÃO

No últimos anos, o crescimento significativo de conteúdos textuais gerados por usuários da internet mudou a forma de socialização e a busca pelo conhecimento dos indivíduos. Além dos conteúdos midiáticos e comerciais, os usuários da *web* expandiram seu papel na navegação e na pesquisa de informações e passaram a compartilhar seus conhecimentos, experiências, críticas e opiniões online, através de blogs pessoais, redes sociais, sites de compra, comentários de artigos e outros meios (ARAUJO, 2014).

Diante disso, é fácil notar o quanto dados textuais são uma fonte de informação muito rica, que fornecem a oportunidade de explorar diversas aplicações que muitas vezes não conseguem ser realizadas apenas com dados qualitativos (TAN; MUI; TERRACE, 2000). Esses dados são obtidos principalmente através de avaliações de produtos e serviços e postagens em redes sociais, o que é bastante útil em aplicações relacionadas à inteligência de negócios, definições de público-alvo, predição dos resultados de eleições, análise de fenômenos sociolinguísticos, dentre inúmeras outras possibilidades (BERTAGLIA; NUNES, 2017).

As avaliações de produtos e serviços, particularmente, permitem a aplicação de diferentes tipos de estratégias que podem ser tomadas por lojas e prestadoras de serviço. Compreender melhor os diferentes tipos de perfis dos seus clientes, por exemplo, viabiliza formas de atendimento mais customizadas, o que tende a elevar a satisfação dos clientes e consequentemente contribui tanto para sua retenção ao consumo daquela loja e/ou serviço, bem como à recomendação espontânea para outras pessoas, promovendo o crescimento e aumento da rentabilidade do negócio. Assim, seja com o objetivo de melhorar o atendimento ou de traçar estratégias de *marketing* mais complexas para determinado público, o principal recurso utilizado é a opinião subjetiva do público em geral. Esse tipo de informação gerada de forma espontânea pelas pessoas em sites, redes sociais e demais formas de comunicação online é definido como Conteúdo Gerado por Usuário (CGU) (KRUMM; DAVIES; NARAYANASWAMI, 2008).

Para lidar com todo esse volume de informação textual disponível e a coloquialidade associada ao CGU, diversos estudos na área de Ciência da Computação vêm sendo desenvolvidos com o intuito de construir novas formas ou melhorar as formas já existentes de métodos automáticos para interpretação e extração desse tipo de conteúdo (EL-KASSAS et al., 2021). Uma das formas de processamento automático de informações textuais é realizada através de uma área

do conhecimento formada pela junção da Ciência da Computação (particularmente através de técnicas de Inteligência Artificial (IA)) com a linguística, denominada Processamento de Linguagem Natural (PLN). O principal objetivo do PLN é projetar e construir sistemas inteligentes capazes de analisar a linguagem natural (falada e escrita por humanos) em qualquer idioma (NEY, 2019) e uma das suas principais aplicações consiste na classificação de sentimentos e opiniões em diferentes polaridades, que é denominada como Análise de Sentimentos (AIRES et al., 2018). Uma das principais dificuldades nesse processo de análise automática, consiste na transformação dos textos em informações capazes de serem processadas por máquinas. Este processo, é extremamente necessário para qualquer problema de PLN que seja resolvido através de IAs.

Apesar da quantidade significativa de estudos relacionados ao PLN, a grande maioria deles foca especificamente no Inglês como idioma de estudo, o que tecnicamente faz sentido, uma vez que esta língua é a mais utilizada em todo mundo, e adotada como padrão no meio acadêmico (INGLESA, 2020). Entretanto, captar e processar informações de forma automática para outros idiomas também é de extrema importância, uma vez que cada língua possui suas particularidades e diferentes níveis de complexidade, o que muitas vezes é necessário para a evolução e otimização das técnicas já existentes para o Inglês. Essa complexidade ocorre, por exemplo, na língua portuguesa, que está entre os 10 idiomas mais falados do mundo, sendo a língua oficial de 9 países [1] (NOVO, 2021) e estava em 5º lugar nos idiomas mais utilizados mundialmente na *web* em Março de 2020 [2]. Com sua vasta riqueza vocabular, um dos motivos para a complexidade desta língua é a significativa contribuição de regionalismos, resultantes da grande extensão territorial de países como o Brasil, onde o idioma é utilizado majoritariamente.

Desse modo, a língua portuguesa acaba se tornando um desafio para muitas técnicas de PLN. Apesar disso, felizmente já existem trabalhos com contribuições significativas como o de (SOUZA et al., 2016), que realizou uma revisão sistemática das principais contribuições em estado da arte para o Português, e o de (PEREIRA, 2021), que pesquisou e compilou diversos trabalhos envolvendo abordagens de técnicas de PLN específicas para a língua portuguesa até o início dos anos 2020. Já trabalhos como o de (NASCIMENTO, 2019) e de (KAIBI; NFAOUI; SATORI, 2019), buscavam, respectivamente, formas de combinação de técnicas de algoritmos de classificação e comparação entre diferentes técnicas de vetorização de palavras para textos

[1] A saber: Brasil, Portugal, Angola, Moçambique, Cabo Verde, Timor-Leste, Guiné-Bissau, Guiné Equatorial, São Tomé e Príncipe

[2] <https://www.internetworldstats.com/stats7.htm>

em Português extraídos da rede social Twitter.

O presente trabalho, por sua vez, busca contribuir com pesquisas relacionadas à Análise de Sentimentos aplicada à língua portuguesa (em particular o Português brasileiro), através de experimentos que serão realizados em cinco bases de dados, todas contendo avaliações de consumidores sobre produtos e serviços, visando responder às seguintes questões de pesquisa: (1) dentre os diferentes tipos de algoritmos utilizados em Aprendizagem de Máquina, dos mais tradicionais aos que utilizam aprendizado profundo, há algum que se destaque para as bases de dados utilizadas? (2) existe a melhor combinação Algoritmo de Classificação *versus* técnica de vetorização de palavras para o Português na área de avaliações de produtos? (3) a combinação entre diferentes técnicas de vetorização resulta em melhor desempenho nos resultados da classificação? (4) dentre os métodos de combinação utilizados, combinações mais complexas entregam melhores resultados quando comparadas a uma combinação que realiza uma simples concatenação entre duas formas de vetorização distintas?

1.2 OBJETIVOS

1.2.1 Objetivo Geral

Este trabalho busca realizar um estudo comparativo entre diferentes técnicas de vetorização de texto, que serão avaliadas em algoritmos comumente utilizados em problemas de Análise de Sentimentos, desde os mais tradicionais, baseados em métodos clássicos, passando pelos que utilizam métodos de árvore, até os de aprendizagem profunda. As bases de dados utilizadas são formadas por avaliações de produtos e serviços de grandes *Market Places*, todas contendo avaliações escritas em Português Brasileiro. A ideia principal é mostrar como diferentes técnicas de vetorização podem auxiliar na melhoria da precisão em diferentes algoritmos classificadores, considerando o cenário de classificação não-binária, uma vez que os exemplos são rotulados como positivo, negativo e neutro.

Além disso, também serão apresentados experimentos que combinam diferentes técnicas de vetorização, com o intuito de verificar se a combinação de diferentes métodos pode auxiliar na classificação para múltiplas classes no caso de avaliações em Português-BR.

1.2.2 Objetivos Específicos

- Reunir diferentes bases de dados em português brasileiro relacionadas à avaliações de produtos e serviços;
- Verificar se existe a melhor combinação de algoritmo classificador com técnica de vetorização no conjunto de dados utilizados;
- Verificar se alguma das técnicas de vetorização utilizadas se destaca para diferentes algoritmos em relação às demais;
- Avaliar diferentes formas de combinação de vetorização de 3 (três) formas distintas e analisar o impacto dessas combinações nos modelos de classificação;
- Contribuir academicamente para a literatura relacionada à Análise de Sentimentos aplicada à língua portuguesa em problemas de classificação de textos com múltiplas classes.

1.3 ESTRUTURA DA DISSERTAÇÃO

A presente dissertação está organizada em 6 capítulos, além das Referências, que contribuíram para o entendimento e execução do trabalho:

- **Introdução**: apresenta o contexto de Análise de Sentimentos de forma geral, bem como a motivação e os objetivos desta dissertação;
- **Fundamentação Teórica**: explana os conceitos e técnicas fundamentais para o embasamento teórico da dissertação;
- **Trabalhos Relacionados**: lista as contribuições mais relevantes relacionadas ao tema da presente pesquisa.
- **Metodologia**: descreve as arquiteturas experimentais utilizadas, bem como as bases de dados e as etapas que envolvem o tratamento dessas bases, além das configurações dos algoritmos classificadores abordados.
- **Experimentos e Resultados**: apresenta a descrição dos experimentos realizados e discute sobre os resultados obtidos;

- **Conclusões e Trabalhos Futuros**: realiza as considerações finais sobre o trabalhos e possibilidades de contribuições futuras;

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta uma revisão de literatura sobre alguns fundamentos teóricos, necessários para realização do presente trabalho. Aqui serão descritos os conceitos de Processamento de Linguagem Natural, Análise de Sentimentos, técnicas de pré-processamento de textos, técnicas de mapeamento de dados textuais vetores, de aprendizagem de máquina e aprendizado profundo, além das métricas de avaliação e testes estatísticos utilizados.

2.1 PROCESSAMENTO DE LINGUAGEM NATURAL

Nas últimas duas décadas, a produção de dados textuais vêm crescendo de forma extremamente acelerada, principalmente em decorrência da expansão significativa da quantidade de serviços disponíveis online (redes sociais, sites de compra, formulários online, publicações de artigos científicos, etc) (NASEEM et al., 2020). A forma como essas informações se apresentam é facilmente compreendida pelos humanos, porém esses dados possuem grande potencial para aplicações em diferentes áreas do conhecimento, como na indústria e no comércio.

Essas e outras aplicações podem ser realizadas a partir de diversas áreas da Ciência da Computação, em especial na de Inteligência Artificial (IA), que estabelece a simulação da inteligência humana a partir de máquinas. Com isso, foi desenvolvido um campo de pesquisa específico capaz de relacionar a ciência da computação e a IA com a linguística. Essa área de pesquisa é conhecida por Processamento de Linguagem Natural PLN, que possui como principal objetivo estabelecer interações e traduções entre a linguagem computacional (das máquinas) e a linguagem humana (natural) (KUMAR, 2011).

Dessa forma, as técnicas desenvolvidas na área de PLN buscam fazer com que computadores consigam entender e capturar aspectos da linguagem humana e, com isso, serem capazes de examinar uma quantidade expressiva de informações textuais não estruturadas e extrair informações de valor a partir delas (NASEEM et al., 2020). A grande dificuldade para uma IA conseguir compreender a linguagem humana está diretamente associada ao caráter subjetivo da comunicação, uma vez que o ato de se comunicar não consiste apenas em utilizar palavras de forma lógica: há sempre um contexto associado ao que se fala (lê, escuta) e também à forma como se fala (JOHNSON, 2021). Outro aspecto complexo da linguagem natural é a resposta para a pergunta: como o ser humano consegue compreender o significado das palavras?

Basicamente, a compreensão vem a partir da experiência: quando uma pessoa se depara com uma palavra nova, normalmente a primeira ação é buscar seu significado (com alguém ou em algum lugar) e depois incorporá-la ao seu vocabulário através da repetição (MITRANI, 2019).

Apesar da complexidade da linguagem natural, toda linguagem apresenta regras e aspectos estruturais. A parte da linguística que estuda essa estrutura e organização das palavras em uma oração é chamada sintaxe, já a área que estuda o significado das palavras e como elas se relacionam é a semântica. Uma vez que a forma de aprendizagem de uma IA é estruturada e baseada em regras, uma das funções do PLN consiste em estabelecer técnicas necessárias para facilitar essa forma de aprendizagem (HOSSIAN, 2018).

Estruturar a linguagem natural para que ela seja processada por uma IA é um dos componentes do PLN, a outra componente está associada a geração de palavras dado um contexto. Do ponto de vista prático, inúmeras são as formas de aplicações das técnicas de PLN: classificação de textos, análise de sentimentos, filtros de *spam*, respostas automáticas, traduções automáticas, etc (CENTER, 2021). A Tabela 1 traz um resumo das principais aplicações realizadas pelo PLN junto com alguns exemplos de utilização.

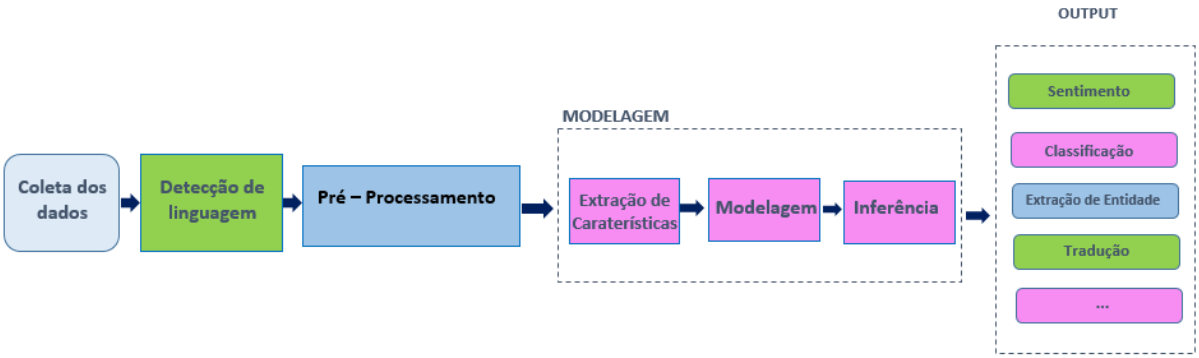
Tabela 1 – Principais componentes do PLN

COMPONENTE	DEFINIÇÃO	EXEMPLO DE APLICAÇÃO
Classificação	Determinar o tipo ou classe de uma unidade ou texto	Análise de Sentimentos; Categorização de Notícias, Filtragem de *spam*
Extração	Analisar o conteúdo de um texto de modo a extrair componentes textuais e relações entre as palavras	Reconhecimento de entidade (identificar pessoas, lugares, organizações, etc); Extração de tópicos/conteúdos de temas específicos
Geração	Gerar uma sequencia de *tokens* (ou palavras) baseadas em um contexto (texto ou discurso)	Tradução por Aprendizagem de Máquina, Sumarização de documentos

Fonte: (CENTER, 2021)

Em relação ao conjunto de técnicas utilizadas no processo de PLN, o estágio responsável por realizar a estruturação dos dados é comumente chamado de *pré-processamento*, que é composto por várias etapas, tais como: tokenização, *stemming*, remoção de *Stop Words*, dentre outras, que serão detalhadas mais a frente no texto. As principais etapas do PLN são mostradas no fluxograma da Figura 1 a seguir.

Figura 1 – Principais Etapas do PLN.



Fonte: (CENTER, 2021) (adaptado)

Embora o aprendizado de máquina seja hoje amplamente utilizado para modelar a linguagem humana, há também a necessidade de compreensão sintática e semântica das informações, além do domínio onde elas estão inseridas. Nem sempre essas considerações são levadas em conta em muitas abordagens de Aprendizagem de Máquina (AM). O PLN é importante porque ajuda a resolver a ambiguidade na linguagem e adiciona uma estrutura numérica útil aos dados para muitas aplicações, como reconhecimento de fala ou análise de textos (HOSSIAN, 2018), em particular o sentimento que determinado texto busca transmitir. A seção a seguir discorre sobre a importância e utilidade dessa aplicação do PLN, denominada Análise de Sentimentos.

2.2 ANÁLISE DE SENTIMENTOS

Todo tipo de informação textual pode ser caracterizada basicamente de duas formas: fatos e opiniões. Os fatos são informações objetivas e concretas, normalmente imutáveis em qualquer situação. Já as opiniões costumam ser expressões subjetivas e associadas a sentimentos(LIU, 2010), e podem ser extremamente mutáveis - a opinião de uma pessoa sobre um fato pode mudar de um dia para o outro, porém o fato segue sendo o mesmo.

Apesar de sua subjetividade e relativismo, a busca por entender a opinião de um ou mais grupos de pessoas tem extrema utilidade sob diversos aspectos, inclusive comercialmente, uma vez que entender as opiniões de clientes, por exemplo, pode auxiliar uma empresa a realizar novos tipos de ações de *Marketing*, melhorar sua qualidade de serviço, sua relação com seus consumidores, dentre outras possíveis estratégias de mercado (PEREIRA, 2021). Essa busca passou a ter um crescimento significativo a partir dos anos 2000, diante da popularização da internet e subsequente transformação nas formas de comunicação, uma vez que a *web* deixou

de ser apenas um local para transações eletrônicas, trocas de *e-mails* e busca de informações (KANSAON; BRANDAO; PINTO, 2019), tornando-se um ambiente repleto de contextos extremamente subjetivos, a partir do crescimento expressivo de fóruns, microblogs e redes sociais (AVANCO; NUNES, 2014).

Do ponto de vista social, a ideia de saber a opinião de outras pessoas pode ser bastante útil no processo de tomada de decisão, algo que já acontecia muito antes da explosão da *Web* nos anos 2000 (SILVA; COLETTA; HRUSCHKA, 2016). Através de pesquisas de opiniões como as que são realizadas em censos demográficos, por exemplo, é possível esclarecer a situação e o momento vivenciado em um país sob diversos aspectos: econômicos, sociais, educacionais, etc; além de auxiliar em todo processo de distribuição de recursos para políticas públicas, em ações de planejamento e na priorização das demandas de um país.

A principal forma de compreender as opiniões em grandes volumes de dados é realizada através da Análise de Sentimentos (AS), uma das áreas de pesquisa mais ativas no contexto de PLN desde do início dos anos 2000 (AGARWALL et al., 2020). Nessas últimas décadas de estudos sobre o tema, diversas técnicas foram desenvolvidas, tais como: extração de características, detecção de emoções e classificação de sentimentos (MEDHAT; HASSAN; KORASHY, 2014).

Conceitualmente, o significado de analisar um sentimento pode possuir diferentes conotações, dependendo da área do conhecimento na qual está inserido, uma vez que a própria definição de sentimento em si mesma é tão subjetiva que acaba tornando difícil caracterizá-la objetivamente (MEJOVA, 2009). Para ilustrar esta complexidade, (PANG; LEE, 2008) mostra algumas definições dos sinônimos de sentimento que são definidos pelo Dicionário Online de Merriam-Websters (que o define como: "*um julgamento que se considera verdadeiro*"), tais como:

- *Opinião*: corresponde a uma conclusão pensada, mas aberta a modificações;
- *Ponto de vista*: sugere uma opinião subjetiva;
- *Crença*: aceitação deliberada e consentimento intelectual;
- *Convicção*: se aplica a uma crença firme e séria. Exemplo: a convicção de que a vida animal é tão sagrada quanto a humana.

Apesar da subjetividade contida nas definições acima, para a ciência da computação é possível definir a análise de sentimentos de modo mais objetivo, o que também pode ser realizado

de diversas formas, que convergem para a ideia central de que essa análise representa uma abordagem de análise textual que busca detectar o significado principal ou a polaridade de um texto automaticamente (CIRQUEIRA et al., 2018). A história do termo Análise de Sentimentos é paralela à da Mineração de Opiniões, por isso a associação do termo *opinião* como sinônimo. O uso do termo *sentimento* para designar a análise e avaliação automática de um texto, foi oficialmente adotado a partir dos artigos de (DAS; CHEN, 2001) e (TONG, 2001) em um trabalho sobre o impacto do sentimento no *Marketing*, em 2001 (PANG; LEE, 2008).

Do ponto de vista prático, essa análise pode ser realizada em pelo menos 3 níveis de granularidade, a saber:

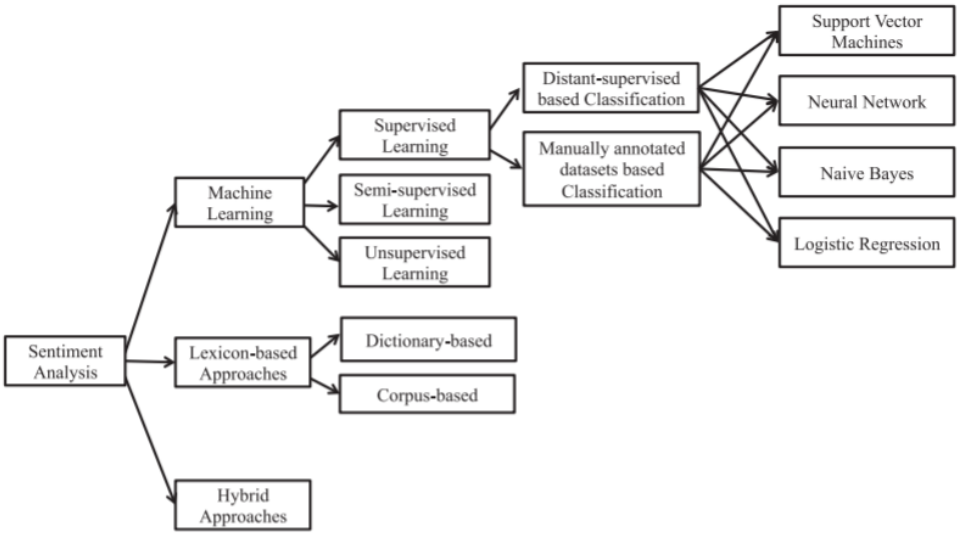
1. **Documento**: esse tipo de análise busca classificar o contexto geral, que normalmente é centrado em uma entidade específica (uma resenha de um livro, por exemplo), e avaliar o teor daquele conteúdo de modo a classificá-lo como negativo, positivo ou neutro, além de outras formas de classificação (PANG; LEE, 2008).

2. **Sentença**: essa análise pode considerar a classificação de partes de um mesmo documento sendo avaliadas individualmente, onde cada sentença pode ter uma polaridade distinta (SILVA; COLETTA; HRUSCHKA, 2016).

3. **Entidade ou palavra**: esse tipo de análise leva em conta as entidades ou palavras-chaves presentes no texto (SILVA; COLETTA; HRUSCHKA, 2016). Por exemplo, na frase "este filme é bom, mas a atuação daquele ator foi péssima", pode-se extrair duas entidades classificadas de formas distintas: "o filme"está associado a um comentário positivo, enquanto que "o ator"está sendo avaliado de forma negativa.

Em relação às abordagens de classificação que podem ser utilizadas, também é possível segmentá-las em 3 grandes grupos: utilização de aprendizagem de máquina, métodos baseados em léxicos e métodos híbridos (MEDHAT; HASSAN; KORASHY, 2014). A Figura 2 mostra uma visão geral dessas diferentes abordagens e as formas de realizá-las.

Figura 2 – Visão geral das abordagens utilizadas na Análise de Sentimentos.



Fonte: (SILVA; COLETTA; HRUSCHKA, 2016)

A abordagem léxica depende da disponibilidade de um léxico de sentimentos, que consiste em uma coleção de palavras cujos sentimentos são conhecidos previamente. Essa abordagem pode ser realizada de duas formas: (1) baseada em dicionário, que utiliza dicionários como fonte de definição das palavras; (2) baseada em um *corpus*, que utiliza métodos estatísticos e semânticos para encontrar a polaridade do sentimento (SILVA; COLETTA; HRUSCHKA, 2016). Já a classificação por aprendizagem de máquina consiste na aplicação de algoritmos de IA em um conjunto de exemplos (que podem ou não ser rotulados) e realiza a classificação de forma probabilística, como será explicado em uma das seções adiante. Finalmente, a abordagem híbrida busca combinar as duas abordagens descritas anteriormente, com os léxicos de sentimento desempenhando um papel fundamental na maioria dos métodos (MEDHAT; HASSAN; KORASHY, 2014).

O estudo aqui apresentado trata da classificação de sentimentos utilizando algorítimos de aprendizagem de máquina (clássicos e de aprendizado profundo), no nível de documentos, uma vez que analisa comentários sobre avaliações de produtos e serviços para diferentes bases de dados. A seção a seguir mostra alguns procedimentos necessários para o tratamento dos textos quando é adotada a abordagem de aprendizagem de máquina para realizar a classificação.

2.3 PRÉ-PROCESSAMENTO DE TEXTO

Uma das primeiras etapas da classificação de sentimentos é o pré-processamento dos dados. Esta etapa é fundamental para tratar os dados coletados (independente da forma de coleta) de modo a torná-los normalizados e estruturados o suficiente para a fase de extração de características (BATISTA, 2003) e envolve diversos passos, tais como:

- **Limpeza dos Dados**

  Considerando o fato de que grande parte das bases de dados utilizadas em diversos problemas de AS são obtidas a partir de informações encontradas na *web*, é muito comum que os textos contidos nelas estejam fora da linguagem formal, ou seja, apresentem gírias, abreviações, erros de digitação, etc. A limpeza de dados consiste na remoção de elementos desnecessários e/ou normalização de palavras digitadas incorretamente, de modo a enxugar o texto mantendo apenas as informações mais significativas do conteúdo. Assim, pontuações e caracteres especiais são removidos, bem como números, acentos, URLs, espaços em brancos e quebras de linhas. Além disso, todas as palavras passam a ser escritas em letras minúsculas (ANGIANI et al., 2016). Dessa forma, tomando como exemplo a frase:

  *Estou contente com a compra!!! *–* A entrega foi rápida, chegou em 10 dias, o único problema com as @Americanas é que se houver troca ou devolução do produto, o consumidor tem problemas com a espera :(*

  Após as remoções mencionadas, ela seria transformada em:

  estou contente com a compra a entrega foi rapida chegou em dias o unico problema com as americanas e que se houver troca ou devolucao do produto o consumidor tem problemas com a espera

- **Tokenização**

  Essa é uma das primeiras etapas realizadas durante o pré-processamento de texto e consiste na ideia de dividir a sentença ou documento menores partes, denominadas *tokens* (HOSSIAN, 2018). Assim, a sentença utilizada no exemplo anterior, após o processo de tokenização, passaria a ser um vetor de palavras da seguinte forma:

["estou","contente","com","a", "compra", "a", "entrega", "foi", "rapida", "chegou", "em", "dia", "o", "unico", "problema", "com", "as", "americanas", "e", "que", "se", "houver", "troca", "ou", "devolucao", "do", "produto", "o", "consumidor", "tem", "problemas", "com","a", "espera"]

A forma de tokenização utilizada no exemplo acima (e nos experimentos realizados na pesquisa) é denominada *unigram*, uma vez que divide a sentença em unidades. Porém a divisão do texto também pode ser realizada por pares de palavras, conhecidos como bi-*grams*. Genericamente, esses pequenos agrupamentos de palavras são denominados N-*grams*, sendo N a quantidade de palavras (e formas de combinação) presentes em cada *token*.

- ***Stemming***

  As técnicas de *stemming* colocam variações de uma mesma palavra em uma só representação, o que diminui a entropia da frase e aumenta a relevância do conceito das palavras no texto (ANGIANI et al., 2016). Isso significa que o *stemming* permite a redução das diferentes classes de palavras (substantivos, verbos e advérbios) à sua raiz. Assim, palavras como "alegre", "alegria", "alegrar"e "alegremente",serão todas consideradas com a mesma representação ("alegr") no vocabulário de palavras.

- **Remoção de *StopWords***

  Nesta etapa do pré-processamento é realiza a eliminação de palavras irrelevantes para compressão do sentido total da frase. Essas palavras são, por exemplo, pronomes, artigos, conjunções, etc., e removê-las tende a melhorar a precisão dos algoritmos classificadores (ANGIANI et al., 2016). Considerando a frase exemplificada anteriormente, após a aplicação do *stemming* e da remoção de *stopwords*, ela seria transformada em:

  content compra entrega rapida chegou dia unico problema americana troca devolucao produto consumidor problema espera

  É fácil notar que o exemplo inicial, após ser submetido às principais etapas do pré-processamento de texto, torna-se mais simples e consegue ao mesmo tempo reter o significado original da frase. Essa simplificação do texto é essencial para a etapa de representação de palavras, que será explanada na seção a seguir.

2.4 MÉTODOS DE EXTRAÇÃO DE CARACTERÍSTICAS

Após o pré-processamento dos dados, a próxima etapa do fluxo da classificação na análise de sentimentos (e problemas de classificação em geral) é a extração de características, que pode ser definida como uma transformação dos dados para redução de dimensionalidade (KHALID; KHALIL; NASREEN, 2014). Em outras palavras, a principal ideia é selecionar as informações que serão mais relevantes para o aprendizado dos algoritmos. Essas características podem ser obtidas a partir de variáveis numéricas e/ou categóricas. Entretanto, algoritmos de AM não conseguem extrair informações de dados textuais/categóricos em forma de texto, uma vez que trabalham apenas com *inputs* numéricos. Dessa forma, as informações textuais de um conjunto de dados precisam passar por uma etapa de codificação que mantenha suas características originais e também consiga ser interpretada pelos algoritmos de AM da melhor forma (MEDHAT; HASSAN; KORASHY, 2014). Assim, esta seção visa descrever as principais abordagens utilizadas no que diz respeito à extração de características textuais e codificação de texto.

2.4.1 Bag of Words

Esse método realiza a representação das palavras de um documento levando em conta apenas a presença (ou ausência) daquela palavra em todo documento de texto (PANTOLA, 2018). Para compreender melhor o funcionamento desta técnica, será utilizado como exemplo uma pequena base de dados formada apenas por três avaliações de produtos e seus respectivos vetores de palavras após a etapa de pré-processamento, que são mostrados na Tabela 2.

Tabela 2 – Exemplo de uma Base com Avaliações de Produtos

Avaliação	Vetor de Palavras da Sentença após Pré-processamento
Excelente produto! A entrega chegou no prazo.	[excelente, produto, entrega, chegou, prazo]
O produto chegou fora do prazo, mas gostei da compra.	[produto, chegou, fora, prazo, gostei, compra]
Entrega rápida, gostei do custo benefício do produto	[entrega, rapida, gostei, custo, beneficio, produto]

Fonte: a autora (2022)

O vocabulário total do documento, em ordem alfabética, seria dado pelo vetor de palavras a seguir:

[benefício, chegou, compra, custo, dentro, entrega, excelente, fora, gostei, prazo, produto, rápida]

Dessa forma, considerando, por exemplo, o valor 1 para a presença de uma palavra do vocabulário naquela sentença e 0 sua ausência, as frases da Tabela 2 seriam transformada nos vetores mostrados na Tabela 3:

Tabela 3 – Exemplificação da Técnica *Bag Of Words*

Avaliação	Vetor de Palavras da Sentença	Transformação após aplicação do *Bag of Words*
Excelente produto! A entrega chegou no prazo.	[excelente,produto, entrega, chegou, prazo]	[0,1,0,0,0,1,1,0,0,1,1,0]
O produto chegou fora do prazo, mas gostei da compra.	[produto, chegou, fora, prazo, gostei, compra]	[0,1,1,0,0,0,0,1,1,1,1,0]
Entrega rápida, gostei do custo benefício do produto	[entrega, rapida, gostei, custo, beneficio, produto]	[1,0,0,1,0,1,0,0,1,0,1,1]

Fonte: a autora (2022)

Ou seja, a transformação vetorial após o *Bag of Words* irá levar cada exemplo da base de dados a um vetor de tamanho correspondente ao total de palavras do vocabulário da base. Nesse exemplo, para fins de simplificação, foram considerados pesos binários na representação das palavras, embora a técnica BoW não diga respeito ao peso dos termos utilizados e sim à representação dos mesmos. Matematicamente, essa transformação costuma ser representada no formato de uma matriz esparsa de dimensões (n, p) sendo n a quantidade de exemplos e p o tamanho total do vocabulário. É fácil notar que essa transformação não é muito viável quando o vocabulário total do documento possui volume significativo, uma vez que, além de não levar em conta a ordem ou o sentido das palavras no texto, o aumento do vocabulário implica no aumento da dimensionalidade dos dados, o que costuma acarretar na degradação do tempo de execução dos algoritmos que utilizarem essa técnica para a classificação.

2.4.2 Term Frequency-Inverse Document Frequency (TFIDF)

O estratégia TFIDF trabalha em cima da frequência dos vocabulários de um documento, fornecendo menores pesos para palavras que se repetem com elevada frequência e que provavelmente são pouco relevantes para a contribuição do significado geral do texto - como é o caso de artigos, preposições e conjunções. Ela atribui maiores pesos a palavras menos frequentes, que normalmente são mais importantes, como ocorre com adjetivos e advérbios (JHA, 2021). O TF-IDF envolve dois conceitos, a saber:

- **Frequência de Termo** (em Inglês, *Term Frequency*): corresponde à relação entre a quantidade de vezes que este termo aparece em um documento e o tamanho total do vocabulário deste documento (PANTOLA, 2018). O que matematicamente equivale a:

$$TF = \frac{num.\ de\ ocorrencias\ de\ uma\ palavra\ no\ documento}{num.\ total\ de\ palavras\ do\ documento} \tag{2.1}$$

Se a avaliação *"Entrega rápida, gostei do custo benefício do produto"* fosse considerada sem a remoção de *StopWords*, então o termo de maior frequência seria a preposição *do*, pois aparece duas vezes na frase, enquanto as demais palavras são unitárias.

- **Frequência Inversa de Documento** (*Inverse Document Frequency*): este conceito está relacionado ao grau de importância de uma palavra em um documento e baseia-se no princípio de que palavras que aparecem em baixa frequência podem ter conteúdos mais significativos (PANTOLA, 2018). Para uma palavra *w* presente em um conjunto de documentos, essa frequência pode ser calculada como:

$$IDF = 1 + log(\frac{num.\ total\ de\ documentos}{num.\ de\ documentos\ onde\ a\ palavra\ w\ ocorre}) \tag{2.2}$$

Considerando cada avaliação da Tabela 2 como um documento, então a palavra mais frequente nesse conjunto de documentos seria *produto*, uma vez que aparece em todas as avaliações.

A partir das definições de TF e IDF, a transformação de uma palavra via TF-IDF pode então ser calculada da seguinte forma:

$$TF - IDF = TF * IDF \tag{2.3}$$

Assim, a palavra *"produto"*, presente na avaliação citada no exemplo anterior e cujo vetor de palavras é mostrado na Tabela 2, seria representada através da técnica TF-IDF como:

$$tfidf(produto) = (\frac{1}{5}) * (1 + log(\frac{3}{3})) = 1,2 \tag{2.4}$$

Em bibliotecas de programação utilizadas para tratamento de dados, como é o caso do *Scikit-learn* [1], há mais de uma forma de cálculo do TF-IDF, que é realizado pela função *TfidfVectorizer*. A forma padrão de cálculo da representação de um termo t realizada por esta função é dada por:

$$tfidf(t) = tf(t,d) * (log(\frac{1+n}{1+df(t)}) + 1) \tag{2.5}$$

[1] <https://scikit-learn.org/stable/index.html>

Onde $tf(t, d)$ é o número de ocorrências do termo t em um documento d, n a quantidade total de documentos e $df(t)$ o número de documentos onde o termo i aparece (UNNIKRISHNAN, 2021). Aplicando a fórmula acima na palavra *produto* calculada no último exemplo, seu valor de representação não normalizado seria igual a 1.

A técnica TF-IDF, apesar de conseguir quantificar a importância das palavras em um documento e ser amplamente utilizada em diversas aplicações, como extração de palavras-chaves e sistemas de recuperação de informação (JHA, 2021), não considera outras sutilezas textuais como o contexto em que uma palavra se encontra ou sua relação com as demais palavras de uma sentença. Diante disso, técnicas mais sofisticadas surgiram para contornar esse problema, como é o caso das que serão descritas na seção a seguir.

2.4.3 Word Embedding

Word Embedding, ou simplesmente *embeddings*, é um método de extração de características utilizada em inúmeros problemas de PLN, e consiste na representação das palavras através de um vetor numérico (composto por números reais), capaz de preservar informações sintáticas e semânticas necessárias para manter o sentido original da palavra (SILVA; CASELI, 2020). Essa característica costuma trazer ganhos de performance quando aplicada à modelos de AM em comparação a métodos baseados apenas na frequência dos termos, como o *Bag of Words* e o TF-IDF. Isso ocorre principalmente devido ao fato de que essas últimas técnicas consideram cada palavra de um documento como uma entidade individual, ignorando a relação entre as palavras e o contexto onde estão inseridas (JHA, 2021).

Os conceitos iniciais de *embedding* começaram a aparecer em trabalhos como o de (BENGIO et al., 2003), que desenvolveu um modelo probabilístico capaz de representar palavras de forma distribuída e com baixa dimensionalidade, o que é realizado a partir de vetores com uma dimensão fixa N, onde N costuma apresentar valores entre 50, e 500 (NASEEM et al., 2020). A ideia principal era observar a sequência de termos onde a palavra a ser codificada estava inserida, de modo que uma sequência de palavras desconhecidas teria alta probabilidade de ser representada de forma similar a palavras já conhecidas pelo modelo. Essa semelhança estava diretamente relacionada à distância entre as palavras da sequência (BENGIO et al., 2003). Por exemplo, considerando as frases:

O **gato** está dormindo na sala enquanto o **cachorro** está comendo na cozinha. (I)

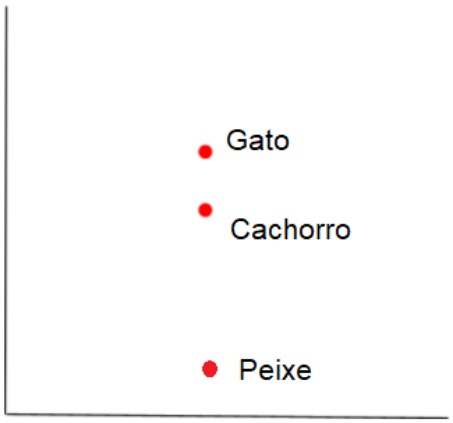
O **cachorro** está comendo na cozinha enquanto o **gato** está dormindo na sala. (II)

É fácil notar que ambas as frases são sintática e semanticamente equivalentes, ou seja, apesar da mudança na ordem das palavras, a frase II continua fazendo sentido e mantendo o mesmo significado que a I. *Word embeddings* conseguem estabelecer essa equivalência pois sua forma de representação vai além de um simples mapeamento de uma frase à uma representação numérica (BARLA, 2021). Matematicamente, o *embedding* representa uma função paramétrica de uma palavra, que pode ser expressa como:

$$f_\theta(W_n) = \theta_n \tag{2.6}$$

Onde W é a palavra a ser representada, θ o parâmetro da função e n o tamanho do vetor que representará W. Desse modo, palavras que apresentam similidade são representadas por parâmetros semelhantes. Assim, considerando que as palavras **gato**, **cachorro** e **peixe** fossem, por exemplo, representadas num espaço vetorial, seria possível notar a proximidade entre as duas primeiras, como mostra a Figura 3.

Figura 3 – Similaridade entre palavras em um espaço vetorial



Fonte: (BARLA, 2021) (adaptado)

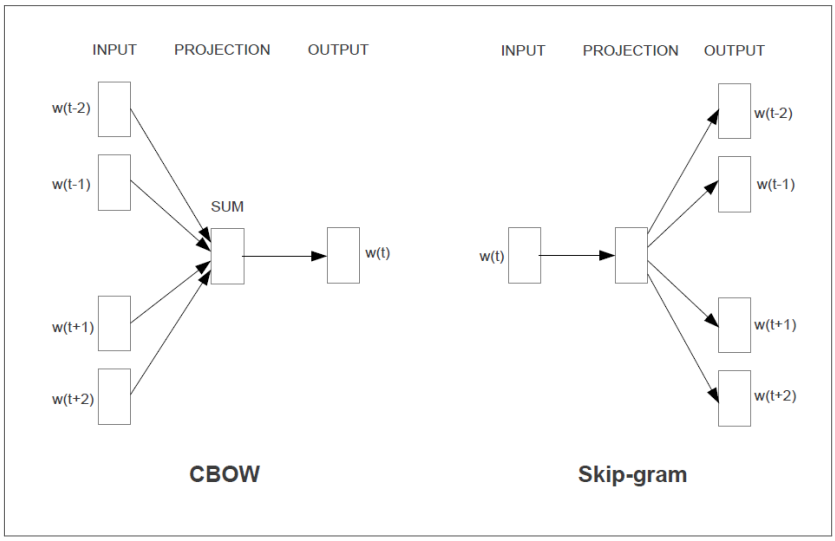
Já a palavra **peixe**, por sua vez, apesar de ser da mesma classe sintática (substantivo que nomeia um animal) que as outras duas palavras, no espaço vetorial de similaridade ela se apresenta mais distante do ponto de vista semântico, uma vez que esse animal possui características bem distintas dos demais. Dessa forma, o exemplo da Figura 3 mostra que a ideia principal dos modelos de *embedding* é a clusterização de informações similares de modo a estabelecer o relacionamento entre elas, o que normalmente é realizado a partir da utilização de redes neurais, como será visto mais adiante.

Desde o trabalho de (BENGIO et al., 2003) até o presente momento, diversos algoritmos de *embedding* foram desenvolvidos. Alguns deles se destacaram na literatura devido aos ganhos de performance mais expressivos em diversos problemas de AS. Neste trabalho foram analisados, de forma comparativa, vetores de palavras formados por quatro técnicas de *embedding* bastante conhecidas: *Word2Vec*, *Wang2Vec*, *Glove* e *FastText*, que serão descritas a seguir.

2.4.3.1 Word2Vec

Desenvolvido por (MIKOLOV et al., 2013b), esta técnica é composta por uma rede neural simples, com apenas duas camadas escondidas que são usadas para criar o vetor representativo de cada palavra. Esses vetores podem ser obtidos a partir de duas técnicas: CBOW e *Skip-Gram*, ambas propostas em (MIKOLOV et al., 2013a), que serão brevemente descritas a seguir e cujas arquiteturas são mostradas na Figura 4.

Figura 4 – Estrutura das arquiteturas aplicadas às técnicas de *embedding*.



Fonte:(MIKOLOV et al., 2013a)

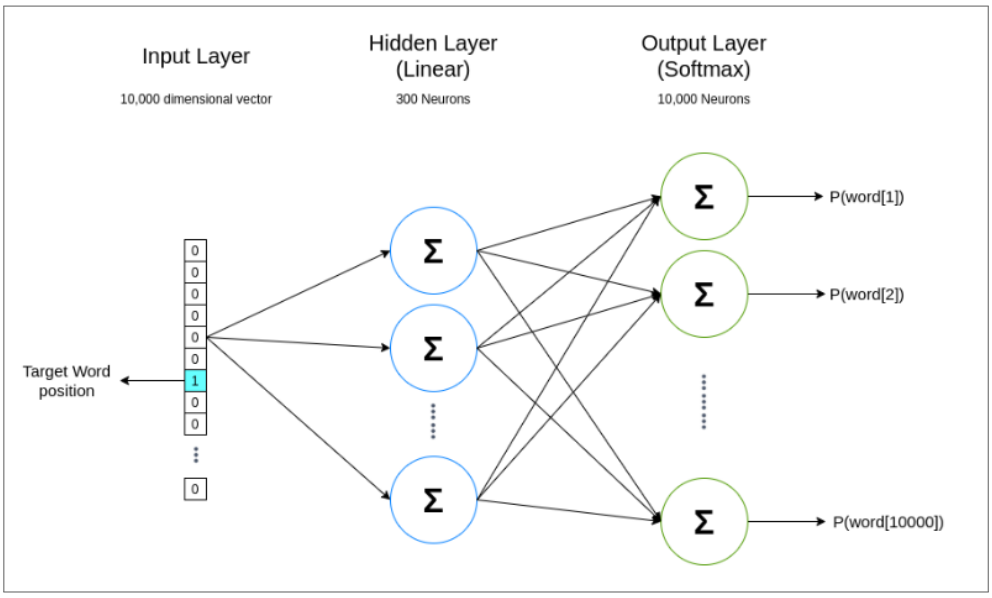
2.4.3.1.1 Skip-Gram

Essa arquitetura é projetada para obtenção do contexto em que uma palavra está inserida, tomando esta palavra como ponto de partida para dedução das demais (NASEEM et al., 2020). Na Figura 4 esta palavra é representada por $w[i]$, onde *i* é sua localização na sentença, enquanto a saída do modelo consiste na predição da probabilidade de encontrar as duas palavras anteriores e posteriores à palavra na posição *i* (JHA, 2021). A quantidade de palavras

que podem ser preditas é determinada por um parâmetro chamado "tamanho da janela". Essa "janela"se estende em ambas as direções a partir da palavra central.

Na prática, para determinar a probabilidade de uma palavra pertencente ao um vocabulário de tamanho M, a arquitetura da rede neural utilizada no *Skip-Gram* será semelhante à mostrada na Figura 5 onde é considerado um vocabulário com M = 10000 palavras.

Figura 5 – Exemplo de Rede Neural utilizada no *Skip-Gram*.



Fonte:(JHA, 2021)

Nessa rede neural, a camada de entrada é formada por um vetor binário de dimensão 10000, onde o valor 1 corresponde à posição da palavra-alvo a ser analisada e os demais valores são iguais a zero. Em seguida, existe a camada escondida (*Hidden Layer* na Figura 5), responsável por gerar uma matriz de pesos de dimensões M x N, onde M é o tamanho do vocabulário e N a quantidade de neurônios desta camada. Na rede neural da Figura 5, a camada escondida é formada por 300 neurônios, de modo que a matriz de pesos obtida nesta camada possui dimensão 10000x300. Finalmente, a camada de saída retorna a representação de cada palavra do vocabulário como um vetor de dimensão N formado pelos pesos calculados na camada escondida e que indica a similaridade entre cada palavra do vocabulário e a palavra-alvo utilizada na entrada da rede.

2.4.3.1.2 CBOW

CBOW corresponde é um acrônimo em Inglês para *Continuous Bag of Words*, e a ideia principal desta arquitetura é predizer uma palavra inserida em determinado contexto (sentença,

parágrafo, documento, etc) a partir das palavras vizinhas (NASEEM et al., 2020). Observando sua estrutura na Figura 4, é fácil notar que esta técnica se comporta como uma versão espelhada do *Skip Gram*. Ou seja, também corresponde a uma arquitetura composta por 3 camadas, porém sua primeira camada recebe as palavras ao redor da palavra a ser transformada, que é obtida na camada de saída. A segunda camada possui a mesma função que a descrita para o caso do *Skip Gram*: é responsável por atribuir os pesos que serão estimados na camada de saída, a diferença neste caso é que, devido às múltiplas entradas na primeira camada (correspondendo à quantidade de palavras pertencentes ao contexto da palavra a ser transformada em vetor), os pesos da camada escondida são obtidos a partir da média dessas informações de entrada. O vetor de saída também terá a dimensão N correspondente à quantidade de neurônios na camada escondida (JHA, 2021).

Em relação à performance, CBOW costuma apresentar menor tempo de treinamento em relação ao *Skip-Gram* e apresenta bons resultados na representação de palavras com elevada frequência vocabular. Já o *Skip-Gram* apresenta melhores resultados quando há menos dados de treinamento e consegue representar melhor palavras mais raras do vocabulário (NASEEM et al., 2020). Com isso, a escolha de uma técnica em detrimento da outra é puramente relacionada às características do corpus utilizado.

2.4.3.2 Wang2Vec

Essa arquitetura, proposta por (LING et al., 2015), consiste em uma pequena modificação da *Word2Vec*, com o intuito de considerar o impacto da ordem das palavras no texto e conseguir melhorar a captura de aspectos sintáticos das palavras. Essa característica faz com que o *Wang2Vec* seja bastante útil para problemas de marcação de classe gramatical (*Part of Speech Tagging* (POS)) e análise de dependência entre os elementos textuais. Assim como o Word2Vec e as demais técnicas de *embeddings* que serão descritas, esse método também pode ser realizado utilizando o CBOW ou *Skip-Gram* como forma de mapeamento das palavras em vetor.

2.4.3.3 Glove

Abreviação de *Global Vectors for Word Representation*, o *Glove* é método apresentado em 2014 por (PENNINGTON; SOCHER; MANNING, 2014), com o objetivo de ser uma versão

expandida do *Word2Vec*, no sentido de obter vetores de representação que consigam capturar o significado da palavra de forma mais aprofundada. No caso do *Word2Vec* (e *Wang2Vec*), existe uma limitação do contexto da palavra de análise, dado pelo tamanho da janela de palavras em seu entorno. Já o *Glove* busca representar uma palavra capturando o contexto geral de todo o vocabulário onde ela está inserida. O próprio nome da técnica explica isso: a ideia de "vetores globais", de acordo com (PENNINGTON; SOCHER; MANNING, 2014), indica que esse modelo é capaz de capturar tanto a estatística local quando a estatística global do corpus de treinamento utilizado.

Essa estatística global consegue ser obtida uma vez que o método *Glove* é realizado a partir da razão da probabilidade de co-ocorrência entre duas palavras como uma diferença entre vetores. Para exemplificar a construção da matriz de co-ocorrência, pode-se considerar um corpus formado apenas pelas seguintes frases :

O **gato** está **dormindo** na cozinha. (I)
O **cachorro** está **comendo** na cozinha. (II)

Sua matriz de co-ocorrência é obtida a partir de todas as palavras do vocabulário, removendo as duplicações. Considerando uma janela de tamanho n = 1, cada casela da matriz corresponde à quantidade de ocorrências em que uma palavra antecede ou sucede a outra. Por exemplo, as palavras **gato** e **cachorro** são antecedidas por "O"e precedidas pela palavra "está", de modo que a correspondência entre essas palavras é igual a 1. Como as demais palavras do vocabulário estão a uma distância maior que 1, as demais correspondências entre essas palavras são iguais a 0, como mostrado em destaque na Figura 6.

Figura 6 – Exemplo de uma matriz de co-ocorrência.

	o	está	na	cozinha	cachorro	comendo	gato	dormindo
o	0	0	0	0	1	0	1	0
esta	0	0	0	0	1	1	1	1
na	0	0	0	2	0	1	0	1
cozinha	0	0	2	0	0	0	0	0
cachorro	1	1	0	0	0	0	0	0
comendo	0	1	1	0	0	0	0	0
gato	1	1	0	0	0	0	0	0
dormindo	0	1	1	0	0	0	0	0

Fonte: a autora (2022).

Genericamente, considerando um corpus com V palavras, a matriz de co-ocorrência desse corpus será de tamanho VxV, onde cada palavra *i* é única no corpus e cada palavra *j* denota o número de vezes que ocorreram, no tamanho da janela, a palavra i (GHARIBI, 2019). Desse modo, a função de pesos utilizada no *Glove* é obtida a partir da probabilidade da palavra *j* aparecer no contexto da palavra *i*, que é calculada através da equação:

$$J(\theta) \sum_{i,j=1}^{W} f(P_{i,})(u_i^T v_j - \log P_{i,j})^2 \tag{2.7}$$

Que busca minimizar a diferença entre o produto da representação vetorial de (i,j) e o *log* da contagem de co-ocorrência de (i,j). O termo $f(P_{i,j})$ realiza uma soma ponderada que atribui menores pesos para palavras com elevada co-ocorrência, conseguindo então capturar a importância de cada par de palavras (JHA, 2021).

Devido à sua abordagem de estatística global, o *Glove* consegue capturar características semânticas de palavras raras e entregar boa performance mesmo para corpus considerados pequenos. Em termos de aplicabilidade, ele é amplamente utilizado em modelos voltados pra detecção de similaridade e problemas de Reconhecimento de Entidades (JHA, 2021).

2.4.3.4 FastText

A principal diferença e destaque do método *FastText* em relação aos outros modelos de *embedding* descritos anteriormente, é sua capacidade de aprender informações sobre palavras que estão fora do vocabulário do corpus utilizado no treinamento (NASEEM et al., 2020).Ele consegue fazer isso pois, em vez de utilizar palavras para a construção dos *embeddings*, ele se aprofunda no nível de caracteres das palavras, construindo blocos de possíveis combinações de caracteres.

O modelo, proposto pelo grupo de pesquisa em Inteligência Artificial do *Facebook*, tem como princípio a quebra de palavras em n-*grams*, com n podendo assumir diversos valores, conforme mostrado na Figura 7 para a palavra *reading* ("lendo", em Inglês).

Uma vez que a quantidade de n-*grams* de um corpus é muito superior à de vocábulos, essa estratégia tende a melhorar expressivamente a forma de capturar o relacionamento entre as palavras, juntamente com sua semântica (BOJANOWSKI et al., 2017).

Figura 7 – Combinações de caracteres para diferentes valores de n-gram de uma palavra

Word	n-gram	combinations
reading	3	<re, rea, ead, adi, din, ing, ng>
reading	4	<rea, read, eadi, adin, ding, ing>
reading	5	<read, readi, eadin, ading, ding >
reading	6	<readi, readin, eading, ading>

Fonte: (JHA, 2021).

2.4.4 Meta Embeddings

Diante dos diversos métodos de obtenção de *word embeddings* citados nas seções anteriores, surge um questionamento comum na etapa de extração de características da maioria dos problemas de PLN: considerando o uso de *embeddings* pré-treinados (vetores construídos a partir da aplicação de determinada técnica de *embedding* em grandes fontes de dados textuais obtidas de contextos diversos - livros, notícias, Wikipédia, etc), qual o tipo que melhor se encaixa ao contexto do problema a ser resolvido? Uma vez que diferentes técnicas de *embedding* utilizam diferentes formas de capturar o significado de sentenças e documentos, e diferentes vetores de *embeddings* pré-treinados normalmente são oriundos de diferentes tipos de corpus, uma opção viável consiste na ideia de combinar diferentes tipos de *embeddings* com o objetivo de aproveitar essas diversidades para aprender incorporações de palavras com melhor desempenho (YIN; SCHÜTZE, 2015).

De acordo com (YIN; SCHÜTZE, 2015), a ideia de combinação de *embeddings*, também chamada de meta-*embedding*, possui dois benefícios significativos. O primeiro corresponde ao aprimoramento na representação de palavras: meta-*embeddings* normalmente possuem um desempenho melhor do que os conjuntos de *embeddings* individuais. Em segundo, a cobertura: as combinações possuem maior representação vocabular que cada *embedding* pré-treinado individualmente (YIN; SCHÜTZE, 2015). Além disso, um dos maiores problemas nos sistemas de PLN é lidar com palavras fora do vocabulário utilizado para o treinamento dos *embeddings* (KIELA; WANG; CHO, 2018), o que pode ser contornado a partir de meta-*embeddings*. Nos últimos 7 anos, diversas formas e modos de aplicação dos meta-*embeddings* foram propostas na literatura. A forma mais simples de combinar *embeddings* é apresentada no trabalho de (YIN;

SCHÜTZE, 2015) e consiste na concatenação dos vetores. Já em 2018, (KIELA; WANG; CHO, 2018) apresentaram uma forma mais elaborada de combinação, chamada de Meta-*Embedding* Dinâmico (*Dynamic Meta Embedding*, em Inglês) e sua extensão, baseada em contexto: Meta-*Embedding* Dinâmico Contextualizado (*Contextualized Dynamic Meta Embedding*). Essas três formas de combinação de *embeddings* foram utilizadas no presente trabalho e seus métodos serão explicados a seguir.

2.4.4.1 Concatenação

A concatenação de vetores de *embeddings* (CONCAT), consiste na simples junção das representações dos diferentes *embeddings* para uma mesma palavra. Assim, considerando um termo t representado por n vetores de pesos de distintos, então a representação do termo t a partir da combinação de n técnicas de *embedding*, será dada por:

$$w_t^{CONCAT} = [w_{1,t}, w_{2,t}...w_{n,t}] \tag{2.8}$$

Onde n é a quantidade de *embbedings* utilizados na concatenação e $d_i(i = 1, 2, 3..n)$ representa a dimensão de cada *embedding*. Logo, a dimensão final do *embedding* obtido através da concatenação será $D_{w(concat)} = \sum_{i=1}^{n} d_i$. É fácil notar que esta forma de combinação pode comprometer o tempo de treinamento de um algoritmo de AM, devido ao aumento da dimensionalidade dos vetores de representação (KIELA; WANG; CHO, 2018). Entretanto, é um método de fácil aplicação e que costuma trazer melhoras de performance em vários problemas de PLN, uma vez que combina as formas de representação de diferentes métodos de *embedding* para um mesmo termo (WANG et al., 2021).

2.4.4.2 Meta Embeddings Dinâmicos

A principal diferença da forma dinâmica de combinar diferentes vetores de *embbeding* pré-treinados para outras formas de combinação mais simples (como é o caso da concatenação) é que ela aplica mecanismos de auto-atenção de modo a aprender como os diferentes pesos de cada *embedding* individual para determinado termo se combinam (KIELA; WANG; CHO, 2018). Isso é realizado através de uma rede neural que calcula a importância dos pesos de cada *embedding* na representação de determinada palavra, fornecendo a representação como uma

combinação de pesos que melhor represente o termo de análise (KIELA; WANG; CHO, 2018).

Dessa forma, meta-*embeddings* dinâmicos, são obtidos a partir da projeção de cada *embedding* em um espaço dimensional comum $d^{'}$ que transforma cada vetor de *embedding* w_i em $w_i^{'}$, obtido aplicando-se uma função linear sobre os pesos de w_i. Esses novos vetores são então combinados a partir de uma soma ponderada, de modo que o vetor final representativo do termo t será obtido através da equação 2.9:

$$w_t^{DME} = \sum_{i=1}^{n} \alpha_{i,t} w^{'} i, t \tag{2.9}$$

Onde $\alpha_{i,t} w^{'} i, t$ são os pesos obtidos através do mecanismo de auto-atenção realizado por uma rede neural recorrente (KIELA; WANG; CHO, 2018).

A combinação dinâmica de *embeddings* também pode ser realizada considerando mecanismos de auto-atenção que dependem do contexto em que a palavra está inserida, gerando assim a combinação Meta *Embedding* Dinâmico Contextualizado (CDME) (KIELA; WANG; CHO, 2018). A principal diferença entre o cálculo do vetor de *embedding* final entre as técnicas DME e CDME consiste apenas no tipo de rede neural utilizada no mecanismo de auto-atenção, que no caso da CDME corresponde a uma rede neural recorrente bidirecional, uma vez que esse tipo de rede consegue realizar o aprendizado dos pesos do começo ao fim e do fim ao começo dos vetores de *embedding*. A rede neural utilizada em ambas as técnicas é uma LSTM, cujo funcionamento será explicado na próxima seção.

2.5 MODELOS DE APRENDIZAGEM DE MÁQUINA

A AM é uma das sub-áreas de estudo da ciência da computação e consiste em um tipo de Inteligência Artificial que fornece às máquinas a capacidade de identificar padrões e aprender sobre eles de forma automatizada (MAHDAVINEJAD et al., 2018). O desenvolvimento do mecanismo para identificar automaticamente esses padrões é chamado de modelo de classificação, que pode ser desenvolvido de três formas:

- **Aprendizado Supervisionado**: esse tipo de aprendizagem necessita de um conjunto de dados previamente classificado, chamado conjunto de treinamento, a partir dos quais o modelo será capaz de aprender essas características e associá-las a cada classe (rótulo) de treinamento, de modo a predizer informações para novos conjuntos de dados não

observados pelo algoritmo de classificação (MAHDAVINEJAD et al., 2018), comumente chamados de dados de teste ou validação.

- **Aprendizado Não Supervisionado**: nesse tipo de classificação o conjunto de treinamento não está previamente classificado. Ele consiste em dados distribuídos, que devem ser classificados de acordo com a densidade do conjunto apresentado. Um exemplo de configuração comum no aprendizado não supervisionado consiste no agrupamento de dados com base em suas similaridades criando grupos, denominados *clusters* (ARAUJO, 2019).

- **Aprendizado Semi-supervisionado**: essa estratégia combina informações rotuladas e não rotuladas com o objetivo de prever a classificação de dados não rotulados. Normalmente algoritmos desse tipo utilizam poucos dados rotulados, que são incluídos no treinamento com o objetivo de melhorar a precisão de classificação do algoritmo.

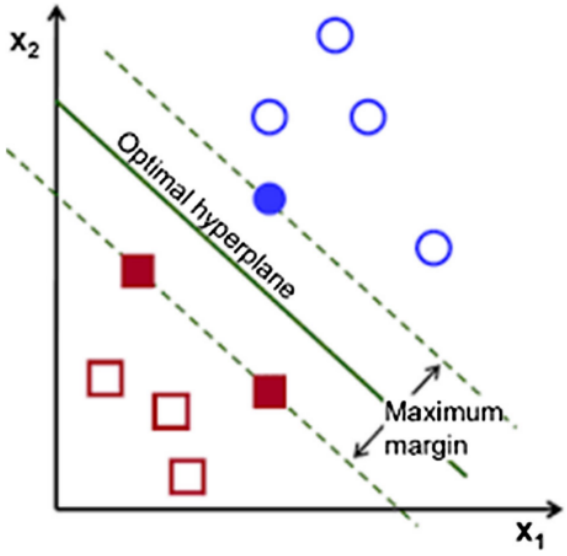
O presente trabalho utilizou apenas modelos de aprendizado supervisionado, desde os mais clássicos (SVM e Regressão Logística), os baseados em árvores (Xgboost e LGBM) e redes as neurais (MLP, CNN e LSTM).

2.5.1 SVM

O algoritmo classificador Máquina de Vetor de Suporte (*Supported Vector Machine* (SVM), em inglês), foi introduzido por (CORTES; VAPNIK, 1995), inicialmente com o objetivo de separar dados em apenas duas classes. A ideia principal deste classificador consiste na busca por um hiperplano capaz de separar bem as classes em um plano cartesiano. Este hiperplano é obtido a partir do treinamento dos dados para, posteriormente, classificar os dados de teste, de modo que a predição da classe de um novo exemplo não visto pelo algoritmo é determinada a partir de qual área do hiperplano o conteúdo daquele exemplo se encontra (CORTES; VAPNIK, 1995).

Um hiperplano é considerado ótimo quando consegue separar as classes de modo que a margem de separação entre as diferentes classes seja máxima (KADHIM, 2019). A Figura 8 ilustra geometricamente este conceito.

Figura 8 – Representação de um hiperplano ótimo para uma classificação binária



Fonte: (KADHIM, 2019)

A popularidade da utilização do SVM nos problemas mais clássicos de PLN ocorre justamente devido à linearidade deste classificador no caso de modelos de classificação binária, uma vez que cada coeficiente da função linear pode ser associado à contagem de cada palavra em uma sentença (GARG; VERMA, 2018). Dessa forma, apesar de ser uma técnica frequentemente utilizada em problemas de classificação binária, é possível aplicá-la também para dados que não são linearmente separáveis (conjuntos de dados com mais de duas classes), como é o caso dos dados utilizados nessa pesquisa. Para isso, o algoritmo utiliza funções que permitem o mapeamento de um conjunto não-linear para um espaço dimensional maior, denominado de espaço de características, tornando possível a separação linear (FERREIRA, 2018). Essas funções de núcleo estão disponíveis no classificador SVM da biblioteca *scikit-learn*.

2.5.2 Regressão Logística

Assim como o SVM, o algoritmo de Regressão Logística foi concebido inicialmente para lidar com problemas binários, de modo que ele computa a probabilidade de um exemplo pertencer à determinada classe a partir de uma função logarítmica (LAVALLEY, 2008). No caso de problemas de classificação não-binários, é utilizada uma extensão da Regressão Logística chamada de Regressão Logística Multinomial, que, na área de PLN é mais conhecida como Modelo de Máxima Entropia (MaxEnt) (CARVALHO, 2018). Na prática, o MaxEnt busca encontrar a classe (c) de modo a maximizar a função de probabilidade que relaciona cada documento (d)

dos dados de treinamento com os demais documentos da base de dados, juntamente com a probabilidade de d pertencer à classe c (GARG; VERMA, 2018).

2.5.3 Gradient Boosting Machines

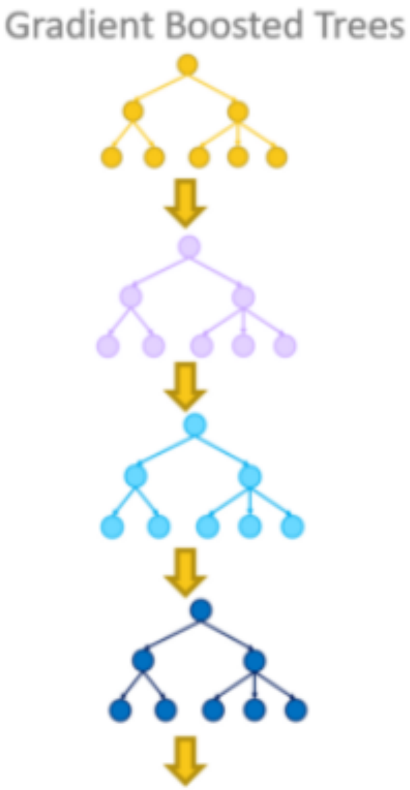
Os modelos baseados em árvores de decisão utilizados nos experimentos deste trabalho fazem parte de uma família de algoritmos similares, conhecidos como *Gradient Boosting Machines* (GBM).

Algoritmos baseados em árvore iniciam o processo de classificação atribuindo pesos iguais para todos os exemplos de treinamento. Após a avaliação da primeira árvore, os pesos das observações mais difíceis de classificar começam a aumentar enquanto os pesos das observações mais triviais diminuem, fornecendo então árvores ponderadas (SINGH, 2018). Nas técnicas de GBM, a principal ideia é combinar o resultado das diferentes árvores a cada iteração, de modo a melhorar as previsões do resultado anterior até que um ponto de convergência seja atingido (DAOUD, 2019). Nos últimos anos, três técnicas baseadas em GBM vêm ganhando destaque devido aos bons resultados que apresentam no âmbito da indústria, no meio acadêmico e em sites de competições de AM, são eles: CatBoost, XGBoost e LightGBM (WANG; HASTIE, 2014). Nessa pesquisa apenas as duas últimas técnicas foram utilizadas, por isso serão descritas a seguir.

2.5.4 XGBoost

O nome *XGBoost* corresponde à aglutinação da expressão em Inglês *eXtreme Gradient Boosting*, ou seja, trata-se de uma versão mais escalável, rápida e robusta em relação à GBMs anteriores (DAOUD, 2019). A estrutura da árvore de decisão do *XGBoost* presente na Figura 9 mostra como o algoritmo é construído: a cada iteração uma nova árvore é treinada a partir dos resíduos das folhas da árvore anterior (SILIPO, 2020). A principal diferença entre o *XGBoost* e outras formas de GBMs está relacionada à função de regularização deste algoritmo, que é calculada a partir da sua função de perda e projetada de modo a diminuir um problema comum em AM, conhecido como *Overfitting*, que representa a dificuldade de um modelo em prever novos exemplos após o treinamento dos dados (DAOUD, 2019).

Figura 9 – Estrutura de Árvore do *XGBoost*.

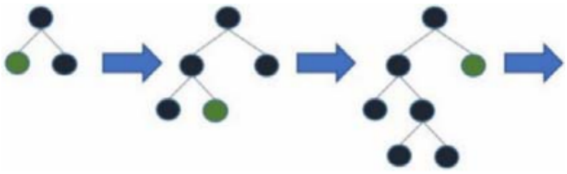


Fonte: (SILIPO, 2020)

2.5.5 LightGBM

Para reduzir o tempo de implementação de técnicas como o XGBoost, uma equipe da *Microsoft* (KE et al., 2017) desenvolveu o algoritmo *LightGBM*: cuja tradução para o Português significa literalmente uma versão mais "leve"de GBM, comumente chamado apenas de LGBM. A principal diferença em relação à outras técnicas baseadas em árvore, é que as árvores de decisão no LGBM são cultivadas em folha. Isso significa que, em vez de verificar todas as as folhas anteriores para gerar cada nova folha, todos os atributos são classificados e agrupados previamente antes da próxima iteração. Esse tipo de implementação é chamado de implementação de histograma e pode ser observado na estrutura mostrada na Figura 10 (DAOUD, 2019).

Figura 10 – Estrutura de Árvore do LGBM.



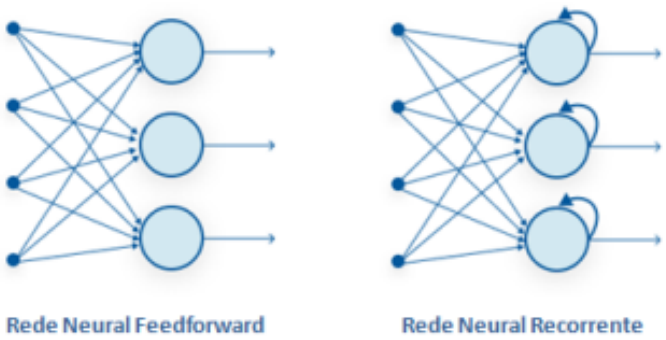
Fonte: (DAOUD, 2019)

O LGBM possui diversas vantagens, tais como: melhor precisão, velocidade de treinamento mais rápida, capacidade de manipulação em larga escala e pode ser executado em unidades de processamento comuns como é o caso das GPUs (DAOUD, 2019).

2.5.6 MultiLayer Perceptron

Este tipo de modelo classificador consiste em uma Rede Neural Artificial (RNA), que por sua vez corresponde a um conjunto de neurônios (nós) conectados, de forma semelhante ao cérebro humano (GARG; VERMA, 2018). No caso da *Multilayer Perceptron* (MLP), a rede é construída com pelo menos três camadas e de modo que as conexões entre os nós não formam um ciclo, como ocorre no caso de uma Rede Neural Recorrente (RNR). Esse tipo de rede não cíclica que compõe a MLP é conhecida como *feedfoward*. A diferença entre essas redes é mostrada esquematicamente na Figura 11.

Figura 11 – Comparação entre uma rede neural *feedfoward* e uma rede cíclica.



Fonte: (NASCIMENTO, 2019)

Assim como o modelo SVM, a MLP também implementa uma função discriminante linear para resolver problemas categóricos. Além disso, ela possui significativa capacidade de mapear bem grandes espaços de características, reduzindo regiões de ambiguidade entre diferentes

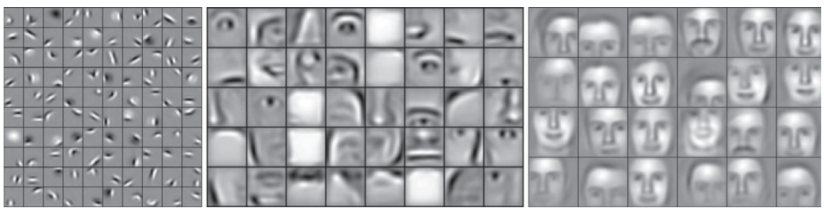
classes (DUDA; HART; STORK, 2012). Essa característica torna esse tipo de modelo bastante atrativo para problemas de PLN, e justifica também sua escolha como um dos classificadores utilizados nesse trabalho.

2.5.7 Modelos de Aprendizado Profundo

Os modelos de aprendizado profundo (*Deep Learning*, em Inglês), vêm se mostrando como uma forma de AM extremamente poderosa nos últimos anos, uma vez que esse tipo de aprendizagem consegue formar padrões a partir de múltiplas camadas de representações dos dados e produzir resultados de previsão considerados de "última geração"(ZHANG; WANG; LIU, 2018). A arquitetura de aprendizado profundo atingiu grande sucesso na área de AM, especialmente nos campos de visão computacional e PLN, uma vez que os principais problemas relacionados a essas áreas de pesquisa envolvem elevadas quantidades de dados (JAIN; SARAVANAN; PAMULA, 2021).

De modo geral, o aprendizado profundo usa uma cascata de várias camadas de unidades de processamento não-lineares para extração e transformação de recursos. As camadas inferiores, próximas à entrada de dados, aprendem recursos simples, enquanto as camadas superiores aprendem recursos mais complexos derivados de recursos da camada inferior. A arquitetura forma uma representação hierárquica e poderosa de recursos. Esse mecanismo pode ser mostrado a partir do exemplo da Figura 12, que mostra o processo hierárquico de obtenção de recursos da esquerda (uma camada inferior) para a direita (uma camada superior) realizado pelo aprendizado profundo na classificação de imagens faciais (LEE et al., 2009). É fácil notar que os recursos de imagem aprendidos crescem em complexidade, começando com bolhas/-bordas, passando para narizes/olhos/bochechas até evoluir para rostos (ZHANG; WANG; LIU, 2018).

Figura 12 – Forma hierárquica de extração de características realizada no Aprendizado Profundo.



Fonte: (LEE et al., 2009).

Neste trabalho, duas configurações de redes de aprendizado profundo bastante comuns

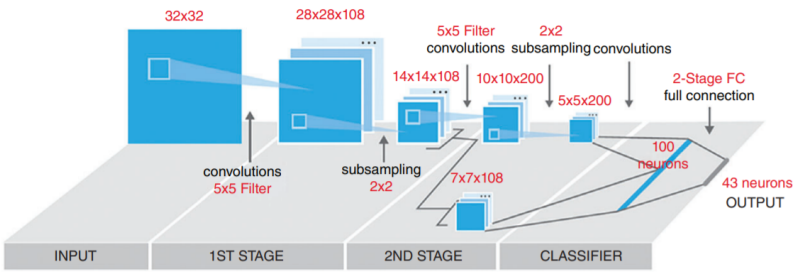
foram utilizadas: as redes convolucionais (*Convolucional Neural Network* (CNN)) e de aprendizado recorrente (*Long Short-Term Memory* (LSTM)). Elas foram escolhidas principalmente devido à sua popularidade e boa performance diante de problemas relacionados à PLN.

2.5.7.1 Redes Neurais Convolucionais

Essa redes são RNAs do tipo *feedfoward*, originalmente aplicadas na área de visão computacional, mas que expandiram para outras áreas devido a sua significativa capacidade de extrair informações localmente (AGARWALL et al., 2020). Seu design foi inspirado no córtex visual humano, que possui muitas células responsáveis pela detecção da luz, chamadas de campos receptivos. Essas células atuam como filtros locais sobre o espaço de entrada. Seguindo esta lógica, a CNN consiste em múltiplas camadas convolucionais, onde cada uma desempenha a função que é processada pelas células no córtex visual (ZHANG; WANG; LIU, 2018).

A Figura 13 mostra a arquitetura da CNN. A camada convolucional é composta por um conjunto de filtros que visam extrair características da informação de entrada (*input*). Cada filtro é na verdade uma matriz de números (chamados pesos ou parâmetros) e, à medida que o filtro está deslizando (ou convoluindo), ele está multiplicando seus valores de peso com o valores originais dos dados de entrada, de modo que esses são segmentados em vetores denominados *feature maps*. No exemplo da Figura 13, são utilizados 108 filtros no primeiro estágio da rede, de modo que após essa etapa são obtidas 108 *features* "empilhadas", que formam a primeira camada convolucional da rede. Em seguida, uma camada chamada *pooling* é aplicada para redução dimensional dos vetores produzidos na camada anterior, visando reduzir a complexidade computacional da rede. Após a segunda convolução dos dados, a CNN utiliza uma função de ativação para fornecer as classes de saída do modelo.

Figura 13 – Arquitetura de uma CNN.



Fonte: (ZHANG; WANG; LIU, 2018).

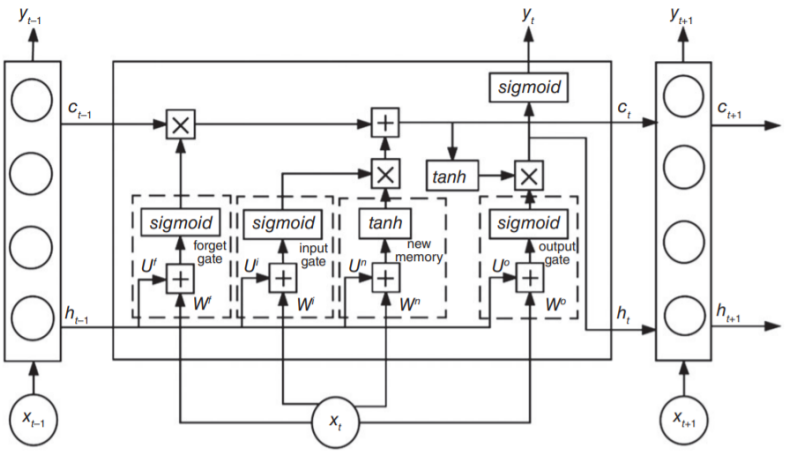
As camadas convolucionais da CNN conseguem estabelecer bons padrões de conectividade

local entre neurônios de camadas adjacentes. Tal característica se mostra bastante útil nos problemas de classificação em PLN, uma vez que é necessário encontrar fortes pistas locais sobre a pertinência de uma dada sentença a uma classe, baseando-se em pistas que podem aparecer em diferentes trechos de uma frase (ZHANG; WANG; LIU, 2018).

2.5.7.2 Long short-term Memory

A rede LSTM corresponde a um tipo de RNR que tem por principal característica o envio de sinais de *feedback* entre diversos neurônios, permitindo que eles apresentem um comportamento temporal dinâmico (GROSSBERG, 2013). Ao contrário das redes *feedforwards*, as RNRs podem usar seu estado interno (memória) para processar as sequências de entradas. Todas as redes recorrentes possuem a forma de uma cadeia de módulos repetitivos, que no caso da LSTM são estruturados de forma mais complicada: ao invés de ter uma única camada de rede neural, há quatro camadas interagindo de uma maneira especial na LSTM (ZHANG; WANG; LIU, 2018).

Figura 14 – Arquitetura de uma rede LSTM.



Fonte: (ZHANG; WANG; LIU, 2018).

A arquitetura de uma LSTM é mostrada na Figura 14. O armazenamento da informação de entrada em um determinado instante t (representado na Figura 14 por x_t), é realizado a partir de uma função sigmoide que atua na primeira camada da rede, denominada *forget gate* (f_t). Se o valor de f_t for igual a 1, então a informação é armazenada na rede e descartada em caso contrário. A próxima etapa do aprendizado ocorre na camada *input gate*, que decide quais informações da rede deverão ser atualizadas. Em seguida, uma função tangente hiperbólica

(*tanh*) cria um novo vetor de valores candidatos, que finalmente serão combinados na última camada, onde ocorre a atualização do estado t para $t+1$. O resultado final após a seleção dos valores de saída desejados é enviado para a camada de saída (*output gate*) (ZHANG; WANG; LIU, 2018).

Um dos motivos para os bons resultados de classificação obtidos com esta rede para problemas de PLN está diretamente associado à característica de processamento dos dados de forma mais lenta e cuidadosa, bem como sua capacidade de decidir o que é necessário lembrar, atualizar e prestar atenção (AGARWALL et al., 2020). Esse recurso é muito útil quando o objetivo é extrair o contexto ou polaridade de sentenças mais extensas e complexas, o que viabiliza sua utilização nas técnicas de meta-*embeddings* dinâmicos utilizadas nessa pesquisa.

2.6 VALIDAÇÃO E MÉTRICAS DE AVALIAÇÃO

Essa seção discorre sobre a forma de validação considerada durante o treinamento dos modelos realizados, bem como as principais métricas utilizadas para avaliar a performance dos modelos desenvolvidos em problemas de classificação.

2.6.1 Validação Cruzada

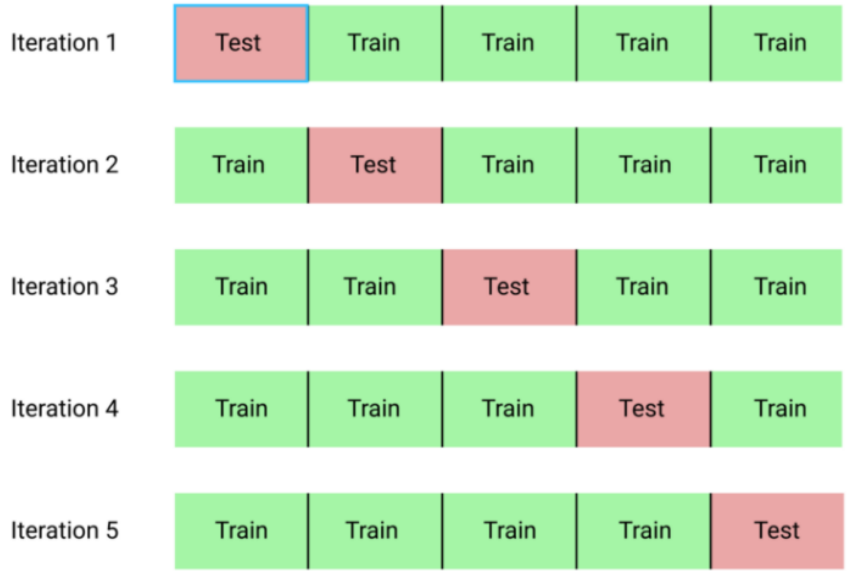
O principal intuito da validação cruzada durante o processo de treinamento de um modelo de aprendizagem de máquina é reduzir os possíveis erros de predição do modelo obtido quando o mesmo tiver que lidar com dados desconhecidos. Normalmente, durante o processo de modelagem, os dados são divididos em apenas duas ou três partes: treino e teste ou treino, teste e validação. Os dados de teste e validação são aqueles nunca vistos pelo modelo e servem para medir sua capacidade de predição após a etapa de treinamento (RABELLO, 2019).

Apesar de ser uma boa abordagem, dividir os dados aleatoriamente em apenas duas ou três partes pode não ser suficiente para garantir a capacidade de generalização do modelo. Quando o processo de validação cruzada é aplicado, o modelo realiza a validação dos seus resultados ainda durante o treinamento, o que torna esta técnica mais atrativa.

A Figura 15 ilustra a forma como o processo de validação cruzada ocorre. Inicialmente, é definido um valor k, que representa a quantidade de divisões aleatórias em que a base de treinamento será dividida, de modo que cada divisão possua aproximadamente a mesma quantidade de exemplos. O parâmetro k é popularmente chamado de *fold*, que significa pasta,

em Inglês. Isso porque a cada iteração um conjunto formado por k-1 subconjuntos dos dados são utilizados para treinamento e o subconjunto que sobra será utilizado para avaliação do modelo, gerando um resultado de métrica para avaliação (por exemplo: precisão ou acurácia). Esse processo garante que cada subconjunto de dados seja utilizado para avaliar o modelo em algum momento durante seu treinamento.

Figura 15 – Processo de Validação Cruzada com k = 5



Fonte: (RABELLO, 2019).

Algumas das possíveis métricas de avaliação que podem ser utilizadas durante um processo de validação cruzada são explanadas a seguir.

2.6.2 Métricas de avaliação

A abordagem proposta neste trabalho está relacionada a métodos que buscam inferir a polaridade dos sentimentos e opiniões presentes em textos, ou seja, problemas de classificação. Diversas são as métricas que podem ser utilizadas para mensurar a qualidade de um modelo, porém na área de PLN alguns métodos de avaliação se destacam. São eles: precisão, *recall*, F_1score e acurácia (KADHIM, 2019). Cujas forma de cálculo são dadas por:

$$precisao = \frac{VP}{VP + FP} \tag{2.10}$$

$$recall = \frac{VP}{VP + FN} \tag{2.11}$$

$$F_1 = 2 * \frac{precisao * \textit{recall}}{precisao + \textit{recall}} \tag{2.12}$$

$$acurácia = 2 * \frac{VP + VN}{VP + FP + VN + FN} \tag{2.13}$$

Onde VP (Verdadeiro Positivo) representa a quantidade de exemplos positivos classificados como positivos; VN (Verdadeiro Negativo) significa o total de elementos positivos classificados como negativos; FP (Falso Positivo) é a quantidade de exemplos falsos que foram classificados como positivos e FN (Falso Negativo) corresponde a quantidade de exemplos falsos classificados como falsos.

Das métricas acima, a acurácia é a mais comum para a maioria dos modelos de classificação, desde que as classes sejam balanceadas, fornecendo uma medida direta da qualidade de um modelo (BOOK, 2021). As demais métricas requerem que o rótulo de classificação dos exemplos seja necessariamente binário (apenas duas possibilidades de valores: comentários positivos e negativos, por exemplo). Para problemas com mais de duas classes, como é o caso dos dados utilizados neste trabalho, pode-se calcular essas métricas para cada classe individualmente: ou seja, para a métrica *precisão*, por exemplo, seriam calculadas a precisão da classe positiva, da classe neutra e da classe negativa. No caso de problemas com múltiplas classes, existem também outras formas de computar as métricas de precisão, *recall* e F_1 gerais do modelo: isso pode ser realizado, por exemplo, a partir da média dos resultados de cada classe ou considerando apenas as classes positivas (nesse caso a métrica é chamada Macro F_1, Macro *recall* ou Macro precisão).

2.7 TESTES ESTATÍSTICOS

Apesar das métricas de avaliação de classificadores serem essenciais para um estudo comparativo de diferentes técnicas e informarem o nível de assertividade de um modelo na solução de um problema, elas não são o suficiente para comparar o comportamento dos classificadores em diferentes cenários experimentais. Diante disso, a avaliação estatística dos resultados é necessária para comprovar (ou descartar) a hipótese do ganho de performance de um modelo em relação ao outro em diferentes conjuntos de dados (DEMŠAR, 2006).

Dessa forma, neste trabalho são considerados os testes estatísticos apresentados em (DEMŠAR, 2006) como formas de validação das performances obtidas pelos diferentes algoritmos de AM

utilizados na fase experimental dessa pesquisa. Considerando o pequeno volume de amostras que serão comparadas (uma vez que validação cruzada foi realizada com k $= 10$ *folders*), não será realizado teste para verificação de normalidade das distribuições, apenas os testes não-paramétricos. Dois tipos de comparações serão realizadas: as comparações entre pares de classificadores, para as quais será utilizado o teste de Wilcoxon, e as que envolvem a comparação entre mais de duas amostras, onde o teste de Friedman será aplicado. A abordagem de ambos os testes serão discutidas nas subseções a seguir.

2.7.1 Teste de Wilcoxon

O teste de Wilcoxon corresponde a uma alternativa não-paramétrica ao teste *t-Student* e avalia as diferenças de desempenho entre dois classificadores para cada conjunto de dados a partir da diferença entre a mediana das amostras (DEMŠAR, 2006). Ele é realizado da seguinte forma:

1. Elaboração das hipóteses:

   $H_0 : \Delta = 0$: *a mediana da diferença entre as amostras é nula*

   $H_1 : \Delta \neq 0$: *a mediana da diferença entre as amostras não é nula*

2. Definir nível de significância (α). Para os testes realizados nessa pesquisa, será considerado $\alpha = 0.05$, dessa forma tem-se um nível de 95% de confiança no resultado do teste;

3. Cálculo da estatística w e tomada de decisão:

   Rejeitar a H_0 caso o $w_{calculado} < w_{\alpha_1}$ ou $w_{calculado} > w_{\alpha_2}$ onde w_1 e w_2 são os valores críticos da distribuição Wilcoxon. Na prática, a hipótese H_0 só poderá ser rejeitada se o p-valor da estatística for tal que: p-valor $<= 0.05$.

2.7.2 Teste de Friedman

O Teste de Friedman corresponde a uma variação não-paramétrica do teste de Análise de Variância (ANOVA). Nos problemas de comparação entre classificadores de AM, ele se apresenta como uma alternativa mais completa que a ANOVA, uma vez que esta estatística é baseada na distribuição normal, o que não pode ser garantido na análise do desempenho

de muitos algoritmos de aprendizado de máquina (DEMŠAR, 2006). O teste de Friedman se baseia na estratégia de ranqueamento: ele classifica os algoritmos para cada conjunto de dados separadamente, o algoritmo de melhor desempenho obtendo a classificação de 1, a segunda melhor classificação 2 e assim sucessivamente. Em caso de empates, a posição do *ranking* é definida a partir da média das demais posições.

Considerando r_i^j o valor da performance do j-ésimo algoritmo no i-ésimo de N bases de dados. O teste de Friedman compara as médias das performances dos algoritmos e tem como hipótese nula a afirmação de que todos os algoritmos são equivalentes se suas médias de performance $R_j = \frac{1}{N}\sum_i r_i^j$ forem ser iguais. Com isso, a distribuição dessa estatística é dada pela equação 2.14:

$$\chi^2 = \frac{12N}{k(k+1)}\left[\sum_j R_j^2 - \frac{k(k+1)^2}{4}\right] \tag{2.14}$$

Que apresenta k - 1 graus de liberdade, quando N e k são grandes o suficiente (N $>$ 10 e k $>$ 5). As hipóteses para este teste são:

- H_0 : $\Delta = 0$: *As distribuições de amostras pareadas são iguais*
- H_1 : $\Delta \neq 0$: *As distribuições de amostras pareadas não são iguais*

A sim como no teste de Wilcoxon, considerando um nível de significância $\alpha = 0.05$, a hipótese H_0 só poderá ser rejeitada se o p-valor da estatística for tal que: p-valor $<= \alpha$.

2.8 CONSIDERAÇÕES DO CAPÍTULO

O presente capítulo discorreu sobre os principais conceitos relacionados à área de análise de sentimentos, desde uma visão macro sobre PLN e seus principais problemas de pesquisa, passando por todas etapas existentes na maioria dos problemas de classificação de sentimentos.

A descrição desses conceitos faz-se necessária para uma compreensão mais aprofundada do projeto de pesquisa, além de trazer justificativas para as decisões tomadas durante o procedimento experimental adotado. Além disso, serve como referencial teórico a ser utilizado para pesquisas futuras relacionadas ao mesmo tema. Assim, conceitos gerais sobre diferentes modelos de aprendizagem de máquina, formas de pré-processamento de texto e testes estatísticos podem ser utilizados para outros estudos comparativos como o realizado nessa pesquisa.

Diante disso, uma das contribuições que pode ser dada do ponto de vista teórico está relacionada ao levantamento das principais arquiteturas de *embeddings* comumente usadas em problemas de PLN, em especial na comparação entre as principais características de cada uma dessas técnicas. Este levantamento está sumarizado na Tabela 4 e pode ser utilizado como critério de escolha de determinada técnica diante dos tipos de dados a serem avaliados.

Tabela 4 – Principais características de diferentes técnicas de *Embedding*

Embedding	Características				
	considera sintaxe	considera semântica	considera ordem	limitação de contexto	granularidade
Word2Vec	sim	sim	não	limitado	palavras
Wang2Vec	sim	sim	sim	limitado	palavras
Glove	sim	sim	sim	global	palavras
FastText	sim	sim	sim	limitado	caracteres (n-*grams*)

Fonte: a autora (2022)

No Capítulo a seguir serão descritas as principais contribuições atuais da literatura na área de AS, com ênfase nos trabalhos que, assim como esta pesquisa, buscaram aumentar o ganho de performance nas classificações dos sentimentos a partir de intervenções e melhorias na etapa de extração de características dos textos.

3 TRABALHOS RELACIONADOS

Inúmeros têm sido os trabalhos publicados na área de PLN nos últimos anos, em especial aqueles relacionados à problemas de Análise de Sentimentos. Nessa pesquisa, o problema de AS em questão consiste no estudo comparativo entre diferentes técnicas de *embedding* aplicadas à avaliações de produtos escritas em Português brasileiro e no impacto da combinação de vetores de *embedding* pré-treinados.

Na literatura, é fácil encontrar muitos trabalhos de AS para bases de dados formadas por textos em Inglês, devido à popularidade e hegemonia deste idioma em todo mundo. Um compilado de referenciais teóricos é realizado por (HUSSEIN, 2018), e apresenta um levantamento das publicações de maior relevância relacionadas à diversos problemas de AS na última década. Nele são citados desafios como detecção de *fake news* e *spam*. Já em (KADHIM, 2019), são levantadas todas as etapas envolvidas e principais técnicas de aprendizagem de máquina para diversos tipos de classificações de texto escritos em Inglês.

No que diz respeito à análise de sentimento voltada para a detecção de polaridade e recomendação a partir de avaliações de usuários em lojas *on-line*, (LIN, 2020) realiza um estudo comparativo entre diversas técnicas de AM aplicadas a uma base composta por avaliações de uma loja de roupa escritas em Inglês. Considerando algoritmos populares como SVM, *XGBoost* e LGBM, eles conseguiram obter resultados de F1 *score* médio igual a 97% para uma classificação binária utilizando o TF-IDF como técnica de vetorização.

Outras formas de *embeddings* além das tradicionais (*Word2vec*, *Glove* e *FastText*), têm sido cada vez mais exploradas no que diz respeito à vetorização de texto. Em (SILVA; CASELI, 2020), é apresentado um estudo de *embeddings* que consideram o sentido da palavra em vez da sua ocorrência no texto, tendo como objetivo resolver o problema da ambiguidade de determinadas palavras presentes em uma sentença. A técnica, denominada *Sense Embedding*, apresentou maior acurácia quando comparada aos outros métodos de *embbeding* e foi realizada em um problema de AS para uma base com dados em Português-BR e Europeu. Já em (DOVAL; VILARES; GÓMEZ-RODRÍGUEZ, 2020) é realizada uma modificação na arquitetura do *SkipGram* com o intuito de melhorar a performance de representação das palavras presentes em textos que apresentam muitos ruídos, tais como: gírias, abreviações de palavras, regionalismos, etc. Foram obtidos resultados promissores quando comparados com as técnicas mais comuns na literatura.

Em relação à comparação entre diferentes arquiteturas de *embeddings*, assim como proposto nessa pesquisa, alguns trabalhos publicados nos últimos anos também visaram avaliar o impacto dessa abordagem em diversas tarefas de PLN aplicadas à língua portuguesa, incluindo AS. O trabalho de (HARTMANN et al., 2017), por exemplo, realizou uma comparação entre as técnicas de *embedding* *Word2vec*, *Wang2vec*, *Glove* e *FastText*, para bases de dados em Português brasileiro e de Portugal aplicados à problemas de POS e similaridade. Para esse contexto o *FastText* foi a técnica que apresentou os melhores resultados para a menor dimensionalidade na tarefa de POS, enquanto o *Word2Vec* se destacou no problema de similaridade semântica.

No campo de AS voltada para o Português brasileiro, trabalhos como o de (MARTINS; OLIVEIRA; MOREIRA, 2017) e (BRITTO; PACíFICO, 2020) contribuíram para a criação de corpus compostos por avaliações de produtos e serviços e comparam algumas técnicas de classificação utilizadas no presente trabalho, como SVM e Regressão Logística. No que diz respeito ao tipo de classificação utilizada, o trabalho de (MARTINS; OLIVEIRA; MOREIRA, 2017), assim como este, realiza uma classificação não binária, ou seja: também inclui as avaliações neutras nos dados de análise. Em (SILVA; MALHEIROS, 2019) o problema de AS em questão é relacionado às polaridades de comentários da rede social *Twitter*, onde também são consideradas três classes (positiva, negativa e neutra). Seu trabalho realiza a comparação entre algoritmos de AM e a abordagem léxica utilizada pela API *SenticNet* [1]. Utilizando uma Regressão Logística como técnica de classificação e o TF-IDF na etapa de vetorização, eles conseguem obter como melhor resultado uma acurácia de 45%, superior à obtida pelo *SenticNet* para o mesmo problema.

Considerando a abordagem de busca pela melhoria na performance de vetorização das palavras através de *embeddings* pré-treinados, (REZAEINIA et al., 2019) realiza um estudo envolvendo a combinação entre léxicos, vetores de POS e os *embeddings* *Word2Vec* e *Glove* pré-treinados, a partir da construção de uma arquitetura baseada em concatenação. O método, chamado de Vetores de Palavras Melhorados (*Improved Word Vectors*, em Inglês), apresentou bons resultados quando aplicado em modelos de aprendizado profundo. A concatenação de *embeddings* também trouxe resultados interessantes para idiomas mais complexos, como é o caso do Árabe, que possui um vetor de *embedding* pré-treinado construído especificamente para este idioma, denominado *AraVec*. Em (KAIBI; NFAOUI; SATORI, 2020), é realizado o impacto da combinação deste vetor de *embedding* com o *FastText* para a Análise de Sentimentos de *Tweets*. O estudo também compara o impacto da concatenação para diversos algoritmos

[1] <https://github.com/yurimalheiros/senticnetapi>

de AM mais clássicos, como o SVM e a Regressão Logística. Os resultados obtidos mostraram um ganho de cerca de 1.5% de acurácia quando a concatenação foi utilizada para um modelo de classificação semelhante ao SVM.

Ainda considerando os estudos voltados para a combinação entre diferentes tipos de *embeddings*, em (PETROLITO; DELL'ORLETTA, 2018), diferentes métodos de combinação são explorados para representação de documentos em Inglês. Além da concatenação, a soma e a média entre os pesos dos vetores também são analisadas. Os resultados obtidos mostraram que a soma e a concatenação de *embeddings* trouxeram os melhores ganhos para o problema de análise. Já trabalhos como o de (YIN; SCHÜTZE, 2015), (KIELA; WANG; CHO, 2018) e (R; DUBEY, 2020) apresentam formas mais elaboradas de combinação além da concatenação de vetores, visando melhorar a performance e a cobertura de vocabulários em problemas de PLN como uma alternativa ao desenvolvimento de novas técnicas que sejam mais complexas e de difícil implementação.

Os resultados de trabalhos mais antigos como o de (YIN; SCHÜTZE, 2015), indicaram o impacto que diferentes formas de combinação de *embeddings* poderiam fornecer para melhoria de performance em problemas de POS. Já o trabalho de (R; DUBEY, 2020) aplica o conceito de *meta-embedding* no estudo da similaridade entre as palavras, também obtendo bons resultados em comparação aos trabalhos que utilizaram apenas uma técnica para resolver o mesmo tipo de problema. Finalmente, os resultados de (KIELA; WANG; CHO, 2018) mostram que a utilização de Redes Neurais Recorrentes para combinar diferentes tipos de *embedding* pode ser um bom caminho a ser seguido, pelo menos no que diz respeito à tarefa estabelecer o relacionamento entre sentenças - um técnica conhecida como RTE (*Recognizing Textual Entailment*, em Inglês).

Em (BOLLEGALA; HAYASHI; KAWARABAYASHI, 2017) é proposta uma forma de *meta-embedding* que considera o contexto local das palavras em um texto, em vez que realiza-lo de forma global. O método proposto é realizado de forma não supervisionada e linear e tem como base a técnica de vizinhos mais próximos, que é comumente utilizada em problemas de clusterização. A avaliação da acurácia do método foi realizada em problemas de PLN, incluindo a classificação de sentimentos em textos curtos. Seus resultados mostraram melhor performance em diversas métricas estatísticas quando comparados a outros métodos de combinação de *embedding*, como a concatenação.

Em relação à comparação entre diferentes tipos de *embeddings* pré-treinados aplicados à diferentes algoritmos de aprendizagem de máquina, o trabalho de (SOUZA et al., 2021)

apresentou uma abordagem bem semelhante à utilizada nesta pesquisa. Foram utilizados cinco corpus contendo avaliações de produtos em Português-BR, onde quatro desses são os mesmos utilizados no presente trabalho. A comparação foi realizada entre a vetorização através de TF-IDF e dos vetores de *embeddings* pré-treinados *Glove*, *Word2Vec* e *FastText*, cada um considerando as dimensões 50, 100 e 300 como tamanho do vetor. Dois dos três algoritmos de aprendizagem utilizados por (SOUZA et al., 2021), também são iguais aos utilizados nessa pesquisa, a saber: Regressão Logistica e LGBM.

Entretanto, o principal foco do trabalho de (SOUZA et al., 2021) foi verificar o impacto da variação da dimensionalidade das diferentes formas de vetorização, além de comparar a técnica TF-IDF com diferentes tipos de *embedding*, mas não foram consideradas abordagens de combinações entre os métodos de vetorização. Ainda assim, devido à relevância e similaridade entre os trabalhos, alguns dos resultados obtidos por (SOUZA et al., 2021) podem ser utilizados como referências comparativas para esta pesquisa.

3.1 CONSIDERAÇÕES DO CAPÍTULO

Neste Capítulo foram apresentadas algumas das contribuições mais relevantes na área de PLN, principalmente no que diz respeito ao estudo de *embeddings*, desde a comparação entre diferentes técnicas e seus impactos em diferentes problemas de PLN, à estratégias de combinação entre técnicas já existentes, sempre com o intuito de fornecer o máximo de informação possível na forma de representação das palavras, uma vez que isso está diretamente ligado à melhorias de performance de qualquer problema relacionado à extração de informações textuais. É possível verificar a existência de uma vasta literatura associada a este tema, porém muitos estudos se limitam a novas abordagens de *embeddings* para tarefas de PLN mais relacionadas à reconhecimento de palavras e suas relações de similaridade. Do ponto de vista de classificação de sentimentos, em especial considerando sentenças escritas em Português, apesar dos avanços ainda é possível notar certa escassez de estudos voltados para as particularidades deste idioma.

Além disso, em relação à estratégias de combinações entre técnicas de *embedding*, ainda há muito a ser estudado, pois poucos trabalhos focam no desenvolvimento de combinações, o que pode ser justificado pela complexidade associada a esta tarefa, cujo esforço de implementação pode não ser proporcional ao ganho de performance obtido. Diante disso, o estudo comparativo deste trabalho também busca trazer contribuições para este pressuposto.

Assim, considerando os resultados obtidos nos trabalhos citados anteriormente, essa pesquisa se propõe a analisar os impactos da utilização de diferentes técnicas *embeddings* quando aplicadas ao contexto de Análise de Sentimentos voltado para a classificação, considerando múltiplas classes de textos escritos em Português-BR. Além da comparação entre vetores de *embeddings* individuais, três formas de combinação entre vetores serão consideradas: a concatenação, forma mais simples e comumente encontrada na literatura, e as duas abordagens de *meta-embeddings* apresentadas em (KIELA; WANG; CHO, 2018). O principal objetivo desta análise consiste em verificar se alguma dessas combinações se destaca quando comparadas à utilização de *embeddings* pré-treinados individualmente. Essa pesquisa também se propõe a avaliar o impacto da relação entre tipo de *embedding* considerado e modelo de aprendizagem de máquina utilizado, buscando mostrar se existe uma combinação ótima de tipo de *embedding* e modelo de AM dentro do conjunto de experimentos realizados.

4 METODOLOGIA

Este capítulo buscar mostrar uma visão sistemática do presente problema de pesquisa, detalhando suas principais etapas, a descrição das bases de dados utilizadas, bem como as técnicas de vetorização de palavras e algoritmos de classificação necessários para o estudo comparativo realizado.

4.1 ANÁLISE DO PROBLEMA

O problema de análise de sentimentos consiste essencialmente em um problema de classificação automática. Nesse sentido, na área de Aprendizagem de Máquina, inúmeros são os algoritmos de classificação que podem ser utilizados para detectar a polaridade de um texto. Desde algoritmos mais clássicos, que realizam a classificação a partir da separação do público em grupos com características semelhantes, como ocorre no caso do SVM e da Regressão Logística, a algoritmos que utilizam uma forma profunda de aprendizado a partir de redes neurais construídas de forma mais complexa.

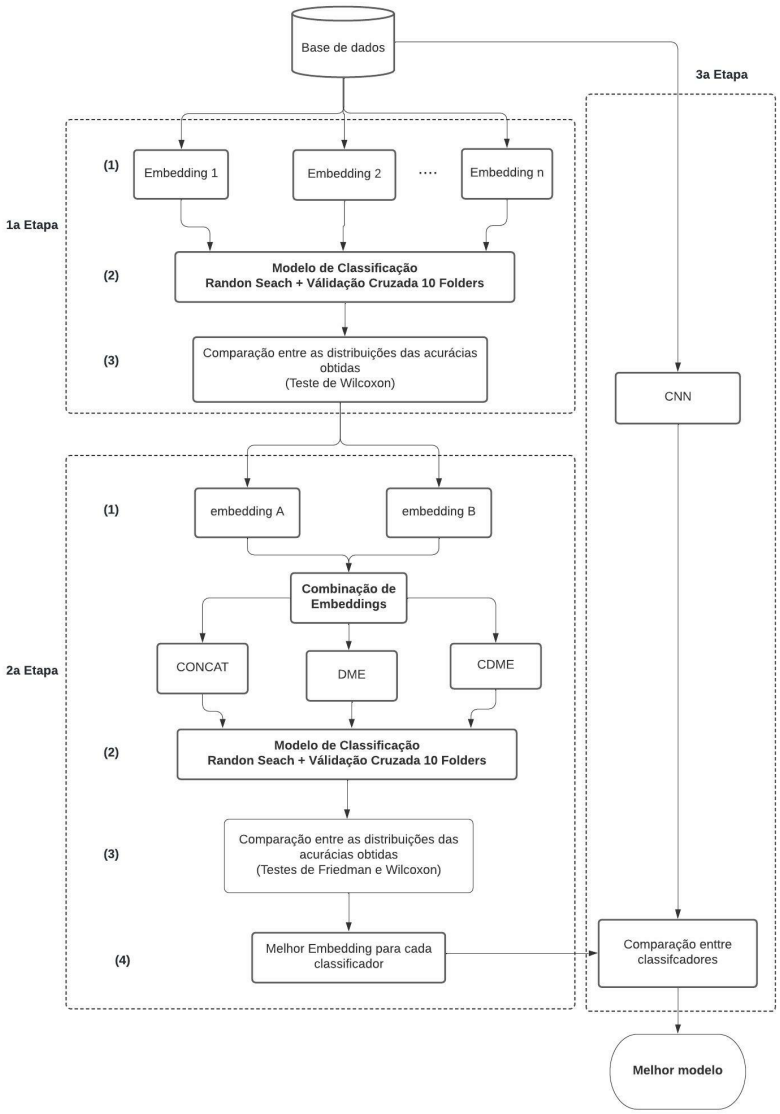
Independente do tipo de aprendizagem utilizado, uma etapa anterior à execução do algoritmo precisa necessariamente ser realizada quando os dados em questão são textos: a conversão das palavras em uma linguagem que possa ser compreendida pelas máquinas. Ou seja, os textos precisam passar por uma etapa de codificação que os transforme em números. Essa forma de codificação, chamada comumente de vetorização, é realizada de forma a manter o sentido do conteúdo textual, de modo que os modelos de aprendizagem de máquina consigam estabelecer relações e separações dos exemplos que serão treinados.

Como qualquer problema de AM, o desempenho de um modelo de classificação de sentimentos não depende apenas do tipo de algoritmo e sua configuração de parâmetros, mas também, e talvez principalmente, do poder de explicabilidade dos dados a serem utilizados. No caso de dados textuais, esse poder está diretamente relacionado à técnica de vetorização aplicada aos textos, de modo que, intuitivamente, espera-se que técnicas com maior potencial de conservação das características sintáticas e semânticas dos textos contribuam com a melhoria da performance dos modelos de classificação.

Diante deste cenário, este trabalho realiza um estudo comparativo no qual 7 (sete) formas de vetorização são aplicadas a 5 (cinco) algoritmos de aprendizagem de máquina distintos,

com o intuito de verificar se uma das técnicas de vetorização se destaca em todos os algoritmos ou se é possível estabelecer o melhor par de técnica de vetorização e algoritmo que consiga obter melhor performance nas 4 (quatro) bases de dados utilizadas, que foram obtidas a partir fontes distintas, mas são associadas ao mesmo tema: avaliação de produtos e serviços. Além disso, a outra parte da pesquisa consiste em analisar se a combinação entre duas técnicas de vetorização contribui para a melhoria de performance dos modelos de classificação e se a utilização de um *embedding* construído a partir dos pesos de uma rede de aprendizado profundo sempre consegue fornecer melhores resultados quando comparada à vetorização a partir de *embeddings* pré-treinados. Para isso, foi desenvolvida a arquitetura experimental mostrada na Figura 16 que será detalhada na próxima seção.

Figura 16 – Fluxo Experimental Desenvolvido



Fonte: a autora (2022).

Para mostrar o impacto da utilização das técnicas de *embedding* na etapa de vetorização de textos, foram realizados experimentos utilizando a técnica *Bag of Words*, que serão utilizados como *baseline* para discussão dos resultados obtidos desta pesquisa.

4.2 ARQUITETURA EXPERIMENTAL DESENVOLVIDA

O estudo comparativo realizado nesta pesquisa foi feito através de uma arquitetura composta por três etapas, cujos resultados experimentais buscam responder às questões de pesquisas citadas na introdução deste trabalho. Para cada uma das quatro bases de dados utilizadas, foi realizado um processo composto por três etapas: dois blocos de experimentos principais, que ocorrem de forma sequencial, uma vez que a segunda etapa depende dos resultados obtidos na etapa anterior, e um bloco experimental que ocorre em paralelo, que independe dos resultados das outras etapas para ser executado. O desenho esquemático do fluxo completo realizado é mostrado na Figura 16, cujas etapas serão detalhadas na sequencia.

A primeira etapa do processo (em destaque na Figura 16) consiste na avaliação dos n diferentes tipos de *embeddings* pré-treinados (onde n = 4: *Glove*, *Word2Vec*, *Wang2Vec* e *FastText*). Para isso, cada *embedding* é utilizado em um mesmo algoritmo de classificação, e as distribuições das acurácias dos modelos resultantes são então avaliados (Fase (3) da 1ª etapa) para realizar a escolha das duas técnicas de *embedding* que serão combinadas na etapa a seguir. As escolhas serão definidas por meio dos resultados do teste estatístico de Wilcoxon, conforme explicado com mais detalhes no Capítulo 5. Em seguida, são iniciados os experimentos envolvendo as combinações dos dois *embeddings* selecionados (Fases (1) e (2) da 2ª etapa na Figura 16): cada combinação é então aplicada ao mesmo algoritmo de classificação dos experimentos anteriores e agora os resultados a serem avaliados são os das classificações utilizando as diferentes formas de combinação de *embeddings* consideradas (Fase (3) da 2ª etapa).

Paralelamente a essas etapas (3ª etapa na Figura 16), é realizado um experimento utilizando uma CNN como algoritmo de classificação. A escolha deste algoritmo foi feita com o intuito de ter pelo menos um algoritmo de aprendizado profundo no rol de algoritmos classificadores explorados. Esta fase é executada de forma paralela pois independe das demais para ocorrer, porém o modelo obtido nela será comparado com os resultados provenientes das etapas anteriores, para que seja escolhido o melhor modelo para a classificação de cada base de dados. No experimento da 3ª etapa a vetorização dos textos é realizada na camada de

embedding do próprio modelo de aprendizado profundo, que calculará os melhores pesos para representar as palavras da base de dados que está sendo modelada.

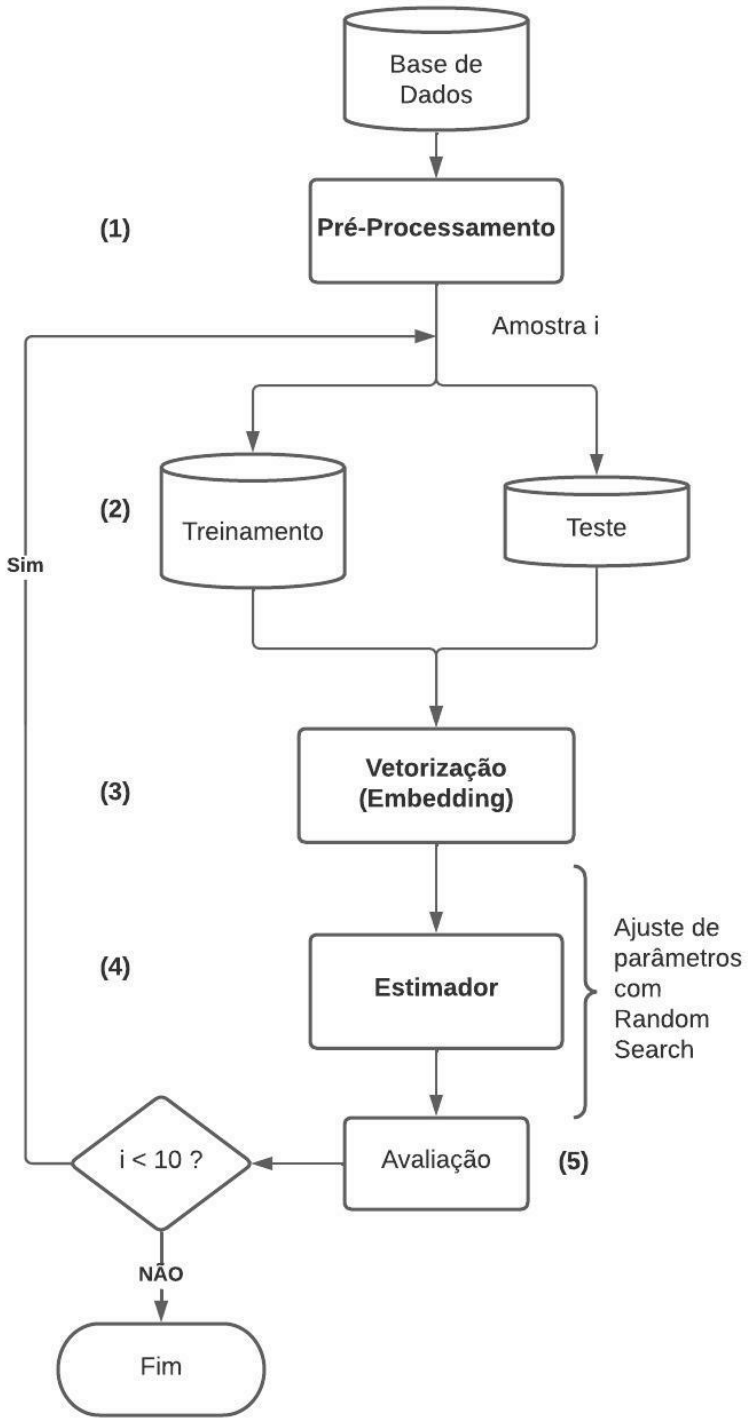
Por fim, é realizada a comparação entre todos os modelos que utilizaram *embeddings* simples e combinados (Fase (4) da 2ª etapa), encontrando-se a melhor forma de *embedding* para cada classificador. O resultado obtido através da CNN é então comparado aos demais classificadores (cada um com seu melhor *embedding*), para definição do melhor modelo final da base de dados analisada.

A seção a seguir descreve os detalhes do processo geral de modelagem realizado em cada experimento da arquitetura.

4.3 ETAPAS DO PROCESSO DE CLASSIFICAÇÃO

O fluxo de modelagem utilizado em cada experimento realizado durante a pesquisa possui as cinco etapas principais comuns a qualquer problema de análise de sentimentos, conforme mostrados na Figura 17, a saber: pré-processamento dos dados (1), divisão da base de dado em treinamento, validação e teste (2), vetorização dos dados textuais (3), aplicação dos modelos de aprendizagem de máquina (4) e avaliação dos resultados (5). O processo exibido na 17 corresponde a uma visão mais detalhada da fase de modelagem do fluxo proposto na seção anterior, que é representada pelos blocos "Modelo de Classificação"(Fases (2) das 1ª e 2ª etapas) e "CNN"(da 3ª etapa) do esquema experimental mostrado na Figura 16). Para fins de comparação entre os modelos, cada experimento foi realizado 10 vezes. Assim, na Figura 17, "i"representa a iteração de cada experimento, ou seja, cada base foi dividida aleatoriamente em 10 partes e o mesmo experimento foi treinado em cada uma delas e a performance total do modelo foi dada pela média de cada uma das 10 performances individuais. Essas performances foram obtidas a partir da base de teste em cada iteração.

Figura 17 – Fluxo do Processo de Modelagem Utilizado

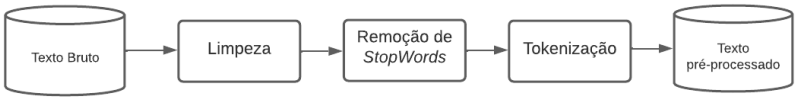


Fonte: a autora (2022).

Nessa pesquisa, a etapa de pré-processamento consistiu na limpeza dos dados (remoção de números, caracteres especiais, links de sites e remoção de espaços em branco), seguida pela remoção das palavras consideradas irrelevantes ou dispensáveis para a compreensão do conteúdo do texto (*stopwords*). O último passo dessa etapa consiste no processo de tokeni-

zação dos textos da base de dados, ou seja, a divisão do texto em unidades simples (*tokens*). Nos experimentos deste trabalho foi utilizado o método *Tokenizer* presente no módulo de pré-processamento de textos da biblioteca *Keras* (Tensorflow)[1]. Este método realiza a associação de cada palavra (ou outra unidade textual que esteja sendo considerada como *token*) a um índice (número inteiro), correspondente à posição daquela palavra no dicionário obtido a partir de todo vocabulário da base de dados. As fases desse processo são mostradas na Figura 18.

Figura 18 – Transformações realizadas durante a etapa de Pré-processamento.



Fonte: a autora (2022).

O próximo passo consistiu na etapa de vetorização das palavras. Para isso, este trabalho considerou considerou quatro diferentes tipos de *word embeddings* pré-treinados, obtidos através do repositório do Núcleo Interinstitucional de Linguística Computacional [2]. Esse repositório disponibiliza gratuitamente vetores gerados a partir de um grande corpus com textos escritos em Português do Brasil e Português europeu, construído a partir de diversas fontes de dados e diferentes tipos de gêneros textuais. Foram utilizadas 17 fontes de dados para a construção dos *embeddings*, cuja descrição dos tipos de informações utilizadas são mostradas na Figura 19.

[1] <https://www.tensorflow.org/guide/keras?hl=pt-br>
[2] <http://www.nilc.icmc.usp.br/embeddings>

Figura 19 – Informações dos corpus utilizados no pré-treinamento de *Word Embeddings*

Corpus	Tokens	Types	Genre	Description
LX-Corpus [Rodrigues et al. 2016]	714,286,638	2,605,393	Mixed genres	A huge collection of texts from 19 sources. Most of them are written in European Portuguese.
Wikipedia	219,293,003	1,758,191	Encyclopedic	Wikipedia dump of 10/20/16
GoogleNews	160,396,456	664,320	Informative	News crawled from GoogleNews service
SubIMDB-PT	129,975,149	500,302	Spoken language	Subtitles crawled from IMDb website
G1	105,341,070	392,635	Informative	News crawled from G1 news portal between 2014 and 2015.
PLN-Br [Bruckschen et al. 2008]	31,196,395	259,762	Informative	Large corpus of the PLN-BR Project with texts sampled from 1994 to 2005. It was also used by [Hartmann 2016] to train word embeddings models
Literacy works of public domain	23,750,521	381,697	Prose	A collection of 138,268 literary works from the Domínio Público website
Lacio-web [Aluísio et al. 2003]	8,962,718	196,077	Mixed genres	Texts from various genres, e.g., literary and its subdivisions (prose, poetry and drama), informative, scientific, law, didactic technical
Portuguese e-books	1,299,008	66,706	Prose	Collection of classical fiction books written in Brazilian Portuguese crawled from Literatura Brasileira website
Mundo Estranho	1,047,108	55,000	Informative	Texts crawled from Mundo Estranho magazine
CHC	941,032	36,522	Informative	Texts crawled from Ciência Hoje das Crianças (CHC) website
FAPESP	499,008	31,746	Science Communication	Brazilian science divulgation texts from Pesquisa FAPESP magazine
Textbooks	96,209	11,597	Didactic	Texts for children between 3rd and 7th-grade years of elementary school
Folhinha	73,575	9,207	Informative	News written for children, crawled in 2015 from Folhinha issue of Folha de São Paulo newspaper
NILC subcorpus	32,868	4,064	Informative	Texts written for children of 3rd and 4th-years of elementary school
Para Seu Filho Ler	21,224	3,942	Informative	News written for children, from Zero Hora newspaper
SARESP	13,308	3,293	Didactic	Text questions of Mathematics, Human Sciences, Nature Sciences and essay writing to evaluate students

Fonte: (HARTMANN et al., 2017).

Entretanto, antes da aplicação dos *embeddings* pré-treinados, é necessário transformar cada exemplo da base de dados em uma sequencia numérica, que é obtida a partir da transformação dos *tokens* oriundos do pré-processamento. Cada avaliação (exemplo) da base de dados é transformada em um vetor formado pelos índices de cada palavra da sentença de exemplo. Obviamente, cada avaliação terá uma quantidade de palavras, de modo que os vetores inicialmente serão de tamanhos distintos, correspondentes à quantidade de *tokens* daquele exemplo.

Desse modo, é necessário fixar um tamanho de vetor para todos os exemplos da base de dados, o que é feito acrescentando uma quantidade de zeros sobre cada vetor original para chegar ao tamanho desejado (se o exemplo tiver mais *tokens* que o tamanho fixado, o texto será truncado). A Tabela 5 mostra o percentual de exemplos de cada base de dados com mais de 100 palavras, considerando esses valores, foi fixado o tamanho máximo de representação das avaliações igual a 100, uma vez que boa parte dos comentários não ultrapassava essa quantidade de *tokens*, de modo que fosse possível capturar o máximo de informação dos comentários de cada base.

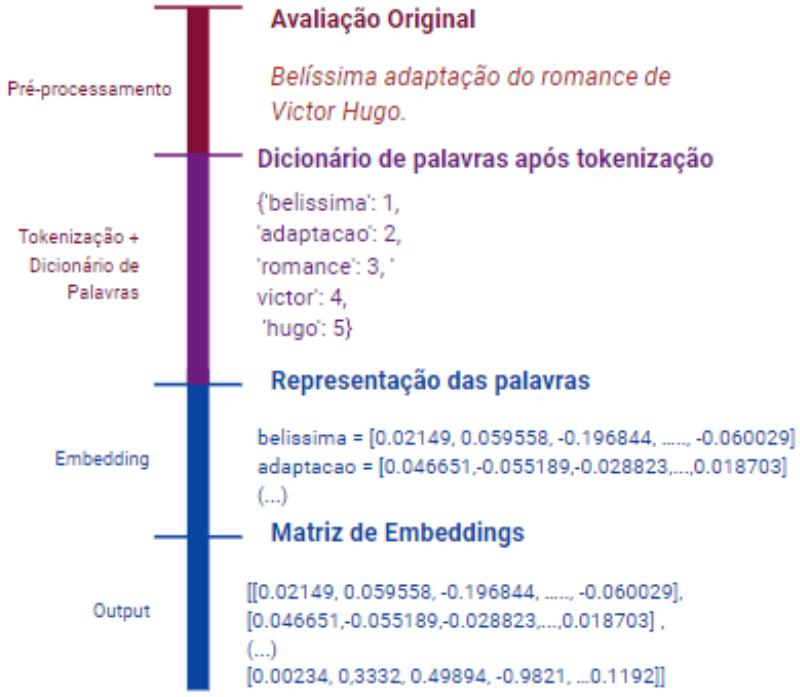
Tabela 5 – Representatividade dos tamanhos das sentenças por base de dados

Dataset	% de sentenças com mais de 100 tokens
Olist	0,2 %
Buscapé	2,8 %
B2W Reviews	0,3 %
UTLC Movies	2,9 %

Fonte: a autora (2022)

Fixado o tamanho máximo de cada exemplo, inicia-se o processo de busca de cada palavra da avaliação na matriz de *embeddings* pré-treinados. Nessa matriz a primeira posição de cada linha é uma palavra enquanto as demais são os valores do vetor de *embedding* que representa aquela palavra. A Figura 20 apresenta todas as etapas envolvidas no processo de codificação de um exemplo: o retorno desse processo corresponde a uma matriz de dimensão (n x d) onde n representa a quantidade total de *tokens* daquele exemplo e d a dimensão do *embedding* que representa cada palavra. Finalmente, a representação da avaliação completa é obtida através da média entre os vetores de *embedding* de cada palavra, uma vez que cada avaliação precisa ser representada por um vetor de dimensão fixa d.

Figura 20 – Etapas da vetorização utilizando *Embeddings* pré-treinados



Fonte: a autora (2022).

Os tipos de *embedding* utilizados nos experimentos foram todos com dimensão d = 100. No início da pesquisa foram realizadas algumas experimentações para a escolha da dimensão dos *embeddings*, considerando os vetores com dimensões 50, 100 e 300. Os resultados desses testes mostraram que os *embeddings* de tamanho 100 em geral forneceram melhores resultados nas classificações quando comparados aos experimentos realizados com d = 50. Ao aumentar a dimensionalidade dos vetores para d = 300, não foram obtidas melhorias significativas, mas em compensação o tempo de processamento dos experimentos se tornou bem maior, devido ao aumento da dimensionalidade, de modo que a possibilidade de utilização desse tamanho de vetor foi desconsiderada.

Além da dimensionalidade, os modelos *Word2Vec*, *Wang2Vec* e *FastText* também possuem variações nas formas de obtenção dos *embeddings*, que podem ser construídos a partir da arquitetura CBOW ou *Skip-Gram*. Devido à diversidade dos comentários que podem ser encontrados no cenário de avaliação de produtos, optou-se por utilizar os modelos baseados na técnica *Skip-Gram*, uma vez que essa arquitetura tende a apresentar bons resultados na representação de palavras mais raras.

Finalmente, após a etapa de vetorização, os dados já estão representados de forma a serem utilizados para o treinamento dos algoritmos de AM, que ocorrem na etapa "Estimador"(4) ilustrada na Figura 17. Para fins de confiabilidade e reprodutibilidade dos resultados, o processo de treinamento e validação foi realizado a partir da divisão das base de dados em 10 partes, de modo que os resultados finais de cada métrica de avaliação foram dados a partir da média dos resultados obtidos em cada uma das 10 iterações (processo de validação cruzada com k = 10). Em cada iteração, foi realizada a divisão daquela amostra em dados treino e teste, que passavam pela etapa de vetorização e seguiam para a aplicação do estimador (modelo de classificação).

A etapa de classificação foi realizada através do método *RandomizedSearchCV* presente na biblioteca *scikit-learn*. Este método realiza uma busca aleatória de valores para os principais parâmetros do modelo, com o objetivo de encontrar o conjunto de parâmetros que forneça a melhor performance do modelo em relação a determinada métrica. No caso dos experimentos deste trabalho, a métrica considerada foi a acurácia. O processo de busca randômica é então executado n vezes, sendo n a quantidade de iterações definida. Nos experimentos realizados nesse trabalho foi utilizado n=10 para todos os estimadores. Além disso, em cada busca foi realizada uma validação cruzada com k = 5 *folders*. Assim, para cada chamada da aplicação do *RandomSearch*, foram realizadas 5x10 = 50 iterações no total. A Tabela 6 mostra a lista

de parâmetros que foram tunados via *RandomSearch* para cada algoritmo utilizado.

Tabela 6 – Parâmetros Utilizados na Busca Randômica

Algoritmo	Parâmetros de Busca
SVM	kernel, tol, C, gamma, max_iter, decision_function_shape
Regressao Logistica	solver, penalty, tol, C, max_iter, multi_class
MLP	max_iter, hidden_layer_sizes, activation, solver, alpha, learning_rate,tol
LGBM	boosting, num_iterations, reg_lambda, bagging_fraction, bagging_freq, num_leaves, max_depth, max_bin, learning_rate, objective
XGBoost	max_depth, learning_rate, n_estimators, booster, gamma, min_child_weight, subsample, colsample_bytree, reg_alpha, reg_lambda, objective
CNN	num_filters, activation, kernel_size, batch_size, epochs, optimizer

Fonte: a autora (2022)

4.4 BASES DE DADOS UTILIZADAS

Para realização dos experimentos deste trabalho, foram selecionadas quatro bases de dados para a análise de sentimentos. Foram realizados experimentos considerando o problema de classificação com múltiplas classes, mas especificamente três: avaliações positivas, negativas e neutras. Todas as bases foram encontradas através do Kaggle[3], uma comunidade online que reúne diversos fóruns, dados e competições relacionadas à Ciência de Dados. Com o objetivo de analisar sentimentos associados a um mesmo tipo de tema, as bases escolhidas são todas formadas a partir de avaliações online em sites de produtos e serviços, a saber: Buscapé, B2W, Olist e UTLC-*movies*.

Outro aspecto comum aos dados utilizados é o fato de que a avaliação dos produtos foi feita através da escala visual de estrelas, que normalmente varia de 1 a 5, onde 1 representa o máximo de insatisfação e 5 o máximo de satisfação pelo produto. Os valores dessa escala são utilizados como rótulos para cada comentário de modo que, com o objetivo de aumentar a representatividade de cada classe e simplificar o processo de classificação, os valores de 1 a 5 foram redistribuídos em 3 classes, mostradas na Tabela 7:

Tabela 7 – Regra de redistribuição para as avaliações das bases de dados

Avaliação do Produto	Classe do Comentário
4 ou 5 estrelas	Classe 2
1 ou 2 estrelas	Classe 0
3 estrelas	Classe 1

Fonte: a autora (2022)

[3] <https://www.kaggle.com/fredericods/ptbr-sentiment-analysis-datasets>

Considerando essa regra de redistribuição, as classes 0 e 2 representam os comentários negativos e positivos, respectivamente. Enquanto a classe 1 foi associada arbitrariamente aos comentários de teor neutro, apenas para fins de separação dos dados em classes distintas.

4.4.1 Buscapé Corpus

O corpus Buscapé foi obtido e apresentado por (HARTMANN et al., 2014) e é composto originalmente por 85.910 avaliações extraídas a partir da base de dados do site Buscapé [4] em Setembro de 2013. A base é composta por avaliações dos produtos das seguintes categorias, em ordem de representatividade: TV, celulares e *smartphones*, câmeras digitais, perfumes, jogos, aparelhos de ar-condicionado, notebooks e tabletes. As notas dadas pelos consumidores variavam originalmente de 0 a 5, porém os exemplos com notas iguais a 0 foram removidos da base utilizada neste trabalho, devido à baixa representatividade deste rótulo (menos que 1% dos casos), assim como ocorreu para outras bases utilizadas neste trabalho. A Tabela 8 mostra a quantidade de exemplos por nota antes da redistribuição em 3 classes.

Tabela 8 – Quantidade de exemplos por classe original - Buscapé Corpus

classe	1	2	3	4	5
qtd exemplos	3.043	3.622	11364	33577	33239

Fonte: a autora (2022)

Após a etapa de pré-processamento dos textos esse corpus apresenta um vocabulário total de 58.475 *tokens*. Considerando a remoção dos exemplos com nota 0 e a redistribuição dos exemplos em 3 classes, a volumetria das 84.990 avaliações restantes é rotulada conforme mostrado na Tabela 9.

Tabela 9 – Quantidade de exemplos por classe após redistribuição - Buscapé Corpus

classe	0	1	2
qtd exemplos	6.810	11.364	66.816

Fonte: a autora (2022)

É fácil notar que a quantidade de avaliações positivas é bem superior às negativas e neutras, implicando em um desbalanceamento entre as classes - um padrão que ocorre também para as demais bases utilizadas. Essa característica é refletida na nuvem de palavras formada pelo

[4] <https://www.buscape.com.br/>

vocabulário total do corpus Buscapé mostrada na Figura 21, que mostra a forte presença de palavras com teor positivo, como "benefício", "bom"e conjugações do verbo "gostar".

Figura 21 – Nuvem de Palavras - Buscapé Corpus

Fonte: a autora (2022).

4.4.2 B2W Reviews

O grupo B2W *Marketplace* é uma empresa responsável pelo comércio eletrônico de três lojas virtuais de grande destaque no Brasil: *americanas.com*, Submarino e *Shoptime*. O corpus com avaliações dos produtos vendidos por essas lojas virtuais foi lançado por (REAL; OSHIRO; MAFRA, 2019) e possui mais de 130.000 avaliações escritas em Português-BR. Além do comentário sobre o produto e a nota dada pelo cliente (também no sistema de classificação por estrelas que varia de 1 a 5), esta base também apresenta outra informação que pode ser usada como rótulo de classificação e indica se o cliente recomenda (ou não) aquele produto para um amigo, sendo por tanto um rótulo binário. Como esta pesquisa foca em classificação não-binária, esse rótulo foi desconsiderado.

A base utilizada para os experimentos apresenta um total de 131.072 exemplos, distribuídos inicialmente nas cinco classes mostradas na Tabela 10. Após a redistribuição em três classes, as novas representatividades correspondem aos valores mostrados na Tabela 11.

Tabela 10 – Quantidade de exemplos por classe original - B2W *Reviews*

classe	1	2	3	4	5
qtd exemplos	27.099	8.307	16.162	32.026	47.478

Fonte: a autora (2022)

Tabela 11 – Quantidade de exemplos por classe após redistribuição - B2W *Reviews*

classe	0	1	2
qtd exemplos	35.406	16.162	79.504

Fonte: a autora (2022)

A Figura 22 exibe as palavras mais frequentes no vocabulário desse corpus, que possui mais de 50 mil *tokens* após a etapa de pré-processamento dos dados.

Figura 22 – Nuvem de Palavras - B2W *Reviews*

Fonte: a autora (2022).

4.4.3 Olist Dataset

Assim como a B2W, a Olist também corresponde a uma gigante do *Marketplace* para vendas em lojas virtuais como o Mercado Livre, Extra, Carrefour, Amazon, dentre outras lojas de departamento. Em 2018, a própria empresa disponibilizou gratuitamente no Kaggle o *Brazilian E-Commerce Public Dataset*, um conjunto de dados contendo informações sobre pedidos realizados por suas lojas virtuais entre 2016 e 2018, correspondendo a cerca de 100 mil pedidos [5]. Dentre as informações disponibilizadas, estão o *ranking* com o nível de satisfação do cliente na compra (entre 1 e 5 estrelas) e os comentários referentes aos produtos dos diferentes *marketplaces* parceiros da empresa.

A base de dados correspondente às avaliações de produtos utilizados neste trabalho apresenta 41.744 exemplos, divididos conforme as classes mostradas na Tabela 12.

[5] <https://www.kaggle.com/olistbr/brazilian-ecommerce>

Tabela 12 – Quantidade de exemplos por classe original - Olist

classe	1	2	3	4	5
qtd exemplos	9.179	2.229	3.665	6.031	20.640

Fonte: a autora (2022)

Após a etapa de pré-processamento, o vocabulário total da base passou a possuir cerca de 15 mil *tokens*. A quantidade de exemplos por classe após a transformação das 5 separações iniciais em 3 classes corresponde aos valores presentes na Tabela 13.

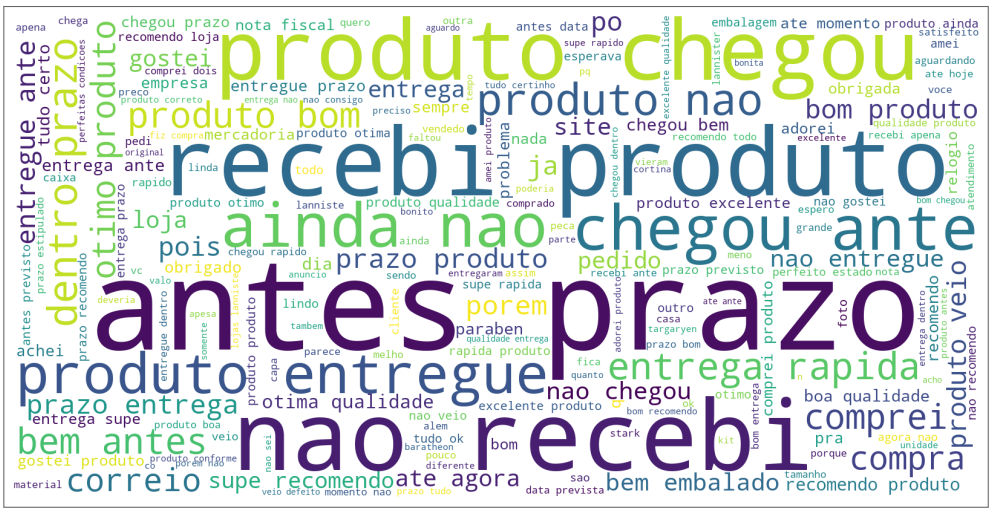
Tabela 13 – Quantidade de exemplos por classe após redistribuição - Olist

classe	0	1	2
qtd exemplos	11.408	3.665	26.671

Fonte: a autora (2022)

Assim como nos demais corpus referentes à avaliações de produtos, as principais palavras presentes no vocabulário dessa base também foram associadas à experiência de compras online. Sendo os termos mais comuns "recebi", "prazo", "produto", "chegou"e os demais presentes na Figura 23.

Figura 23 – Nuvem de Palavras - Olist



Fonte: a autora (2022).

4.4.4 UTL-Corpus

Assim como o corpus de avaliações do Buscapé, o corpus UTL (abreviado como UTLC) também foi disponibilizado através do repositório do Opinando, um grupo de pesquisa fundado

pela Universidade de São Paulo (USP) que busca contribuir com pesquisas relacionadas a diversos problemas de PLN envolvendo o Português Brasileiro [6].

Construído por (SOUSA; BRUM; NUNES, 2019), esse é o maior conjunto de dados dentre os utilizados na pesquisa, possuindo mais de 2 milhões de avaliações no total. Ele é formado por dois grandes corpus: UTLC-*movies*, que contém as avaliações de filmes coletados do site *Filmow*, e o UTLC-*Apps*, formado por avaliações de aplicativos para *smartphones*, obtidos a partir da *Google Play Store*.

Inicialmente, foram realizados experimentos com os dois corpus, porém os experimentos envolvendo o UTLC-*Apps* apresentavam tempo de processamento significativamente superior aos que utilizavam UTLC-*movies*. Uma vez que em cada base de dados foram realizados mais de 30 experimentos, considerando as diferentes combinações de algoritmo e técnica de *embedding*, não foi possível concluir todos os experimentos propostos para o UTLC-*Apps*, de modo que os resultados serão apresentados apenas para o UTLC-*movies*.

A base de dados do UTLC-*movies* utilizada apresenta 131.072 avaliações referentes a mais de 4 mil filmes considerados os mais populares do mundo (SOUSA; BRUM; NUNES, 2019). A nuvem de palavras mostrada na Figura 24 mostra a intensidade das palavras presentes nas avaliações. Os termos associados ao universo cinematográfico, como "história", "personagem"e "trilha sonora", aparecem em grande frequência.

Figura 24 – Nuvem de Palavras - UTLC-*movies*

Fonte: a autora (2022).

Assim como as avaliações de produtos, os filmes normalmente são avaliados utilizando o sistema de notas através das quantidades de estrelas, que variam entre 0 e 5. Novamente,

[6] <https://sites.google.com/icmc.usp.br/opinando/>

para os fins de classificação realizados nesse trabalho, os filmes avaliados com 0 estrelas foram desconsiderados e as notas de 1 a 5 foram redistribuídas em três rótulos de classificação, cujas quantidades são mostradas na Tabela 14.

Tabela 14 – Quantidade de exemplos por classes após redistribuição - UTLC-*movies*

classe	0	1	2
qtd exemplos	12.089	26.287	92.696

Fonte: a autora (2022)

Após o processo de limpeza e demais etapas do pré-processamento de dados, o total de palavras presentes no vocabulário da base foi de 82.350 *tokens*. A forma de tokenização aplicada aos dados do UTLC-*movies* considerou a separação dos textos em *uni-grams* (cada *token* corresponde apenas a uma palavra da avaliação), o que também foi realizado para os demais corpus utilizados nessa pesquisa.

4.5 CONSIDERAÇÕES DO CAPÍTULO

Neste capítulo foi abordada a arquitetura de experimentação construída para realização dos estudos dos diferentes *embeddings* considerados, bem como as formas de combinação de *embeddings* propostas.

As bases de dados que foram descritas neste Capítulo, apesar de possuírem o tema comum de avaliação de produtos, apresentam vocabulários bem distintos, conforme mostrados pelas nuvens de palavras geradas para cada fonte de dados. Além disso, possuem grandes diferenças de volumetria, como o Olist *Dataset*, que apresenta menos de 50 mil exemplos e o UTLC-*Movies*, com mais de 130 mil avaliações. Essas diferenças são essenciais para qualquer estudo comparativo.

Além disso, foi mostrada a estratégia de realização do fluxo experimental proposto, composto por três partes, onde cada uma delas será necessária para responder as questões de pesquisas levantadas neste trabalho. A primeira etapa busca verificar o impacto da aplicação de diferentes técnicas de *embedding* quando utilizadas com o mesmo classificador e apontar quais delas apresentam maiores distinções estatísticas entre si. Já a etapa seguinte tem como principal objetivo a avaliação de três formas de combinação entre vetores de *embeddings* gerados por técnicas distintas e seu impacto nos algoritmos de classificação utilizados. Espera-se

que essa combinação traga ganhos de performance para os classificadores e busca-se analisar se alguma delas apresenta maior destaque para os dados de análise.

Por fim, a última etapa do processo visa comparar os melhores resultados obtidos nas duas primeiras etapas com uma classificação realizada por um algoritmo de aprendizado profundo (CNN) cujos pesos dos vetores de *embedding* são calculados em uma das camadas da própria rede. De forma geral os resultados encontrados na literatura indicam que as redes de aprendizado profundo performam melhor que algoritmos de AM clássicos, e esta etapa do experimento busca analisar se essa hipótese realmente ocorre para as bases de dados utilizadas. No Capítulo a seguir são descritos os resultados obtidos a partir desta metodologia de pesquisa, bem como as discussões acerca dos mesmos.

5 EXPERIMENTOS E RESULTADOS

Este capítulo busca responder às questões de pesquisa apontadas na Introdução deste trabalho. Para isso, serão apresentados os resultados obtidos para os trinta e seis modelos de classificação realizados para cada uma das quatro bases de dados analisadas. A execução desses modelos foi realizada em duas etapas: a primeira consistiu em verificar o impacto dos diferentes vetores de *embeddings* pré-treinados sendo executados para cada um dos cinco classificadores de AM utilizados. A segunda etapa consistiu na análise das três diferentes formas de combinação de *embeddings* que estão sendo consideradas nessa pesquisa.

As combinações foram realizadas apenas entre duas técnicas de *embedding* distintas, escolhidas a partir dos resultados da comparação entre as distribuições das acurácias de cada modelo para cada possível combinação de *embeddings* através do teste estatístico de Wilcoxon. As combinações consideradas foram: (*Glove* e *Word2Vec*), (*Glove* e *Wang2Vec*), (*Glove* e *FastText*), (*Wang2Vec* e *FastText*), (*Word2Vec* e *FastText*), ou seja: todas as combinações dois a dois possíveis, com exceção da (*Word2Vec* e *Wang2Vec*), devido à similaridade entre estas técnicas de vetorização.

Para avaliar e comparar os resultados dos diferentes *embeddings* aplicados à mesma técnica de AM, utilizou-se o teste estatístico de Friedman, uma vez que as comparações envolviam mais de duas amostras por vez (ver seção 2.7 do Capítulo 2), para cada uma das técnicas de *embeddings* consideradas. As amostras utilizadas tanto para a realização dos teste de Friedman como de Wilcoxon consistiram nas acurácias obtidas através da validação cruzada com $k = 10$ *folders*. Uma vez que ambos os testes utilizados são não-paramétricos, não foi necessário verificar a normalidade das distribuições consideradas. Em relação à confiabilidade dos resultados de cada teste, é considerado um nível de confiança de 95%, ou seja, as hipóteses nulas foram rejeitadas apenas para p-valores maiores que 0.05.

Além disso, para cada base de dados utilizada e cada algoritmo de AM considerado nesse estudo, realizou-se a classificação utilizando o método *Bag of Words* como técnica de vetorização das palavras. Esses resultados serão utilizados como uma informação comparativa para avaliar o impacto da aplicaçã de formas mais complexas de vetorizar textos, como é o caso dos *embeddings* e as combinações propostas neste trabalho.

Por fim, além das comparações entre *embeddings*, também foi analisada a comparação entre a performance do modelo obtido através de uma CNN com a camada de **embedding**

realizada pela própria rede neural e os melhores modelos de AM obtidos após as duas primeiras etapas experimentais. Além de testes estatísticos, também foi analisado o impacto do tempo de execução de cada modelo final. Para obtenção tempo de execução aproximado de cada modelo, foi realizada com uma execução única (1-*fold*) de uma amostra de tamanho fixo de cada base em um notebook com processador Intel(R) Core(TM) i5-8265U com 16GB de memória RAM e 1.80 GHz.

5.1 ANÁLISE DOS RESULTADOS PARA O OLIST *DATASET*

Esta seção exibe os resultados experimentais obtidos para as avaliações presentes no Olist *Dataset* e está dividida em três subseções, onde a primeira consiste na análise dos resultados considerando os diferentes tipos de *embeddings* sem combinações, aqui chamados de "*embeddings* simples", enquanto a segunda discorre sobre os resultados dos *embeddings* combinados. A última subseção, por sua vez, realiza a comparação do resultado obtido pela CNN com os dos demais classificadores de AM utilizados.

5.1.1 Comparação entre embeddings simples

Conforme descrito na Introdução deste trabalho, uma das perguntas que se busca responder ao realizar a aplicação de diferentes tipos de *embeddings*, fixando-se o algoritmo de classificação e a base de dados, é se o tipo de vetorização dos dados impacta de forma significativa na performance de um modelo de classificação de textos. Considerando a acurácia como métrica de performance, foi possível observar diferenças entre as médias de acurácia para um mesmo tipo de classificador quando a técnica de *embedding* era modificada. Esses resultados estão sumarizados na Tabela 15, destacando a melhor acurácia obtida para cada classificador, em relação à base de teste.

Considerando os dados do Olist *Dataset*, é fácil notar que, em relação à acurácia, o classificador que apresentou as piores performances foi o que teve maiores impactos com a mudança na técnica de *embedding*, que neste caso ocorreu para o SVM. Para os demais classificadores, a diferença entre as acurácias não foram tão significativas, mas é possível observar que o pior resultado obtido para eles ocorreu quando o *Word2Vec* foi utilizado na vetorização. As melhores performances foram conquistadas pela MLP e pelo *XGBoost*, este último apresentando o melhor de todos os resultados quando o *embedding Wang2Vec* foi utilizado.

Tabela 15 – Acurácia média e desvio padrão dos classificadores por *Embedding* (Olist *Dataset* - dados de teste)

Embedding	SVM		RL		MLP		LGBM		XGB	
	acc (%)	std	acc (%)	std	acc (%)	std	acc (%)	std	acc (%)	std
Glove	55.76	0.111	74.58	0.007	81.41	0.005	78.15	0.007	81.60	0.005
Wang2Vec	43.37	0.177	**75.61**	0.004	**81.60**	0.008	78.27	0.008	**82.05**	0.002
Word2Vec	51.72	0.134	73.78	0.007	80.32	0.009	75.64	0.015	81.14	0.004
FastText	**63.51**	0.053	74.78	0.007	80.71	0.008	**78.33**	0.009	81.89	0.004

Fonte: a autora (2022)

Para verificar se a diferença entre essas acurácias é estatisticamente significativa, foi aplicado o teste de Friedman sobre cada classificador com o fim de comparar se a mudança na técnica de *embedding* de fato impactou na distribuição das acurácias de cada modelo. Os resultados dos testes são apresentados na Tabela 16 e mostram que a hipótese nula, de que as distribuições são iguais para os diferentes tipos de *embeddings*, só pode ser rejeitada para o classificadores *XGBoost* e Regressão Logística (RL). Isso significa que, mesmo diante dos diferentes valores de acurácia obtidos para o SVM, não é possível afirmar que a mudança no tipo de *embedding* de fato impactou na performance deste classificador.

Tabela 16 – Resultados do teste de Friedman para os diferentes *embeddings* (Olist *Dataset* - dados de teste)

classificador	*p-value*	*p-value* $< \alpha$	resultado
SVM	0.1059	não	*não é possível rejeitar* H_0
RL	0.0034	sim	*rejeita-se* H_0
MLP	0.1176	não	*não é possível rejeitar* H_0
LGBM	0.0503	não	*não é possível rejeitar* H_0
XGBoost	0.0277	sim	*rejeita-se* H_0

Fonte: a autora (2022)

A partir dos resultados obtidos para cada *embedding* individual, a etapa de combinação de *embeddings* foi definida, conforme mostrado na seção a seguir.

5.1.2 Comparação entre combinações de embeddings

Para escolher quais seriam os *embeddings* a serem combinados, foi utilizado como critério a diferença entre as distribuições de cada par de *embedding* considerado, visando garantir a diversidade das formas de vetorização combinadas, uma vez que o principal objetivo de uma combinação de técnicas é a melhora de performance em relação à uma técnica aplicada

isoladamente a partir da combinação entre as diferenças de cada técnica. As combinações foram realizadas em pares devido ao tempo de execução associado ao aumento da dimensionalidade. Diante disso, realizou-se o teste de Wilcoxon para cada dupla de *embedding*, cujos resultados dos p-valores obtidos são mostrados na Tabela 17.

Tabela 17 – p-valores obtidos nos testes de Wilcoxon realizados para cada par de *embedding* (Olist *Dataset* - dados de teste)

Combinação	Classificador				
	SVM	RL	MLP	LGBM	XGBoost
Glove x Wang2Vec	0.1050	0.0184	0.3381	0.5000	0.0866
Glove x Word2Vec	0.5000	0.0718	0.0718	0.0184	0.0718
Glove x FastText	0.1050	0.3381	0.0718	0.3381	0.3381
Wang2Vec x FastText	0.0184	0.0375	0.1050	0.4173	0.2017
Word2Vec x FastText	0.1481	0.0301	0.3381	0.0184	0.0108

Fonte: a autora (2022)

Os resultados destacados são os casos em que o p-valor obtido para a comparação entre as distribuições de cada par de *embedding* para determinado classificador foi < 0.05, ou seja, onde é possível rejeitar a hipótese de que as distribuições são iguais. O par de *embedding* a ser escolhido foi determinado pelo que tivesse a maior quantidade de p-valor < 0.05 para os diferentes classificadores, o que ocorreu para os *embeddings* *Word2Vec* e *FastText*, conforme destacado na Tabela 17.

Uma vez escolhidos os *embeddings*, foram realizadas as classificações de cada algoritmo para os três tipos de combinações de *embedding* considerados. Os resultados de acurácia e tempo de processamento obtidos para classificador são exibidos na Tabela 18, onde é possível notar que a forma de combinação mais simples, que consiste apenas na concatenação dos vetores de cada *embedding*, trouxe a melhor acurácia para todos os classificadores. Por outro lado, esta também é a técnica com maior tempo de processamento quando comparada às demais combinações, com exceção do *XGBoost*, onde a combinação CDME apresentou maior tempo de preprocessamento.

Assim como a comparação entre os *embeddings* simples, no caso das combinações também foi realizado o teste de Friedman para verificar se as diferenças entre as performances são estatisticamente relevantes para as diferentes formas de combinações de *embeddings*. Os resultados são mostrados na Tabela 19 e indicam que as três formas de combinação consideras só podem ser distintas para o caso da Regressão Logística, da MLP e do *XGBoost*. Para esses

Tabela 18 – Acurácia e tempo de processamento médio para cada modelo (Olist *Dataset* - dados de teste)

Embedding	**SVM**		**RL**		**MLP**		**LGBM**		**XGB**	
	Acc (%)	Tempo (min)	Acc (%)	Tempo (min)	Acc (%)	Tempo (min)	Acc (%)	Tempo (min)	Acc (%)	Tempo (min)
concat	**58.35**	10.74	**76.66**	19.59	**82.34**	74.74	**78.27**	61.07	**82.26**	768.28
dme	46.05	7.48	75.21	13.19	80.56	27.96	77.41	42.20	81.75	387.15
cdme	53.62	6.04	75.02	12.10	80.46	40.77	77.14	40.64	81.78	976.57

Fonte: a autora (2022)

três classificadores, pode-se considerar que a concatenação de *embedding* resultou na melhor performance de classificação em relação à acurácia.

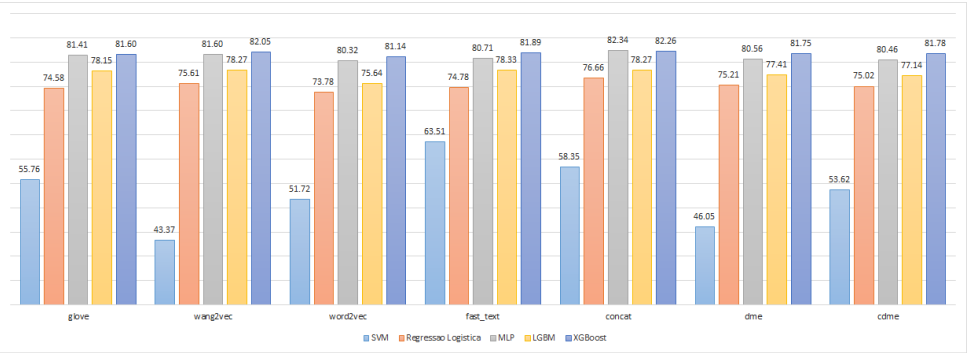
Tabela 19 – Resultados do teste de Friedman para as diferentes combinações de *embeddings* (Olist *Dataset* - dados de teste)

classificador	***p-value***	***p-value*** $< \alpha$	**resultado**
SVM	0.5488	não	*não é possível rejeitar* H_0
RL	0.0224	sim	*rejeita-se* H_0
MLP	0.0150	sim	*rejeita-se* H_0
LGBM	0.5488	não	*não é possível rejeitar* H_0
XGBoost	0.0498	sim	*rejeita-se* H_0

Fonte: a autora (2022)

Para sumarizar os resultados obtidos para cada classificador em relação a cada uma das sete formas de *embeddings* utilizadas no total (quatro *embeddings* simples e três combinações) para os dados da base Olist, foi obtido o gráfico comparativo mostrado na Figura 25.

Figura 25 – Acurácia média dos classificadores para todos os *embeddings* (Olist *Dataset* - dados de teste)



Fonte: a autora (2022).

Os valores das acurácias obtidos mostram a forte estabilidade dos algoritmos *XGBoost* e MLP em relação à mudança na técnica de *embedding* utilizada, onde não ocorre nenhuma diferença de acurácia superior à 2%. Já Para o LGBM e a Regressão é possível notar algumas

variações mais discretas, enquanto que para o SVM elas ocorrem de forma bem expressiva. Além disso, de forma geral, o algoritmo que apresentou melhor performance para todos os experimentos na base Olist foi a MLP, chegando a uma acurácia de 82,34% quando a concatenação entre os vetores de *embedding Word2Vec* e *FastText* foi utilizada.

Por fim, os resultados presentes na Tabela 20 mostram as acurácias obtidas para os diferentes classificadores quando a técnica *Bag of Words* foi utilizada nessa mesma base de dados. Esses valores indicam que essa forma mais simples de vetorização fornece melhores resultados que os *embbedings* quando classificadores clássicos como o SVM e a Regressão Logística são utilizados. Para os demais classificadores, a utilização de *embeddings* impacta positivamente no aumento da performance.

Tabela 20 – Acurácia média e desvio padrão dos classificadores para o BoW (*Olist Dataset* - dados de teste)

Classificador	Acc (%)	std
SVM	69.64	0.112
RL	79.08	0.178
MLP	81.12	0.211
LGBM	75.86	0.063
XGB	80.43	0.099

Fonte: a autora (2022)

5.1.3 Comparação entre CNN e os classificadores de AM

Com o intuito de verificar a performance de um algoritmo de aprendizado profundo aplicado ao problema de análise dos sentimentos das avaliações da base Olist sem utilizar *embeddings* pré-treinados, nesta subseção a comparação é realizada entre a CNN e o melhor resultado dos demais algoritmos de AM em relação aos diferentes *embeddings*. As acurácias e tempos de execução médios de cada modelo são apresentadas na Tabela 21, onde é possível observar que a CNN fornece a melhor acurácia em relação aos demais classificadores. Entretanto, também apresenta o maior tempo de execução quando comparado aos demais.

Para confirmar se a CNN é de fato o melhor classificador para a base Olist, realizou-se o teste de Wilcoxon entre este classificador e cada um dos demais, cujos resultados são mostrados na Tabela 22. Uma vez que para todas as comparações a hipótese nula pode ser rejeitada, pode-se confirmar que, em termos de acurácia, a CNN apresenta o melhor resultado para esta base.

Tabela 21 – Acurácia, desvio padrão e tempo médio de execução para os modelos de melhor performance (Olist *Dataset* - dados de teste)

classificador	**tempo (min)**	**acc (%)**	**std**
CNN	1126	**84.35**	0.003
FastText_SVM	39	63.51	0.053
concat_Regressao Logistica	20	76.66	0.004
concat_MLP	75	82.34	0.008
FastText_LGBM	61	78.33	0.009
concat_XGBoost	768	82.26	0.007

Fonte: a autora (2022)

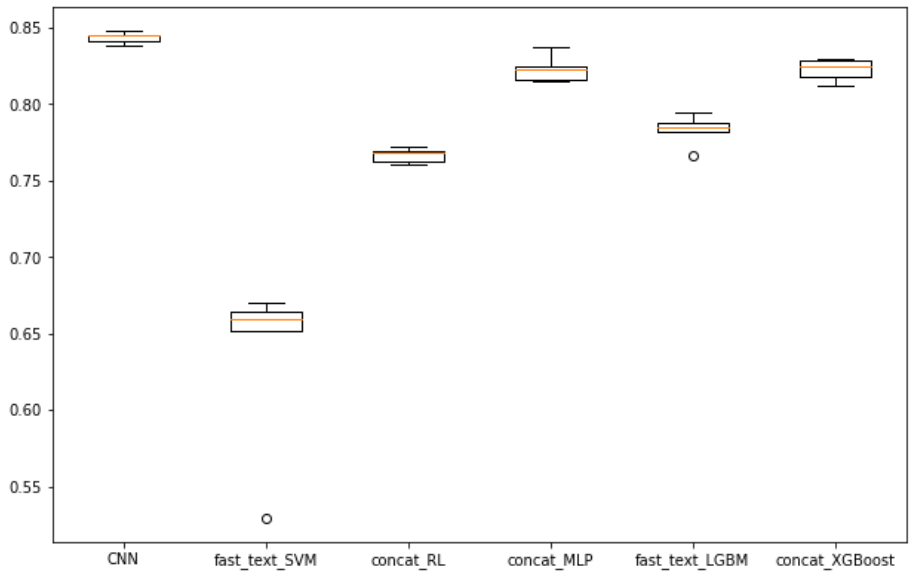
Tabela 22 – Comparação da performance entre a CNN e os demais classificadores via teste de Wilcoxon (Olist *Dataset* - dados de teste)

Classificador	**p-value**	***p-value*** $< \alpha$	**resultado**
CNN x FastText_SVM	0.006093	sim	*rejeita-se* H_0
CNN x concat_RL	0.006093	sim	*rejeita-se* H_0
CNN x concat_MLP	0.006093	sim	*rejeita-se* H_0
CNN x s FastText_LGBM	0.006093	sim	*rejeita-se* H_0
CNN x concat_XGB	0.006093	sim	*rejeita-se* H_0

Fonte: a autora (2022)

Além dos resultados dos testes estatísticos, através dos *boxplots* mostrados na Figura 26, é possível observar que a distribuição das acurácias para o CNN é a mais estável dentre as demais, concentrada em valores entre 80% e 85%. Além disso, pode-se observar que as acurácias da CNN estão distribuídas bem acima das acurácias dos demais classificadores, corroborando com a diferença estatística das performances.

Figura 26 – Comparação da distribuição das acurácias entre os melhores classificadores (Olist *Dataset* - dados de teste)



Fonte: a autora (2022).

5.2 ANÁLISE DOS RESULTADOS - BUSCAPÉ CORPUS

Esta seção exibe os resultados experimentais obtidos para as avaliações dos consumidores do Buscapé Corpus. Assim como a seção anterior e demais análises dos resultados desse capítulo, o processo experimental foi dividido em três etapas que serão analisadas separadamente.

5.2.1 Comparação entre embeddings simples

Para verificar o impacto dos diferentes tipos de *embeddings* considerados na performance de classificação de textos para diferentes algoritmos, foi observada a métrica de acurácia média, que é mostrada 23, onde é destacado o valor da melhor acurácia obtida para cada classificador e seu desvio padrão associado.

Tabela 23 – Acurácia média e desvio padrão de cada algoritmo de classificação por *Embedding* (Buscapé Corpus - dados de teste)

Embedding	**SVM**		**RL**		**MLP**		**LGBM**		**XGB**	
	acc (%)	std	acc (%)	std	acc (%)	std	acc (%)	std	acc (%)	std
Glove	52.54	0.287	65.84	0.005	80.50	0.004	65.46	0.017	80.57	0.003
Wang2Vec	48.54	0.317	66.50	0.006	80.64	0.003	**71.78**	0.034	80.68	0.002
Word2Vec	**76.95**	0.004	64.78	0.008	80.29	0.005	70.54	0.039	80.05	0.007
FastText	50.86	0.309	**67.01**	0.004	**80.76**	0.003	70.64	0.040	**80.72**	0.003

Fonte: a autora (2022)

Para as avaliações da base do Buscapé, o SVM também foi o classificador que apresentou maior impacto com a mudança na técnica de *embedding*, porém atingindo um resultado bastante superior quando a vetorização via *Word2Vec* foi realizada. Os classificadores *XGBoost* e MLP seguiram apresentando os melhores resultados e quase nenhuma variação de performance para as mudanças de *embedding*. Já a Regressão e o LGBM apresentaram os melhores resultados quando os *embeddings* utilizados foram, respectivamente, *FastText* e *Wang2Vec*. A melhores acurácias foram aproximadamente iguais para a MLP (80.76%) e o *XGBoost* (80.72%), utilizando *FastText* como forma de vetorização em ambos os casos.

Na Tabela 24, são apresentados os resultados dos testes de Friedman realizados para verificar se a mudança na técnica de *embedding* foi estatisticamente significante para os classificadores. Nesse caso a hipótese nula, de que as distribuições são iguais para os diferentes tipos de *embeddings*, só pode ser rejeitada para o classificadores Regressão Logística, XGBoost e MLP, mesmo com os valores de acurácia próximos no caso dos dois últimos classificadores. Já para os classificadores SVM e LGBM não é possível concluir que a mudança do tipo de *embbeding* de fato impacta na distribuição das acurácias desses modelos.

Tabela 24 – Resultados do teste de Friedman para os diferentes *embeddings* (Buscapé Corpus - dados de teste)

classificador	***p-value***	***p-value*** $< \alpha$	**resultado**
SVM	0.1447	*não*	*não é possível rejeitar* H_0
RL	0.0029	sim	*rejeita-se* H_0
MLP	0.0211	sim	*rejeita-se* H_0
LGBM	0.0624	não	*não é possível rejeitar* H_0
XGB	0.0136	sim	*rejeita-se* H_0

Fonte: a autora (2022)

5.2.2 Comparação entre combinações de embedding

A partir dos resultados obtidos para cada *embedding* individual, realizou-se o teste de Wilcoxon para cada possível dupla de *embeddings*, com o intuito de escolher o par de técnicas de vetorização que fossem distintas para a maioria dos classificadores. Os resultados dos p-valores obtidos para cada teste estão consolidados na Tabela 25. Assim como ocorreu para o Olist *Dataset*, para os dados do Buscapé as técnicas de *embedding* que forneceram distribuições distintas para a maioria dos classificadores (três dos cinco utilizados) foi o par de *embeddings*

Word2Vec e *FastText*, cujos p-valores, conforme mostrados em destaque na Tabela 25 são menores que 0.05, sendo possível então descartar a hipótese de que as distribuições são iguais.

Tabela 25 – p-valores obtidos nos testes de Wilcoxon realizados para cada par de *embedding* (Buscapé Corpus - dados de teste)

Combinação	Classificador				
	SVM	RL	MLP	LGBM	XGBoost
Glove x Wang2Vec	0.3381	0.0473	0.2017	0.0061	0.3376
Glove x Word2Vec	0.0718	0.0301	0.2654	0.0473	0.1050
Glove x FastText	0.3381	0.0061	0.2017	0.0184	0.2017
Wang2Vec x FastText	0.3381	0.1481	0.2654	0.3381	0.4170
Word2Vec x FastText	0.0473	0.0061	0.0718	0.5000	0.0473

Fonte: a autora (2022)

Com o par de *embeddings* definido, foram realizadas as classificações de cada algoritmo para os três tipos de combinações de *embedding* estudadas. Os resultados, exibidos na Tabela 26, também mostram a combinação CONCAT como vencedora em relação à acurácia para todos os classificadores, com ganhos expressivos para o SVM e o LGBM quando comparados aos meta-*embeddings* dinâmicos.

Tabela 26 – Acurácia e tempo de processamento médio para cada modelo (Buscapé Corpus - dados de teste)

Embedding	**SVM**		**RL**		**MLP**		**LGBM**		**XGB**	
	Acc (%)	Tempo (min)	Acc (%)	Tempo (min)	Acc (%)	Tempo (min)	Acc (%)	Tempo (min)	Acc (%)	Tempo (min)
concat	**65.07**	14.06	**67.54**	47.20	**80.99**	413.41	**72.75**	51.00	**81.05**	1316.19
dme	38.04	8.60	66.67	62.12	80.62	460.45	64.92	31.80	80.56	954.72
cdme	26.23	10.64	66.39	49.37	80.33	658.81	67.52	27.76	80.63	3072.39

Fonte: a autora (2022)

Assim como ocorreu para os *embeddings* simples, as performances das combinações de *embeddings* também foram comparadas através do teste de Friedman. Os resultados são mostrados na Tabela 27 e indicam que as três formas de combinação de *embeddings* só podem ser distintas para o caso da Regressão, do LGBM e do XGBoost. Uma vez que a utilização da concatenação de *embedding* trouxe o melhor resultados para esses classificadores, pode-se considerá-la como a melhor forma de combinação em relação à acurácia, para os dados do Buscapé.

Para sumarizar os resultados obtidos para cada classificador em relação a cada uma das sete formas de *embedding* utilizadas para os dados do Buscapé Corpus, foi obtido o gráfico

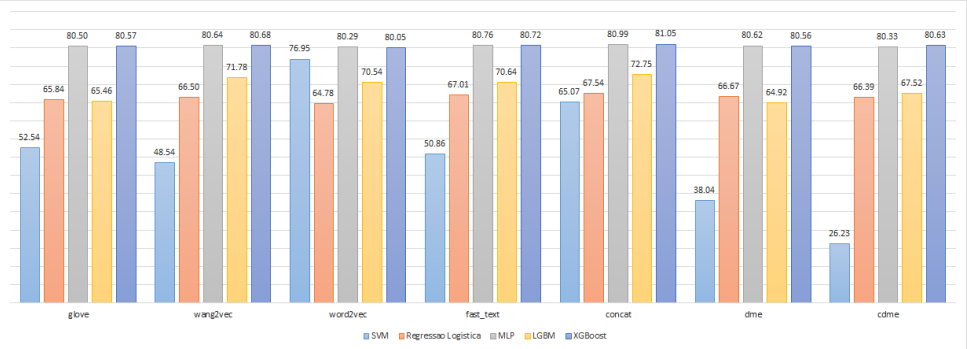
Tabela 27 – Resultados do teste de Friedman para as diferentes combinações de *embeddings* (Buscapé Corpus - dados de teste)

Classificador	**p-value**	***p-value*** $< \alpha$	**resultado**
SVM	0.2466	não	*não é possível rejeitar* H_0
RL	0.0150	sim	rejeita-se H_0
MLP	0.5488	não	*não é possível rejeitar* H_0
LGBM	0.0150	sim	*rejeita-se* H_0
XGBoost	0.0224	sim	*rejeita-se* H_0

Fonte: a autora (2022)

comparativo mostrado na Figura 27.

Figura 27 – Acurácia média dos classificadores para todos os *embeddings* (Buscapé Corpus - dados de teste)



Fonte: a autora (2022).

Os valores das acurácias obtidos mostram que os classificadores mais impactados pela mudança de *embedding* no geral foram SVM, LGBM e Regressão Logística. Par o SVM o melhor resultado foi obtido através da combinação de *embeddings* via concatenação (acurácia média de 65.07%), enquanto o pior ocorreu para a combinação dinâmica CDME (acurácia média de 26.23%). Já para o LGBM, a diferença foi um pouco menos significativa, e o melhor resultado foi obtido através de um *embedding* simples, *Word2Vec*, fornecendo quase 77% de acurácia. O pior desempenho para o LGBM ocorreu quando o *embedding Glove* foi utilizado, chegando a 65.46% de acurácia. Para a Regressão o pior resultado (64.78% de acurácia) ocorreu quando a técnica *Word2Vec* foi utilizada e o melhor com a combinação CONCAT, atingindo quase 73% de acurácia. Novamente, os classificadores MLP e *XGBoost* ficaram em empate técnico para todos os tipos de *embedding* aplicados, com ou sem combinação, além de apresentarem os melhores resultados de acurácia média: aproximadamente 81% em ambos os casos.

Utilizando-se a técnica *Bag of Words* como forma de vetorização, foram obtidas as acu-

rácias mostradas Tabela 28. É possível notar que os resultados são parecidos com os obtidos quando as técnicas de *embedding* foram utilizadas no caso dos classificadores SVM e a Regressão Logística. Porém em relação aos algoritmos baseados em árvore, como o LGBM e o XGBoost, novamente a utilização de *embeddings* trouxe melhoria na performance em relação à esta métrica de comparação utilizada.

Tabela 28 – Acurácia média e desvio padrão dos classificadores para o BoW (*Buscape Corpus* - dados de teste)

Classificador	Acc (%)	std
SVM	68.76	0.099
RL	71.18	0.128
MLP	78.12	0.225
LGBM	70.66	0.033
XGB	79.67	0.102

Fonte: a autora (2022)

5.2.3 Comparação entre CNN e os classificadores de AM

Nesta subseção a comparação é realizada entre a CNN e o melhor resultado dos demais algoritmos de AM em relação aos diferentes *embeddings* para a base Buscapé. As acurácias, desvio-padrão e tempo de execução médio de cada modelo são apresentadas na Tabela 29. Para esta base de dados, houve quase um empate entre as acurácias obtidas pela CNN sem utilizar *embedding* pré-treinado e para MLP e *XGBoost* quando a combinação CONCAT foi utilizada. Porém, a CNN apresenta tempo de execução quase 17 vezes maior que o concat_MLP, o que acaba se tornando um critério de desempate significativo.

Para verificar se há diferença estatística entre a CNN e os demais classificadores, em especial o concat_ML e o concat_*XGBoost*, realizou-se o teste de Wilcoxon entre este classificador e cada um dos demais, cujos resultados são mostrados na Tabela 30. Nota-se que de fato a CNN e o concat_MLP não podem ser considerados distintos, uma vez que o p-valor dessa estatística é maior que 0.05. O mesmo ocorre entre a CNN e o *XGboost*. Uma vez que o classificador MLP utilizado com concatenação entre *embeddings* apresentou o mesmo valor de acurácia que a CNN mas tempo de processamento significativamente menor, o empate técnico acaba ficando entre a MLP e o *XGBoost*, considerando que a média das acurácias de ambos é aproximadamente igual, como o tempo de processamento do concat_MLP é cerca de

Tabela 29 – Acurácia, desvio padrão e tempo médio de execução para os modelos de melhor performance (Buscapé Corpus - dados de teste)

classificador	**tempo (min)**	**acc (%)**	**std**
CNN	6901	80.99	0.021
word2vec_SVM	8	76.95	0.004
concat_Regressao Logistica	47	67.54	0.005
concat_MLP	413	80.99	0.003
concat_LGBM	51	72.75	0.029
concat_XGBoost	1316	**81.05**	0.003

Fonte: a autora (2022)

3 vezes menor que o do concat_XGBoost, pode-se estabelecer o concat_MLP como melhor modelo para os dados do Buscapé Corpus.

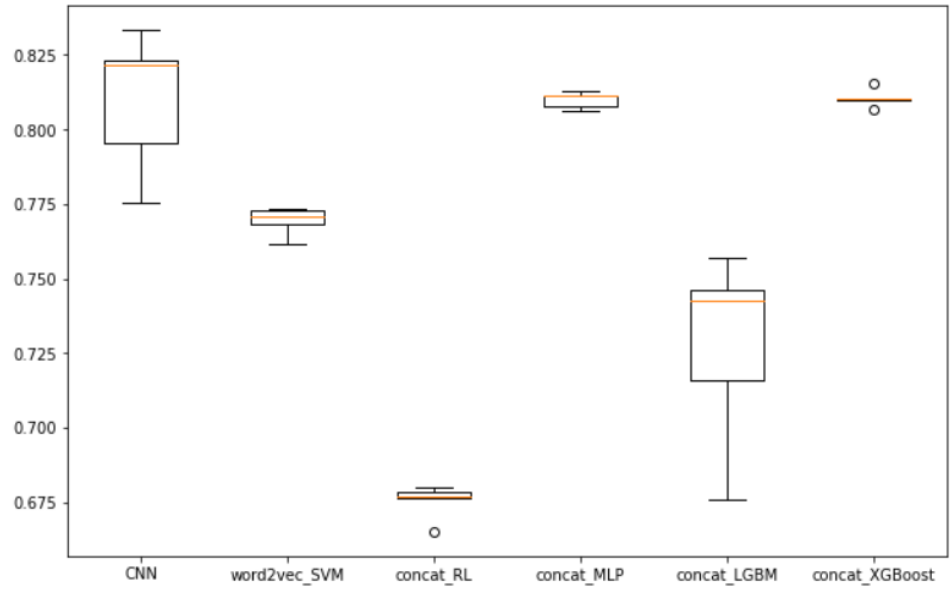
Tabela 30 – Comparação da performance entre a CNN e os demais classificadores via teste de Wilcoxon (Buscapé Corpus - dados de teste)

Classificador	**p-value**	***p-value*** $< \alpha$	**resultado**
CNN x Word2Vec_SVM	0.006093	sim	*rejeita-se* H_0
CNN x concat_Regressao Logistica	0.006093	sim	*rejeita-se* H_0
CNN x concat_MLP	0.338052	não	*não é possível rejeitar* H_0
CNN x concat_LGBM	0.006093	sim	*rejeita-se* H_0
CNN x concat_XGBoost	0.338052	não	*não é possível rejeitar* H_0

Fonte: a autora (2022)

Além dos resultados dos testes estatísticos, através dos *boxplots* mostrados na Figura 28, é possível observar que a distribuição das acurácias para MLP e *XGBoost*, ambos utilizando concatenação de *embeddings*, são as mais estáveis dentre as demais, concentradas em valores muito próximos de 81%. Apesar de estar no mesmo patamar em relação à acurácia, a CNN apresenta uma distribuição com mais variância entre as amostras quando comparada aos outros dois classificadores.

Figura 28 – Comparação da distribuição das acurácias entre os melhores classificadores (Buscape Corpus - dados de teste)



Fonte: a autora (2022).

Em relação aos demais classificadores, a Regressão utilizando a combinação CONCAT apresentou a pior performance em relação à acurácia enquanto o LGBM apresentou a maior vacância entre os valores da distribuição.

5.3 ANÁLISE DOS RESULTADOS - B2W *REVIEWS*

Esta seção exibe os resultados experimentais obtidos para as avaliações dos consumidores presentes no B2W *Reviews*. Assim como a seção anterior e demais análises dos resultados desse capítulo, o processo experimental foi dividido em três etapas que serão analisadas separadamente.

5.3.1 Comparação entre embeddings simples

Para verificar o impacto dos diferentes tipos de *embeddings* considerados diante de diferentes algoritmos de classificação, inicialmente foi observada a métrica de acurácia média, que é mostrada na Tabela 31, onde é destacado o valor da melhor acurácia obtida para cada classificador em cada *embedding*.

Para as avaliações da base do B2W, esses resultados mostram as técnicas de *embedding* *Wang2Vec* e *FastText* como as que trouxeram os melhores ganhos na performance dos clas-

Tabela 31 – Acurácia média e desvio padrão de cada algoritmo de classificação por *embedding* (B2W *Reviews* - dados de teste)

Embedding	**SVM**		**RL**		**MLP**		**LGBM**		**XGB**	
	acc (%)	std	acc (%)	std	acc (%)	std	acc (%)	std	acc (%)	std
Glove	52.12	0.131	69.64	0.002	77.48	0.003	71.94	0.014	78.78	0.002
Wang2Vec	46.22	0.181	**71.09**	0.003	78.56	0.004	71.58	0.012	**79.00**	0.004
Word2Vec	**56.34**	0.090	68.85	0.007	73.31	0.063	71.95	0.018	74.26	0.069
FastText	50.65	0.187	70.58	0.002	**78.84**	0.007	**72.48**	0.020	**79.00**	0.006

Fonte: a autora (2022)

sificadores, com exceção do SVM, onde o melhor resultado (56%) foi obtido utilizando-se o *Word2Vec*. Ainda assim, o SVM segue sendo o algoritmo com pior desempenho em todos os casos quando comparado aos demais. A técnica *Wang2Vec* trouxe a melhor acurácia para a Regressão Logística e um empate com o *FastText* quando aplicados ao *XGBoost*, ambos fornecendo o melhor resultado de acurácia dentre todos os experimentos, igual a 79%. O *FastText* foi a técnica que forneceu a melhor acurácia para a maioria dos classificadores (3 dos 5), fornecendo a segunda maior acurácia geral de 78.84% (aproximadamente 79%) para a MLP.

Considerando a comparação entre as distribuições desses modelos, a Tabela 32 apresenta os resultados dos testes de Friedman realizados para verificar se a mudança na técnica de *embedding* foi estatisticamente significante para os classificadores. Nesse caso a hipótese nula só pôde ser rejeitada para o classificadores Regressão Logística e MLP. Já para os demais classificadores não é possível concluir que a mudança no tipo de *embbeding* tenha impactado na distribuição das acurácias dos resultados.

Tabela 32 – Resultados do teste de Friedman para os diferentes *embeddings* (B2W *Reviews* - dados de teste)

classificador	***p-value***	***p-value*** $< \alpha$	**resultado**
SVM	0.9484	não	não é possível rejeitar H_0
RL	0.0029	sim	rejeita-se H_0
MLP	0.0056	sim	rejeita-se H_0
LGBM	0.7819	não	não é possível rejeitar H_0
XGBoost	0.0694	não	não é possível rejeitar H_0

Fonte: a autora (2022)

5.3.2 Comparação entre combinações de embedding

A partir dos resultados obtidos para cada *embedding* individual, realizou-se o teste de Wilcoxon para cada possível dupla de *embedding*, para a escolha do par de técnicas de vetorização a serem combinados. Os resultados dos p-valores obtidos para cada teste são mostrados na Tabela 33, e novamente as técnicas de *embedding* que forneceram distribuições distintas para a maioria dos classificadores (três dos cinco utilizados) foram *Word2Vec* e *FastText*, cujos p-valores, conforme mostrados em destaque na Tabela 33 são menores que 0.05.

Tabela 33 – p-valores obtidos nos testes de Wilcoxon realizados para cada par de *embedding* (B2W *Reviews* - dados de teste)

Combinação	Classificador				
	SVM	RL	MLP	LGBM	XGBoost
Glove x Wang2Vec	0.5000	0.0061	0.0108	0.3381	0.2017
Glove x Word2Vec	0.2017	0.0718	0.0718	0.4173	0.0108
Glove x FastText	0.4173	0.0061	0.0061	0.4173	0.0718
Wang2Vec x FastText	0.4173	0.0301	0.4173	0.3381	0.4173
Word2Vec x FastText	0.2017	0.0061	0.0061	0.4173	0.0301

Fonte: a autora (2022)

Os resultados de acurácia e tempo médio de execução obtidos para os experimentos com os *embeddings* combinados são mostrados na Tabela 34. Novamente, a combinação CONCAT foi a que trouxe melhores resultados para a maioria dos classificadores, porém o *meta-embedding* dinâmico desta vez trouxe ganho de performance para o SVM e o LGBM, fornecendo os melhores resultados de acurácia para esses classificadores e com menor tempo de execução quando comparados às demais combinações.

Tabela 34 – Acurácia e tempo de processamento médio para cada modelo (B2W *Reviews* - dados de teste)

Embedding	**SVM**		**RL**		**MLP**		**LGBM**		**XGB**	
	Acc (%)	Tempo (min)	Acc (%)	Tempo (min)	Acc (%)	Tempo (min)	Acc (%)	Tempo (min)	Acc (%)	Tempo (min)
concat	59.63	62.03	**71.73**	67.86	**79.47**	161.17	71.68	102.74	**79.47**	3510.23
dme	**61.10**	22.54	70.31	38.97	74.79	119.70	**72.45**	52.31	78.30	725.58
cdme	51.21	20.27	70.40	41.52	77.90	74.74	71.92	119.93	77.76	298.59

Fonte: a autora (2022)

Novamente o melhor resultado obtido com as combinações de *embedding* foi um empate entre a MLP e o *XGBoost*, ambos utilizando a combinação CONCAT, com uma acurácia

média aproximadamente igual a 79,6%, porém com tempo de execução muito menor para a MLP. O teste de Friedman foi então aplicado para avaliar a diferença estatística entre essas combinações para cada classificador e os resultados, mostrados na Tabela 35, revelam que a diferença de impacto só ocorreu para os classificadores cujo melhor resultado foi a concatenação de *embeddings*, a saber: Regressão, MLP e *XGBoost*, para os quais é possível rejeitar a hipótese nula de igualdade entre as distribuições para os diferentes *embeddings*.

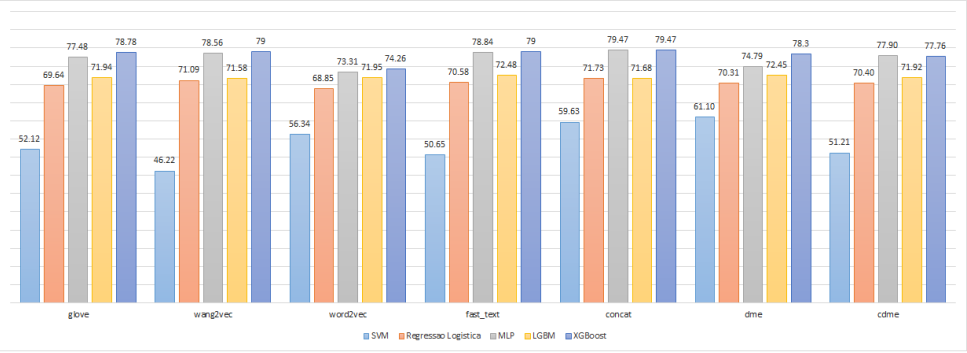
Tabela 35 – Resultados do teste de Friedman para os diferentes meta-*embeddings* (B2W *Reviews* - dados de teste)

classificador	**p-value**	***p-value*** $< \alpha$	**resultado**
SVM	0.0743	não	não é possível rejeitar
Regressao Logistica	0.0150	sim	rejeita-se
MLP	0.0224	sim	rejeita-se
LGBM	0.8187	não	não é possível rejeitar
XGBoost	0.0150	sim	rejeita-se

Fonte: a autora (2022)

A Figura 29 mostra graficamente os resultados das acurácias médias obtidas para cada classificador em a todas as sete formas de *embedding* avaliadas para o B2W *Reviews*. Para este conjunto de dados o patamar da máxima acurácia ficou em torno de 79,5% e está empatado para os algoritmos MLP e *XGBoost*, ambos utilizando a combinação CONCAT como forma de *embedding*. O SVM segue sendo o classificador com pior performance, porém com menos variações nas acurácias para esses dados em relação às outras bases estudadas. Enquanto isso, a Regressão e o LGBM continuam apresentando resultados de performance intermediários em relação aos demais classificadores.

Figura 29 – Acurácia média dos classificadores para todos os *embeddings* (B2W *Reviews* - dados de teste)



Fonte: a autora (2022).

Utilizando-se a técnica *Bag of Words* como forma de vetorização, foram obtidas as acu-

rácias mostradas na Tabela 44. Novamente, verifica-se o impacto positivo desta técnica de vetorização mais simples em relação aos classificadores SVM e a Regressão Logística, uma vez que para estes classificadores foram obtidos melhores resultados que os obtidos a partir das diferentes técnicas de *embedding* utilizadas.

Tabela 36 – Acurácia média e desvio padrão dos classificadores para o BoW (*B2W Reviews* - dados de teste)

Classificador	Acc (%)	std
SVM	67.55	0.112
RL	70.67	0.141
MLP	78.12	0.225
LGBM	70.81	0.088
XGB	75.56	0.021

Fonte: a autora (2022)

5.3.3 Comparação entre CNN e os classificadores de AM

A última etapa experimental consiste na comparação entre o classificador CNN e o melhor resultado de cada um dos demais algoritmos de AM em relação aos diferentes *embeddings*. As acurácias, desvio padrão e tempo de execução médio de cada modelo para os dados da B2W são mostrados na Tabela 37. É possível observar que o melhor desempenho em relação à acurácia neste caso foi obtido pela CNN, com o concat_MLP e concat_*XGBoost* ficando em empate no segundo lugar, porém para esta base o maior tempo de execução foi obtido pelo *XGBoost*, podendo-se então descartá-lo como melhor classificador neste caso.

Tabela 37 – Acurácia, desvio padrão e tempo médio de execução para os modelos de melhor performance (B2W *Reviews* - dados de teste)

classificador	**tempo (min)**	**acc (%)**	**std**
CNN	9383	**80.77**	0.019
dme_SVM	23	61.10	0.003
concat_Regressao Logistica	68	71.73	0.004
concat_MLP	161	79.47	0.004
FastText_LGBM	26	72.48	0.020
concat_XGBoost	3510	79.47	0.002

Fonte: a autora (2022)

A escolha do melhor classificador para o B2W *Reviews* pode ser definida a partir dos resultados dos testes de Wilcoxon realizados entre a CNN e cada um dos melhores modelos de AM obtidos nas etapas anteriores. Os resultados, mostrados na Tabela 38, indicam que não é possível rejeitar a hipótese de que as distribuições são iguais para as comparações entre a CNN e concat_MLP e entre CNN e concat_*XGBoost*, ou seja, pode-se considerar um empate entre esses dois pares de classificadores. Considerando o critério de tempo de execução como desempate, o concat_MLP apresenta os melhores resultados para esta base de dados.

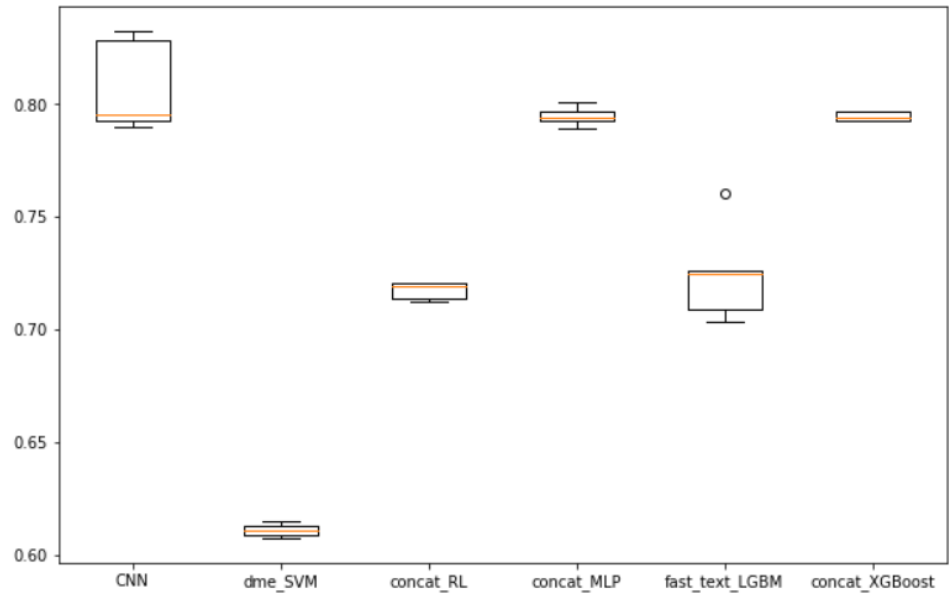
Tabela 38 – Comparação da performance entre a CNN e os demais classificadores via teste de Wilcoxon (B2W *Reviews* - dados de teste)

Classificador	**p-value**	**p-value <alfa**	**resultado**
CNN x dme_SVM	0.006093	sim	*rejeita-se* H_0
CNN x concat_Regressao Logistica	0.006093	sim	*rejeita-se* H_0
CNN x concat_MLP	0.265435	não	*não é possível rejeitar* H_0
CNN x FastText_LGBM	0.006093	sim	*rejeita-se* H_0
CNN x concat_XGBoost	0.500000	não	*não é possível rejeitar* H_0

Fonte: a autora (2022)

Os resultados da Tabela 38 também indicam a superioridade da CNN em relação aos classificadores dme_SVM, concat_Regressao Logística e *FastText*_LGBM, o que é comprovado através dos *boxplots* exibidos na Figura 30, uma vez que as distribuições desses três classificadores estão bem abaixo da dos classificadores CNN, concat_MLP e concat_*XGBoost*. Outro ponto interessante é a variação da distribuição das acurácias no caso da CNN, mostrando que os modelos de AM apresentaram resultados mais estáveis para os dados da B2W do que aqueles obtidos pela rede de aprendizado profundo.

Figura 30 – Comparação da distribuição das acurácias entre os melhores classificadores (B2W *Reviews* - dados de teste)



Fonte: a autora (2022).

5.4 ANÁLISE DOS RESULTADOS - UTLC *MOVIES*

Esta seção exibe os resultados experimentais obtidos para a base de dados contendo as avaliações dos consumidores do UTLC *Movies*, para a qual o processo experimental foi também foi dividido em três etapas que serão analisadas separadamente nas subseções a seguir.

5.4.1 Comparação entre embeddings simples

A primeira etapa de análise é voltada para o estudo do impacto dos diferentes tipos de *embeddings* considerados, diante de diferentes algoritmos de classificação. Para isso, inicialmente foi observada a métrica de acurácia média, que é mostrada na Tabela 39, onde é destacado o valor da melhor acurácia obtida para cada classificador em cada *embedding* considerado.

Para o UTLC *Movies*, o classificador SVM apresentou os piores desempenhos, independente do tipo de *embedding* considerado, com a máxima acurácia média inferior a 36%. Os resultados obtidos para a Regressão Logística e o LGBM também não foram muito satisfatórios e a acurácia mais próxima de 60% foi obtida pelo LGBM quando o *embedding Glove* foi utilizado. Em relação aos melhores resultados, a tendência de empate entre resultados da MLP e *XGBoost* também continuou para esta base, com o *XGBoost* apresentando a acurácia máxima dentre todos os experimentos com *embeddings* simples, correspondente a 71.42%.

Tabela 39 – Acurácia média e desvio padrão de cada algoritmo de classificação por *embedding* (UTLC *Movies* - dados de teste)

Embedding	**SVM**		**RL**		**MLP**		**LGBM**		**XGB**	
	acc (%)	std	acc (%)	std	acc (%)	std	acc (%)	std	acc (%)	std
Glove	34.26	0.298	54.52	0.006	71.00	0.003	**58.61**	0.028	71.24	0.003
Wang2Vec	**35.48**	0.288	54.55	0.001	70.97	0.003	55.52	0.022	71.31	0.002
Word2Vec	21.07	0.236	53.91	0.007	70.92	0.002	53.99	0.029	71.17	0.004
FastText	33.72	0.301	**55.40**	0.005	**71.04**	0.001	55.40	0.021	**71.42**	0.002

Fonte: a autora (2022)

Do ponto de vista estatístico, a Tabela 40 apresenta os resultados dos testes de Friedman realizados para verificar se a mudança na técnica de *embedding* foi estatisticamente significante para cada um dos classificadores. Nesse caso a hipótese nula, de que as distribuições são iguais para os diferentes tipos de *embeddings*, só pode ser rejeitada no caso da Regressão Logística. Para os demais classificadores não é possível concluir que a mudança do tipo de *embbeding* tenha impactado na distribuição das acurácias dos resultados.

Tabela 40 – Teste de Friedman para cada classificador em relação aos *embeddings* (UTLC *Movies* - dados de teste)

classificador	***p-value***	***p-value*** $< \alpha$	**resultado**
SVM	0.5028	não	*não é possível rejeitar* H_0
Regressao Logistica	0.0406	sim	*rejeita-se* H_0
MLP	0.8504	não	não é possível rejeitar H_0
LGBM	0.2407	não	*não é possível rejeitar* H_0
XGBoost	0.3234	não	*não é possível rejeitar* H_0

Fonte: a autora (2022)

5.4.2 Comparação entre combinações de embedding

A etapa de combinação de *embeddings* para análise do sentimento das avaliações do UTLC *Movies* também foi definida a partir dos resultados obtidos pelo teste de Wilcoxon para cada possível combinação dois a dois dos *embeddings* apresentados anteriormente. Os resultados dos p-valores obtidos para cada teste são mostrados na Tabela 41, desta vez nenhuma das combinações apresentou diferença estatística para mais de um classificador. Diante disto, o critério de desempate considerado para a decisão foi o dos classificadores que obtiveram p-valores mais próximos possível de 0.05. Assim, observando os valores da Tabela 41, a dupla de *embeddings* que atende este critério é (*Glove*, *FastText*), uma vez que apresentam os menores

p-valores para mais de um classificador (p-valor < 0.05 para a Regressão e p-valor $= 0.0718$ para o LGBM).

Tabela 41 – Escolha dos *embeddings* para combinação baseados no teste de Wilcoxon (UTLC *Movies* - dados de teste)

Combinação	Classificador				
	SVM	RL	MLP	LGBM	XGBoost
Glove x Wang2Vec	0.5000	0.5000	0.5000	0.1050	0.5000
Glove x Word2Vec	0.1050	0.1481	0.4583	0.0301	0.5000
Glove x FastText	0.2642	0.0301	0.2017	0.0718	0.1481
Wang2Vec x FastText	0.3362	0.0301	0.2017	0.5000	0.2017
Word2Vec x FastText	0.4583	0.0108	0.3381	0.2017	0.1481

Fonte: a autora (2022)

Tabela 42 – Acurácia e tempo de processamento médio para cada modelo (UTLC *Movies* - dados de teste)

Embedding	**SVM**		**RL**		**MLP**		**LGBM**		**XGB**	
	Acc (%)	Tempo (min)	Acc (%)	Tempo (min)	Acc (%)	Tempo (min)	Acc (%)	Tempo (min)	Acc (%)	Tempo (min)
concat	33.29	34.27	**55.78**	70.00	**71.62**	58.50	**59.57**	162.21	**71.45**	480.68
dme	**56.97**	18.05	54.49	104.14	70.85	328.24	54.07	1117.59	71.32	413.36
cdme	32.78	13.70	54.28	116.89	71.01	529.06	54.84	78.80	**71.46**	460.15

Fonte: a autora (2022)

A partir da escolha dos vetores de *embeddings* a serem combinados, foram realizados os experimentos de classificação para cada uma das três combinações em estudo. Os resultados de acurácia e tempo médio de execução obtidos se encontram na Tabela 42. Com exeção da SVM, para qual o melhor desempenho foi obtido com a combinação DME, a combinação CONCAT foi a que trouxe melhores resultados para os demais classificadores. Para o *XGBoost*, tanto o CONCAT com o CDME trouxeram praticamente os mesmos resultados de acurácia média e pouca diferença no tempo de processamento (o modelo com o CDME foi mais rápido cerca de 20 minutos em relação ao que utilizou CONCAT).

A partir dos resultados acima, o teste de Friedman foi aplicado para avaliar a diferença estatística entre essas combinações de *embeddings* para cada classificador. Os resultados, mostrados na Tabela 43, indicam que a diferença de distribuição só ocorre para a Regressão Logística, para os demais classificadores não é possível afirmar que houve diferenças estatisticamente relevantes entre as combinações de *embeddings*.

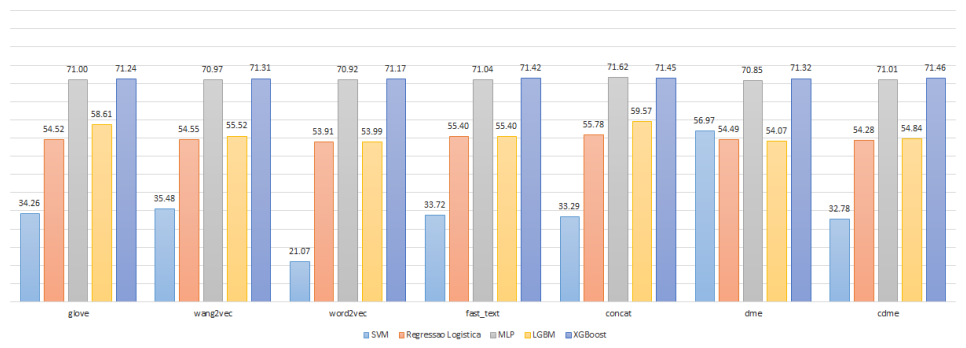
Tabela 43 – Resultados do teste de Friedman para as diferentes combinações de *embeddings* (UTLC *Movies* - dados de teste)

classificador	**p-value**	***p-value*** $< \alpha$	**resultado**
SVM	0.2466	não	*não é possível rejeitar* H_0
Regressao Logistica	0.0067	sim	*rejeita-se* H_0
MLP	0.1040	não	não é possível *rejeitar* H_0
LGBM	0.0743	não	*não é possível rejeitar* H_0
XGBoost	0.8187	não	*não é possível rejeitar* H_0

Fonte: a autora (2022)

A Figura 31 apresenta graficamente os resultados das acurácias médias obtidas para cada classificador em todas as sete formas de *embedding* avaliadas. Para esta base de dados, a máxima acurácia é é aproximadamente de 71% e está empatada entre os classificadores MLP e *XGBoost* quando utilizam a combinação CONCAT como forma de *embedding*. O SVM segue sendo o classificador com pior performance, seguido pela Regressão Logística e o LGBM, que não conseguem chegar a uma acurácia superior a 60% para nenhum dos *embeddings* utilizados.

Figura 31 – Acurácia média dos classificadores para todos os *embeddings* (UTLC *Movies* - dados de teste)



Fonte: a autora (2022).

Utilizando-se a técnica *Bag of Words* como forma de vetorização, foram obtidas as acurácias mostradas Tabela 44. Assim como ocorreu para as demais bases de dados, o impacto positivo desta técnica de vetorização mais simples ocorreu quando os classificadores classificadores SVM e a Regressão Logística, foram utilizados. Já para a MLP, LGBM e XGBoost a utilização das técnicas de *embedding* pode ser considerada uma boa alternativa para o incremento da acurácia em relação a este conjunto de dados.

Tabela 44 – Acurácia média e desvio padrão dos classificadores para o BoW (UTLC *Movies* - dados de teste)

Classificador	Acc (%)	std
SVM	61.15	0.155
RL	64.32	0.211
MLP	69.12	0.125
LGBM	57.01	0.184
XGB	68.33	0.041

Fonte: a autora (2022)

5.4.3 Comparação entre CNN e os classificadores de AM

Para avaliar o impacto de utilização de uma rede de aprendizado profundo na análise de sentimentos dos dados UTLC, foi realizada a comparação entre o classificador CNN e o melhor resultado de cada um dos demais algoritmos de AM em relação aos diferentes *embeddings*, escolhido a partir dos resultados mostrados na Figura 31 da subseção anterior.

As acurácias e tempos de execução médio obtidos para modelo são mostrados na Tabela 45. Mais uma vez a CNN apresentou o melhor resultado em relação à acurácia média dos classificadores. Os classificadores de AM concat_MLP e cdme_XGBoost ficaram mais uma vez empatados em segundo lugar, com a concat_MLP apresentando vantagem em relação ao tempo de execução do modelo.

Tabela 45 – Acurácia, desvio padrão e tempo médio de execução para os modelos de melhor performance (UTLC *Movies* - dados de teste)

classificador	**tempo (min)**	**acc (%)**	**std**
CNN	6227	**72.29**	0.014
dme_SVM	18	56.97	0.240
concat_Regressao Logistica	70	55.78	0.003
concat_MLP	59	71.62	0.007
concat_LGBM	162	59.57	0.025
cdme_XGBoost	460	71.46	0.002

Fonte: a autora (2022)

A tentativa de desempate entre os três melhores classificadores foi definida a partir dos resultados dos testes de Wilcoxon realizados entre a CNN e cada um dos modelos exibidos na Tabela 45. Assim como ocorreu para os dados do B2W *Reviews*, os resultados mostrados na Tabela 46 indicam que não é possível rejeitar a hipótese de que as distribuições são iguais para

as comparações entre a CNN e concat_MLP e entre CNN e cdme_*XGBoost*, ou seja, pode-se considerar um empate entre esses dois pares de classificadores. Novamente considerando o tempo de execução como forma de desempate, o concat_MLP pode ser escolhido como melhor modelo de classificação para este conjunto de dados.

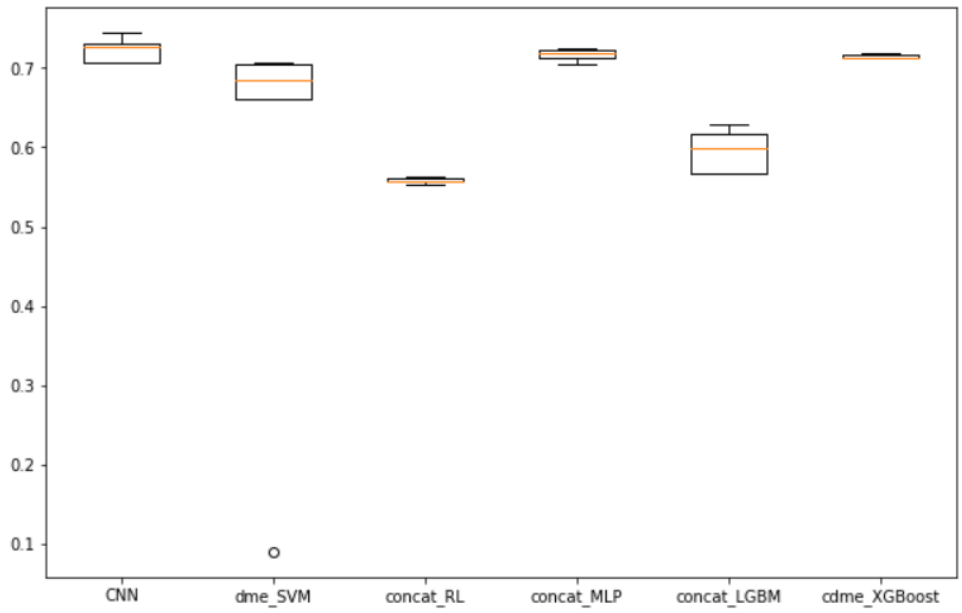
Tabela 46 – Comparação da performance entre a CNN e os demais classificadores via teste de Wilcoxon (UTLC *Movies* - dados de teste)

Classificador	**p-value**	***p-value*** $< \alpha$	**resultado**
CNN x dme_SVM	0.005963	sim	rejeita-se H_0
CNN x concat_Regressao Logistica	0.005963	sim	rejeita-se H_0
CNN x concat_MLP	0.200983	não	não é possível rejeitar H_0
CNN x concat_LGBM	0.005963	sim	rejeita-se H_0
CNN x cdme_XGBoost	0.337587	não	não é possível rejeitar H_0

Fonte: a autora (2022)

A Figura 32 exibe os *boxplots* das distribuições dos classificadores analisados, e através dela é possível confirmar as indicações resultantes do teste de Wilcoxon: é fácil notar que as distribuições dos modelos dme_SVM, concat_RL e concat_LGBM destoam das distribuições de acurácia da CNN, que por sua vez encontra-se no mesmo patamar que os modelos, concat_MLP e cdme_*XGBoost*, diferindo apenas em relação valor máximo e os valores do primeiro quartil.

Figura 32 – Comparação da distribuição das acurácias entre os melhores classificadores (ULTC *Movies* - dados de teste)



Fonte: a autora (2022).

5.5 CONSIDERAÇÕES DO CAPÍTULO

No presente Capítulo foram detalhados os resultados obtidos para cada experimento proposto nessa pesquisa, bem como as discussões acerca desses resultados para cada uma das bases de dados analisadas. Além disso, os resultados aqui obtidos também serão utilizados como comparativo para alguns experimentos de abordagens semelhantes presentes na literatura, que foram citados anteriormente no Capítulo 3.

No trabalho de (SOUZA et al., 2021), que utilizou as mesmas bases de dados usadas nesta pesquisa, os resultados publicados considerando o LGBM como classificador, na comparação entre os *embeddings* *Glove*, *FastText* e *Word2Vec* com dimensionalidades iguais à 100, o *FastText* foi a forma de vetorização que apresentou melhor desempenho para todas as bases de dados, enquanto que no presente trabalho o *FastText* obteve melhor desempenho apenas para os dados do B2W *Reviews* e do Olist *Dataset* quando o LGBM foi utilizado, conforme mostrado na Tabela 47. Entretanto, a métrica de análise utilizada em (SOUZA et al., 2021) foi a área sob a curva ROC (*Receiver Operating Characteristic Curve*, em Inglês), além disso, em (SOUZA et al., 2021) as 5 (cinco) classes das avaliações foram redistribuídas para um problema binário, diferente desta pesquisa, onde a classe neutra também foi considerada.

Já em relação às combinações de *embeddings*, os resultados obtidos na presente pesquisa mostraram que a técnica de concatenação apresentou melhores resultados quando comparada às técnicas de *meta-embeddings* dinâmicos para maioria dos casos, diferentemente da proposta sugerida em (KIELA; WANG; CHO, 2018), onde os *meta-embeddings* apresentaram melhor performance. A diferença é que em (KIELA; WANG; CHO, 2018) as bases de dados utilizadas possuíam informações em Inglês e o problema de PLN em questão era voltado para estabelecer o relacionamento entre sentenças (RTE). Dessa forma, no contexto de AS e para bases de dados em Português-BR, as estratégias de *meta-embedding* utilizadas não trouxeram ganho de performance, apenas redução no tempo de execução, quando comparada à técnica CONCAT.

No que diz respeito à comparação entre os classificadores utilizados nessa pesquisa, dentre os tradicionais de AM, as melhores performance obtidas para todas as bases de dados ficaram concentradas entre a MLP e o *XGBoost*. Com exceção dos dados do Olist *Dataset*, para as demais bases de dados esses classificadores mostraram os melhores desempenhos dentre todos, incluindo a CNN, e ficaram estatisticamente empatados em relação à acurácia, especialmente quando a forma de *embedding* CONCAT foi utilizada. A principal diferença entre as técnicas,

que pode ser utilizada como critério de desempate, é a questão do tempo de processamento, que para o *XGBoost* foi bem maior do que o tempo de execução utilizado pela MLP. Isso pode ser justificado pela arquitetura de árvores do XGBoost, que pode apresentar demora na convergência dos resultados para grandes volumes de dados. A Tabela 47 mostra o concat_*XGBoost* como o melhor classificador para para os dados Buscapé, com 81.05% de acurácia, e o segundo melhor classificador para os dados da B2W, ficando empatado com a MLP em relação à acurácia, ambos com 79.47%.

Os classificadores SVM e Regressão Logística, por sua vez, forneceram os piores desempenhos para a maioria das bases de dados quando as técnicas de *embedding* foram utilizadas, porém apresentaram melhor performance quando a técnica *Bag of Words* foi utilizada para vetorização das palavras. Em relação aos *embeddings* utilizados, no caso do SVM a baixa performance ocorreu de forma unânime, com pouquíssimos valores de acurácia superiores à 60%. Apenas para os dados do Buscapé, quando o *Word2Vec* foi utilizado na vetorização, este classificador forneceu uma acurácia superior a 75%. Esses valores de acurácia para cada tipo de modelo experimental realizado por base de dados utilizada podem ser verificados na Tabela 47 no final deste Capítulo. A Regressão Logística (RL), apesar de ter ficado em penúltimo lugar em relação à acurácia quando comparada aos demais classificadores, foi o algoritmo que obteve as melhores respostas diante das variações das técnicas de *embedding*. Em todos os testes estatísticos realizados, foi possível rejeitar a hipótese nula, de que as distribuições para este classificador foram iguais para os diferentes tipos de *embedding* utilizados. Além disso, o CONCAT forneceu o melhor resultado para este algoritmo em todas as quatro bases de dados utilizadas.

Na comparação entre as técnicas de AM utilizando *embeddings* pré-treinados e a CNN utilizando uma camada de *embedding*, os valores de acurácia obtidos foram muito próximos dos encontrados para os classificadores concat_*XGBoost* e concat_MLP, sem diferença estatística nas distribuições, embora os valores de acurácia média tenham sido maiores para a CNN em todas as bases de dados, com exceção do Buscapé. Além disso, para os dados da base Olist, este algoritmo de aprendizado profundo foi considerado o melhor para a classificação, com uma acurácia média de 84.35%, aproximadamente 2% a mais do que as acurácias dos segundos e terceiros lugares (concat_MLP, com 82.34% e concat_*XGBoost* com 82.26%). O fato do *XGBoost* ter apresentado os melhores resultados na maioria dos casos pode estar diretamente associado à sua estabilidade quando aplicado às bases de dados em estudo. O mesmo pode ter ocorrido para a MLP. Os resultados obtidos pela CNN explicam a superioridade dos modelos

de aprendizado profundo em relação aos de AM, porém o impacto do tempo de processamento desses algoritmos sempre deve ser levado em conta.

Em relação aos *embeddings* pré-treinados utilizados sem combinação, não foi possível estabelecer um padrão de melhor ou pior técnica, uma vez que os valores de acurácia obtidos foram bem diferentes para cada base de dados e algoritmo de AM considerado. Quanto aos *embeddings* combinados, os resultados mostram que a combinação entre os *embeddings* com diferentes distribuições para a maioria das bases de dados através da concatenação dos vetores trouxe ganhos de performance em relação à acurácia, porém associados ao aumento do tempo de execução na maioria dos casos.

Esses resultados são capazes de explicar experimentalmente o quanto mudanças na técnica de vetorização dos dados, em especial neste caso para sentenças escritas em Português, não dependem apenas do contexto dos dados de análise, uma vez que todas as bases de dados utilizadas possuem o mesmo domínio textual, relacionado à avaliação de produtos e ou serviços escritos em sites de *e-commerce*. É possível notar que classificadores diferentes apresentam comportamentos diferentes diante da mudança na técnica de vetorização. Para alguns, como a Regressão Logística e o LGBM, o impacto é mais significativo do que para classificadores como o *XGBoost* e a MLP.

Quanto aos resultados obtidos para as diferentes formas de combinação entre os *embeddings*, é possível que a estratégia de *meta-embeddings* dinâmicos não tenha surtido efeitos significativos especificamente para esses tipos de dados e quando utilizados com essas técnicas. Uma possibilidade futura seria testar as arquiteturas DME e CDME como *embeddings* utilizados juntamente com redes neurais de aprendizado profundo, uma vez que essas técnicas utilizam uma rede teste tipo (LSTM) para realizar as combinações entre os pesos dos vetores.

Tabela 47 – Desempenho de todos os modelos em relação à cada base de dados

Modelo	Acurácia (%)			
	Buscapé Corpus	B2W Reviews	UTLC Movies	Olist *Dataset*
SVM_Glove	52.54	52.12	34.26	55.76
SVM_Wang2Vec	48.54	46.22	35.48	43.37
SVM_Word2Vec	**76.95**	56.34	21.07	51.72
SVM_fast-text	50.86	50.65	33.72	**63.51**
SVM_concat	65.07	59.63	33.29	58.35
SVM_dme	38.04	**61.10**	**56.97**	46.05
SVM_cdme	26.23	51.21	32.78	53.62
RL_Glove	65.84	69.64	54.52	74.58
RL_Wang2Vec	66.50	71.09	54.55	75.61
RL_Word2Vec	64.78	68.85	53.91	73.78
RL_fast-text	67.01	70.58	55.4	74.78
RL_concat	**67.54**	**71.73**	**55.78**	**76.66**
RL_dme	66.67	70.31	54.49	75.21
RL_cdme	66.39	70.40	54.28	75.02
MLP_Glove	80.50	77.48	71	81.41
MLP_Wang2Vec	80.64	78.56	70.97	81.6
MLP_Word2Vec	**80.29**	73.31	70.92	80.32
MLP_fast-text	80.76	78.84	71.04	80.71
MLP_concat	80.99	**79.47**	**71.62**	**82.34**
MLP_dme	80.62	74.79	70.85	80.56
MLP_cdme	80.33	77.90	71.01	80.46
LGBM_Glove	65.46	71.94	58.61	78.15
LGBM_Wang2Vec	71.78	71.58	55.52	78.27
LGBM_Word2Vec	70.54	71.95	53.99	75.64
LGBM_fast-text	70.64	**72.48**	55.4	**78.33**
LGBM_concat	**72.75**	71.68	**59.57**	78.27
LGBM_dme	64.92	72.45	54.07	77.41
LGBM_cdme	67.52	71.92	54.84	77.14
Xgboost_Glove	80.57	78.78	71.24	81.6
Xgboost_Wang2Vec	80.68	79.00	71.31	82.05
Xgboost_Word2Vec	80.05	74.26	71.17	81.14
Xgboost_fast-text	80.72	79.00	71.42	81.89
Xgboost_concat	81.05	**79.47**	71.45	**82.26**
Xgboost_dme	80.56	78.30	71.32	81.75
Xgboost_cdme	80.63	77.76	**71.46**	81.78
CNN	80.99	80.77	72.29	84.35

Fonte: a autora (2022)

6 CONCLUSÕES E TRABALHOS FUTUROS

Nesta dissertação foi abordado o impacto da vetorização dos dados na tarefa de Análise de Sentimentos realizada de forma automática através de classificadores baseados em aprendizagem de máquina tradicionais e de uma rede convolucional de aprendizado profundo, utilizando informações experimentais fornecidas a partir de bases de dados contendo avaliações de produtos e serviços encontrados na *web* em sites de grandes marcas de *E-commerce*. Para isso, foram utilizadas as bases: Olist, B2W e UTLC *Movies*, todas com avaliações escritas em Português-BR.

A principal ideia foi observar o quanto a mudança na forma de traduzir os dados textuais em vetores de pesos compostos por valores numéricos reais, processo de vetorização conhecido como *embedding*, pode contribuir com a mudança na performance de diferentes algoritmos de Aprendizagem de Máquina. Diante disso, foram utilizados vetores de *embeddings* pré-treinados para dados em Português obtidos a partir do repositório de *word embeddings* do NILC. Quatro tipos de vetores de *embedding* de mesma dimensionalidade (d = 100) foram utilizados: *Glove*, *FastText*, *Word2Vec* e *Wang2Vec* na primeira etapa dos experimentos, que buscou avaliar se esses diferentes vetores de *embedding* trariam impacto significativo em cinco classificadores de AM com diversas abordagens: SVM, Regressão Logística, MLP, LGBM e *XGBoost*.

Os resultados obtidos nessa etapa indicam que, para o contexto dos dados utilizados, não foi possível estabelecer uma relação direta entre os vetores de *embedding* e os classificadores de AM. Entretanto, foi possível notar que, para os classificadores *XGBoost* e MLP, a mudança na forma de vetorização, apesar de trazer diferenças nas distribuições estatísticas para algumas das bases de dados, não surtiu muito impacto em relação à variação do valor da acurácia média. Além disso, foi possível concluir que classificadores mais clássicos e comumente utilizados em problemas de classificação binária, como é o caso da SVM e da Regressão Logística foram os que apresentaram os piores resultados e com maiores variações na acurácia média diante da modificação na técnica de *embedding* utilizada. Esse comportamento pode ser justificado pela presença da classe neutra nas bases de dados, cuja predição correta é mais difícil de ser realizada para classificadores especialistas em problemas binários, ainda que tenham sido realizados ajustes paramétricos para este tipo de classificação.

Na segunda etapa dos experimentos, a principal contribuição foi a análise da combinação de vetores de *embeddings* a partir de algumas propostas encontradas na literatura. Três abor-

dagens de combinação foram consideradas: a concatenação dos vetores (CONCAT) e duas técnicas de combinação dinâmicas, que utilizam uma LSTM para encontrar os pesos do vetor resultante na combinação a partir dos pesos dos vetores de entrada. Essas técnicas diferem apenas em relação a considerarem ou não o contexto no qual a palavra está inserida. Denominadas respectivamente como DME e CDME, elas foram aplicadas seguindo os algoritmos descritos em (KIELA; WANG; CHO, 2018). Por fim, realizou-se a comparação entre um modelo de aprendizado profundo com os melhores modelos de AM encontrados para os diferentes *embeddings*. O modelo em questão foi uma CNN, e não foi utilizada nenhuma técnica de *embedding* pré-treinada, os pesos dos vetores foram definidos pela própria camada de *embedding* da rede.

A partir dos resultados obtidos seguindo essas etapas, foi possível responder às questões de pesquisa propostas na Introdução deste texto, a saber:

1. *Dentre os diferentes tipos de algoritmos utilizados em Aprendizagem de Máquina, dos mais tradicionais aos que utilizam aprendizado profundo, há algum que se destaque para as bases de dados utilizadas?*

   A partir dos resultados obtidos no Capítulo anterior, foi possível encontrar um empate entre os classificadores MLP e *XGboost* e a rede de aprendizado profundo CNN, porém com a técnica de MLP caracterizada pelo menor tempo de execução em relação às demais.

2. *Existe a melhor combinação Algoritmo de classificação versus técnica de vetorização de palavras para o Português na área de avaliações de produtos?*

   Considerando os *embeddings* simples, não foi possível encontrar um padrão entre o tipo de *embedding* utilizado e os classificadores de AM, uma vez que os resultados obtidos variaram para cada base de dados de análise. Ainda assim, conseguiu-se verificar que a mudança na técnica de *embedding* impactou diretamente nos resultados para todos os modelos de Regressão Logística, possivelmente devido a seu caráter probabilístico, de modo que as distribuições variaram mais considerando os diferentes pesos de cada vetor de *embedding* estudado.

3. *A combinação entre diferentes técnicas de vetorização resulta em melhor desempenho nos resultados da classificação?*

Foi possível observar que as combinações dos *embeddings Word2Vec* e *FastText* através da concatenação trouxeram melhores resultados que os vetores de *embedding* simples para a maioria dos modelos, em especial a Regressão Logística, em termos de acurácia. Em relação ao tempo de execução, a concatenação apresentou tempo superior às demais formas de combinação na maioria dos casos.

4. *Dentre os métodos de combinação utilizados, combinações mais complexas entregam melhores resultados quando comparadas a uma combinação que realiza uma simples concatenação entre duas formas de vetorização distintas?*

   Conforme informado no item anterior, apenas a concatenação trouxe aumento de performance para a maioria dos classificadores. As técnicas DME e CDME não só não trouxeram ganhos, como também degradaram os resultados em relação aos *embeddings* simples em alguns casos.

Outro possível questionamento comum na área de AM é sobre a superioridade dos algoritmos de aprendizado profundo em relação aos tradicionais. Com base nos resultados dessa pesquisa, não foi possível concluir essa hipótese em relação ao CNN, uma vez que, apesar de apresentar a maior acurácia na maioria dos casos, os ganhos foram muito pequenos em relação aos modelos *XGBoost* e MLP utilizando a concatenação de *embeddings*, assim como não foi possível rejeitar a hipótese de que as distribuições são estatisticamente iguais, com exceção da base Olist, onde esse ganho foi mais significativo.

Em relação às contribuições fornecidas por este trabalho, podem ser considerados dois tópicos principais: o primeiro diz respeito à contribuição para a literatura na área de Análise de Sentimentos, no que diz respeito à experimentações com o Português Brasileiro, uma vez que, considerando a riqueza linguística e popularidade deste idioma, ainda existem poucos trabalhos acadêmicos que realizem experimentos com dados em Português de forma mais aprofundada. Em segundo lugar, o estudo de algumas das diferentes formas de realizar combinações de *embeddings*, mostrando que combinações simples como uma concatenação podem fornecer resultados tão bons quanto a utilização de um modelo mais complexo e custoso computacionalmente, como é o caso das redes de aprendizado profundo usadas nos *meta-embeddings*. Além disso, mesmo diante das degradações de performance obtidas para os modelos que utilizaram *embeddings* dinâmicos, ainda assim esta forma de vetorização pode ser testada em outras áreas da PLN, uma vez que trouxe bons resultados para os cenários considerados em outras pesquisas.

Como limitações enfrentadas durante o projeto de pesquisa, a principal foi a escassez de bases de dados em Português brasileiro que tivesse volumetria significativa com classes já rotuladas, em especial para problemas não binários. Outro ponto foi a comparação com algoritmos de aprendizado profundo: devido ao tempo de execução dessas técnicas e considerando o processo de validação cruzada, optou-se por utilizar apenas a CNN com a camada de *embedding*, porém mais classificadores de aprendizado profundo poderiam ter sido incluídos. No que diz respeito ao estudo comparativo realizado, este trabalho utilizou a acurácia como métrica de desempenho, mensurada na base de teste de cada modelo. Uma vez que os modelos de classificação desta pesquisa apresentam múltiplas classes, utilizar a acurácia como métrica de desempate pode ser um problema, ainda que tenha sido realizado o balanceamento das classes no conjunto de treinamento através de técnicas de re-amostragem (*resample*).

Desse modo, no que diz respeito à trabalhos futuros, uma comparação mais justa a ser realizada consiste em utilizar o F1-Score como métrica de avaliação. Além disso, um dos estudos que podem ser realizados de modo a complementar os resultados obtidos até aqui diz respeito a uma análise comparativa entre algoritmos de aprendizado profundo utilizando suas próprias camadas de *embedding* e utilizando vetores de *embeddings* pré-treinados. Além disso, experimentar mais formas de combinações de *embeddings* existentes na literatura para outros problemas de PLN e aplicá-las ao contexto da Análise de Sentimentos, incluindo outras formas de codificação de texto mais recentes utilizadas na literatura, baseadas em Transformers, como é o caso do BERT (acrônimo para *Bidirectional Encoder Representations from Transformers)* e sua versão voltada para a Língua Portuguesa, denominada BERTimbau (SOUZA; NOGUEIRA; LOTUFO, 2020). Outro ponto diz respeito à investigar uma possível metodologia de tratamento dos dados específica para textos em Português-BR, uma vez que este trabalho utilizou técnicas de pré-processamento mais genéricas semelhantes às utilizadas para dados em Inglês. Ainda no âmbito de customização para o Português, outra possibilidade futura é a de criação de uma técnica de *embedding* customizada para este idioma, a partir da combinação de diferentes tipos de *embeddings* que comprovadamente trazem bons ganhos para problemas de PLN envolvendo o Português brasileiro.

REFERÊNCIAS

AGARWALL, B.; NAYAK, R.; MITTAL, N.; PATNAIK, S. *Deep Learning - Based Aproaches for Sentiment Analysis*. [S.l.]: Springer, 2020.

AIRES, J. P.; PADILHA, C.; QUEVEDO, C.; MENEGUZZI, F. A deep learning approach to classify aspect-level sentiment using small datasets. In: IEEE. *2018 International Joint Conference on Neural Networks (IJCNN)*. [S.l.], 2018. p. 1–8.

ANGIANI, G.; FERRARI, L.; FONTANINI, T.; FORNACCIARI, P.; IOTTI, E.; MAGLIANI, F.; MANICARDI, S. A comparison between preprocessing techniques for sentiment analysis in twitter. In: *KDWeb*. [S.l.: s.n.], 2016.

ARAUJO, D. da C. *Avaliação de Comitês com Classificadores Tradicionais e Profundos para Análise de Sentimentos*. Dissertação (Mestrado em Ciência da Computação) — Universidade Federal de Pernambuco, 2019.

ARAUJO, G. D. de. *Análise de sentimento de mensagens do Twitter em português brasilei-ro relacionadas a temas de saúde*. Dissertação (Programa de Pós-graduação em Gestão e Informática em Saúde) — Universidade Federal de São Paulo, 2014.

AVANCO, L. V.; NUNES, M. d. G. V. Lexicon-based sentiment analysis for reviews of products in brazilian portuguese. In: *2014 Brazilian Conference on Intelligent Systems*. [S.l.: s.n.], 2014. p. 277–281.

BARLA, N. *The Ultimate Guide to Word Embeddings*. 2021. Disponível em: <https://neptune.ai/blog/word-embeddings-guide>.

BATISTA, G. E. *Pré-processamento de dados em aprendizado de máquina supervisionado.* Tese (Doutorado) — Universidade de São Paulo, 2003.

BENGIO, Y.; DUCHARME, R.; VINCENT, P.; JANVIN, C. A neural probabilistic language model. *J. Mach. Learn. Res.*, JMLR.org, v. 3, n. null, p. 1137–1155, mar 2003. ISSN 1532-4435.

BERTAGLIA, T. F. C.; NUNES, M. das G. V. *Exploring Word Embeddings for Unsupervised Textual User-Generated Content Normalization*. 2017.

BOJANOWSKI, P.; GRAVE, E.; JOULIN, A.; MIKOLOV, T. *Enriching Word Vectors with Subword Information*. 2017.

BOLLEGALA, D.; HAYASHI, K.; KAWARABAYASHI, K. Think globally, embed locally - locally linear meta-embedding of words. *CoRR*, abs/1709.06671, 2017. Disponível em: <http://arxiv.org/abs/1709.06671>.

BOOK, D. L. *Capítulo 76 – O Que é BERT (Bidirectional Encoder Representations from Transformers)?* 2021. Disponível em: <https://www.deeplearningbook.com.br/o-que-e-bert-bidirectional-encoder-representations-from-transformers/>.

BRITTO, L.; PACíFICO, L. Análise de sentimentos para revisões de aplicativos mobile em português brasileiro. In: *Anais do XVI Encontro Nacional de Inteligência Artificial e Computacional*. Porto Alegre, RS, Brasil: SBC, 2020. p. 1080–1090. ISSN 0000-0000. Disponível em: <https://sol.sbc.org.br/index.php/eniac/article/view/9359>.

CARVALHO, C. M. A. de. *Estudo Comparativo de Análise de Sentimento Aplicado à Notícias Políticas*. Dissertação (Mestrado em Engenharia Elétrica) — Universidade Federal do Maranhão, 2018.

CENTER, S. D. S. 2021. Disponível em: <https://datascience.ch/the-deep-dive-of-natural-language-processing/>.

CIRQUEIRA, D.; PINHEIRO, M.; JACOB, A. F. L.; LOBATO, F. M. F.; SANTANA, Á. L. de. A literature review in preprocessing for sentiment analysis for brazilian portuguese social media. *2018 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*, p. 746–749, 2018.

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

DAOUD, E. A. Comparison between xgboost, lightgbm and catboost using a home credit dataset. *International Journal of Computer and Information Engineering*, v. 13, n. 1, p. 6–10, 2019.

DAS, S.; CHEN, M. Yahoo! for amazon: Extracting market sentiment from stock message boards. In: BANGKOK, THAILAND. *Proceedings of the Asia Pacific finance association annual conference (APFA)*. [S.l.], 2001. v. 35, p. 43.

DEMŠAR, J. Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research*, JMLR. org, v. 7, p. 1–30, 2006.

DOVAL, Y.; VILARES, J.; GÓMEZ-RODRÍGUEZ, C. Towards robust word embeddings for noisy texts. *Applied Sciences*, Multidisciplinary Digital Publishing Institute, v. 10, n. 19, p. 6893, 2020.

DUDA, R. O.; HART, P. E.; STORK, D. G. *Pattern Classification*. [S.l.]: John Wiley Sons, 2012.

EL-KASSAS, W. S.; SALAMA, C. R.; RAFEA, A. A.; MOHAMED, H. K. Automatic text summarization: A comprehensive survey. *Expert Systems with Applications*, v. 165, p. 113679, 2021. ISSN 0957-4174. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0957417420305030>.

FERREIRA, M. A. D. *Uma abordagem para extração automática de learning analytics relacionadas à colaboração em fóruns educacionais*. Dissertação (Mestrado em Informática Aplicada) — Universidade Federal Rural de Pernambuco, 2018.

GARG, S.; VERMA, N. Study of sentiment classification techniques. *International Journal on Computer Science and Engineering*, v. 6, 2018.

GHARIBI, M. *The Magic Behind Embedding Models*. 2019. Disponível em: <https://www.deeplearningbook.com.br/o-que-e-bert-bidirectional-encoder-representations-from-transformers/>.

GROSSBERG, S. Recurrent neural networks. *Scholarpedia*, v. 8, n. 2, p. 1888, 2013.

HARTMANN, N.; AVANÇO, L.; FILHO, P. B.; DURAN, M. S.; NUNES, M. D. G. V.; PARDO, T.; ALUÍSIO, S. A large corpus of product reviews in portuguese: Tackling out-of-vocabulary words. In: *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*. [S.l.: s.n.], 2014. p. 3865–3871.

HARTMANN, N.; FONSECA, E.; SHULBY, C.; TREVISO, M.; RODRIGUES, J.; ALUISIO, S. *Portuguese Word Embeddings: Evaluating on Word Analogies and Natural Language Tasks*. 2017.

HOSSIAN, Z. 2018. Disponível em: <https://towardsdatascience.com/a-complete-guide-to-natural-language-processing-nlp-c91f1cfd3b0c>.

HUSSEIN, D. M. E.-D. M. A survey on sentiment analysis challenges. *Journal of King Saud University-Engineering Sciences*, Elsevier, v. 30, n. 4, p. 330–338, 2018.

INGLESA, C. 2020. Disponível em: <https://blog.culturainglesa.com.br/ingles/>.

JAIN, P. K.; SARAVANAN, V.; PAMULA, R. A hybrid cnn-lstm: A deep learning approach for consumer sentiment analysis using qualitative user-generated contents. *Transactions on Asian and Low-Resource Language Information Processing*, ACM New York, NY, v. 20, n. 5, p. 1–15, 2021.

JHA, A. *Vectorization Techniques in NLP*. 2021. Disponível em: <https://neptune.ai/blog/vectorization-techniques-in-nlp-guide>.

JOHNSON, D. 2021. Disponível em: <https://www.guru99.com/nlp-tutorial.html>.

KADHIM, A. I. Survey on supervised machine learning techniques for automatic text classification. *Artificial Intelligence Review*, Springer, v. 52, n. 1, p. 273–292, 2019.

KAIBI, I.; NFAOUI, E. H.; SATORI, H. A comparative evaluation of word embeddings techniques for twitter sentiment analysis. In: *2019 International Conference on Wireless Technologies, Embedded and Intelligent Systems (WITS)*. [S.l.: s.n.], 2019. p. 1–4.

KAIBI, I.; NFAOUI, E. H.; SATORI, H. Sentiment analysis approach based on combination of word embedding techniques. In: BHATEJA, V.; SATAPATHY, S. C.; SATORI, H. (Ed.). *Embedded Systems and Artificial Intelligence*. Singapore: Springer Singapore, 2020. p. 805–813. ISBN 978-981-15-0947-6.

KANSAON, D. P.; BRANDAO, M. A.; PINTO, S. A. d. P. Analise de algoritmos de classificação para detecção de emoções em tweets em português brasileiro. *iSys - Brazilian Journal of Information Systems*, v. 12, n. 3, p. 116–138, ago. 2019.

KE, G.; MENG, Q.; FINLEY, T.; WANG, T.; CHEN, W.; MA, W.; YE, Q.; LIU, T.-Y. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, v. 30, p. 3146–3154, 2017.

KHALID, S.; KHALIL, T.; NASREEN, S. A survey of feature selection and feature extraction techniques in machine learning. In: *2014 Science and Information Conference*. [S.l.: s.n.], 2014. p. 372–378.

KIELA, D.; WANG, C.; CHO, K. Dynamic meta-embeddings for improved sentence representations. *arXiv preprint arXiv:1804.07983*, 2018.

KRUMM, J.; DAVIES, N.; NARAYANASWAMI, C. User-generated content. *IEEE Pervasive Computing*, IEEE, v. 7, n. 4, p. 10–11, 2008.

KUMAR, E. *Natural Language Processing*. [S.l.]: I.K International Publishing House Pvt. Ltd., 2011.

LAVALLEY, M. P. Logistic regression. *Circulation*, Am Heart Assoc, v. 117, n. 18, p. 2395–2399, 2008.

LEE, H.; GROSSE, R.; RANGANATH, R.; NG, A. Y. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In: *Proceedings of the 26th annual international conference on machine learning*. [S.l.: s.n.], 2009. p. 609–616.

LIN, X. Sentiment analysis of e-commerce customer reviews based on natural language processing. In: *Proceedings of the 2020 2nd International Conference on Big Data and Artificial Intelligence*. [S.l.: s.n.], 2020. p. 32–36.

LING, W.; DYER, C.; BLACK, A. W.; TRANCOSO, I. Two/too simple adaptations of Word2Vec for syntax problems. In: *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Denver, Colorado: Association for Computational Linguistics, 2015. p. 1299–1304. Disponível em: <https://aclanthology.org/N15-1142>.

LIU, B. Sentiment analysis and subjectivity. In: *Handbook of Natural Language Processing, Second Edition. Taylor and Francis Group, Boca*. [S.l.: s.n.], 2010.

MAHDAVINEJAD, M. S.; REZVAN, M.; BAREKATAIN, M.; ADIBI, P.; BARNAGHI, P.; SHETH, A. P. Machine learning for internet of things data analysis: A survey. *Digital Communications and Networks*, Elsevier, v. 4, n. 3, p. 161–175, 2018.

MARTINS, G. S.; OLIVEIRA, A. de P.; MOREIRA, A. Sentiment analysis applied to hotels evaluation. In: SPRINGER. *International Conference on Computational Science and Its Applications*. [S.l.], 2017. p. 710–716.

MEDHAT, W.; HASSAN, A.; KORASHY, H. Sentiment analysis algorithms and applications: A survey. *Ain Shams Engineering Journal*, v. 5, n. 4, p. 1093–1113, 2014. ISSN 2090-4479. Disponível em: <https://www.sciencedirect.com/science/article/pii/S2090447914000550>.

MEJOVA, Y. Sentiment analysis: An overview. *University of Iowa, Computer Science Department*, 2009.

MIKOLOV, T.; CHEN, K.; CORRADO, G.; DEAN, J. *Efficient Estimation of Word Representations in Vector Space*. 2013.

MIKOLOV, T.; SUTSKEVER, I.; CHEN, K.; CORRADO, G. S.; DEAN, J. Distributed representations of words and phrases and their compositionality. In: BURGES, C. J. C.; BOTTOU, L.; WELLING, M.; GHAHRAMANI, Z.; WEINBERGER, K. Q. (Ed.). *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2013. v. 26. Disponível em: <https://proceedings.neurips.cc/paper/2013/file/9aa42b31882ec039965f3c4923ce901b-Paper.pdf>.

MITRANI, A. 2019. Disponível em: <https://towardsdatascience.com/how-machines-understand-text-994cf4ca9f49>.

NASCIMENTO, P. de A. *Aplicando Ensemble para Classificação de Textos Curtos em Português do Brasil*. Dissertação (Mestrado em Ciência da Computação) — Universidade Federal de Pernambuco, 2019.

NASEEM, U.; RAZZAK, I.; KHAN, S. K.; PRASAD, M. A comprehensive survey on word representation models: From classical to state-of-the-art word representation language models. *CoRR*, abs/2010.15036, 2020. Disponível em: <https://arxiv.org/abs/2010.15036>.

NEY, H. *Natural Language Processing*. 2019. Disponível em: <http://www-i6.informatik.rwth-aachen.de/>.

NOVO, B. N. *A língua portuguesa surgiu na Galícia*. 2021. Disponível em: <https://jus.com.br/artigos/93905/a-lingua-portuguesa-surgiu-na-galicia-espanha>.

PANG, B.; LEE, L. Opinion mining and sentiment analysis. *Found. Trends Inf. Retr.*, Now Publishers Inc., Hanover, MA, USA, v. 2, n. 1–2, p. 1–135, jan 2008. ISSN 1554-0669. Disponível em: <https://doi.org/10.1561/1500000011>.

PANTOLA, P. *Natural Language Processing: Text Data Vectorization*. 2018. Disponível em: <https://medium.com/@paritosh_30025/natural-language-processing-text-data-vectorization-af2520529cf7>.

PENNINGTON, J.; SOCHER, R.; MANNING, C. D. Glove: Global vectors for word representation. In: *Empirical Methods in Natural Language Processing (EMNLP)*. [s.n.], 2014. p. 1532–1543. Disponível em: <http://www.aclweb.org/anthology/D14-1162>.

PEREIRA, D. A. A survey of sentiment analysis in the portuguese language. *Artificial Intelligence Review*, Springer, v. 54, n. 2, p. 1087–1115, 2021.

PETROLITO, R.; DELL'ORLETTA, F. Word embeddings in sentiment analysis. In: *CLiC-it*. [S.l.: s.n.], 2018.

R, S. C.; DUBEY, R. K. Meta-embeddings for natural language inference and semantic similarity tasks. *arXiv e-prints*, p. arXiv–2012, 2020.

RABELLO, E. B. *Cross Validation: Avaliando seu modelo de Machine Learning*. 2019. Disponível em: <https://medium.com/@edubrazrabello/cross-validation-avaliando-seu-modelo-de-machine-learning-1fb70df15b78>.

REAL, L.; OSHIRO, M.; MAFRA, A. B2w-reviews01-an open product reviews corpus. In: *the Proceedings of the XII Symposium in Information and Human Language Technology*. [S.l.: s.n.], 2019. p. 200–208.

REZAEINIA, S. M.; RAHMANI, R.; GHODSI, A.; VEISI, H. Sentiment analysis based on improved pre-trained word embeddings. *Expert Systems with Applications*, Elsevier, v. 117, p. 139–147, 2019.

SILIPO, R. *Ensemble Models: Bagging & Boosting*. 2020. Disponível em: <https://medium.com/analytics-vidhya/ensemble-models-bagging-boosting-c33706db0b0b>.

SILVA, E. P. da; MALHEIROS, Y. Um conjunto de dados extraıdo do twitter para análise de sentimentos na lıngua portuguesa. 2019.

SILVA, J. R. da; CASELI, H. de M. Generating sense embeddings for syntactic and semantic analogy for portuguese. *CoRR*, abs/2001.07574, 2020. Disponível em: <https://arxiv.org/abs/2001.07574>.

SILVA, N. F. F. D.; COLETTA, L. F. S.; HRUSCHKA, E. R. A survey and comparative study of tweet sentiment analysis via semi-supervised learning. *ACM Comput. Surv.*, Association for Computing Machinery, New York, NY, USA, v. 49, n. 1, jun 2016. ISSN 0360-0300. Disponível em: <https://doi.org/10.1145/2932708>.

SINGH, H. *Understanding Gradient Boosting Machines*. 2018. Disponível em: <https://towardsdatascience.com/understanding-gradient-boosting-machines-9be756fe76ab>.

SOUSA, R. F.; BRUM, H. B.; NUNES, M. d. G. V. A bunch of helpfulness and sentiment corpora in brazilian portuguese. In: SOCIEDADE BRASILEIRA DE COMPUTAçãO. *Proceedings of Symposium in Information and Human Language Technology - STIL*. [S.l.], 2019.

SOUZA, E.; VITÓRIO, D.; CASTRO, D.; OLIVEIRA, A. L.; GUSMÃO, C. Characterizing opinion mining: A systematic mapping study of the portuguese language. In: SPRINGER. *International Conference on Computational Processing of the Portuguese Language*. [S.l.], 2016. p. 122–127.

SOUZA, F.; NOGUEIRA, R.; LOTUFO, R. Bertimbau: Pretrained bert models for brazilian portuguese. In: CERRI, R.; PRATI, R. C. (Ed.). *Intelligent Systems*. Cham: Springer International Publishing, 2020. p. 403–417. ISBN 978-3-030-61377-8.

SOUZA, F. et al. Sentiment analysis on brazilian portuguese user reviews. *arXiv preprint arXiv:2112.05459*, 2021.

TAN, A.-H.; MUI, H.; TERRACE, K. Text mining: The state of the art and the challenges. In: . [S.l.: s.n.], 2000.

TONG, R. M. An operational system for detecting and tracking opinions in on-line discussion. In: *Working Notes of the ACM SIGIR 2001 Workshop on Operational Text Classification*. [S.l.: s.n.], 2001. v. 1, n. 6.

UNNIKRISHNAN. *How sklearn's Tfidfvectorizer Calculates tf-idf Values*. 2021. Disponível em: <https://www.analyticsvidhya.com/blog/2021/11/how-sklearns-tfidfvectorizer-calculates-tf-idf-values/>.

WANG, J. C.; HASTIE, T. Boosted varying-coefficient regression models for product demand prediction. *Journal of Computational and Graphical Statistics*, Taylor & Francis, v. 23, n. 2, p. 361–382, 2014.

WANG, X.; JIANG, Y.; BACH, N.; WANG, T.; HUANG, Z.; HUANG, F.; TU, K. Automated concatenation of embeddings for structured prediction. *ArXiv*, abs/2010.05006, 2021.

YIN, W.; SCHÜTZE, H. Learning meta-embeddings by using ensembles of embedding sets. *arXiv preprint arXiv:1508.04257*, 2015.

ZHANG, L.; WANG, S.; LIU, B. Deep learning for sentiment analysis: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, Wiley Online Library, v. 8, n. 4, p. e1253, 2018.