FERNANDA COSTA E SILVA

PROCESSAMENTO DE LINGUAGEM NATURAL NO SEGMENTO DE *E-COMMERCE*:

UMA APLICAÇÃO *FEW SHOT LEARNING* COM REDES NEURAIS SIAMESAS

LAVRAS – MG

2022

FERNANDA COSTA E SILVA

PROCESSAMENTO DE LINGUAGEM NATURAL NO SEGMENTO DE *E-COMMERCE*:
UMA APLICAÇÃO *FEW SHOT LEARNING* COM REDES NEURAIS SIAMESAS

Dissertação apresentada à Universidade Federal de Lavras, como parte das exigências do Programa de Pós-Graduação em Engenharia de Sistemas e Automação, área de concentração Engenharia de Sistemas e Automação, para a obtenção do título de Mestre.

Prof. DSc. Danton Diego Ferreira
Orientador

Prof. DSc. Bruno Henrique Groenner Barbosa
Coorientador

LAVRAS – MG
2022

Ficha catalográfica elaborada pelo Sistema de Geração de Ficha Catalográfica da Biblioteca Universitária da UFLA, com dados informados pelo(a) próprio(a) autor(a).

Silva, Fernanda Costa e.
Processamento de linguagem natural no segmento de E-commerce : uma aplicação few shot learning com redes neurais siamesas / Fernanda Costa e Silva. - Lavras : UFLA, 2022.
112 p. : il.

Orientador(a): Prof. DSc.Danton Diego Ferreira.
Coorientador(a): Prof. DSc. Bruno Henrique Groenner Barbosa.
Dissertação (Mestrado Acadêmico) - Universidade Federal de Lavras, 2022.
Bibliografia.

1. Redes neurais siamesas. 2. One shot learning. 3. Processamento de linguagem natural. I. Ferreira, Danton Diego. II. Barbosa, Bruno Henrique Groenner. III. Título.

FERNANDA COSTA E SILVA

PROCESSAMENTO DE LINGUAGEM NATURAL NO SEGMENTO DE *E-COMMERCE*: UMA APLICAÇÃO *FEW SHOT LEARNING* COM REDES NEURAIS SIAMESAS

Dissertação apresentada à Universidade Federal de Lavras, como parte das exigências do Programa de Pós-Graduação em Engenharia de Sistemas e Automação, área de concentração Engenharia de Sistemas e Automação, para a obtenção do título de Mestre.

APROVADA em 14 de Abril de 2022.

Prof. DSc. Danton Diego Ferreira	UFLA
Prof. DSc. Giovani Bernardes Vitor	UNIFEI
Prof. DSc. Belisario Nina Huallpa	UFLA

Prof. DSc. Danton Diego Ferreira
Orientador

Prof. DSc. Bruno Henrique Groenner Barbosa
Co-Orientador

LAVRAS – MG
2022

RESUMO

O número de empresas que disponibilizam seus produtos para compra online tem aumentado, fazendo com que novas ofertas apareçam a todo momento. Entretanto, não há um padrão entre a descrição dos produtos fornecida pelos vendedores, o que pode levar um produto a ser colocado em uma categoria diferente daquela a que ele pertence e gerar uma experiência de compra ruim. As empresas que trabalham com comércio eletrônico podem utilizar o grande volume de dados gerados nas diversas transações realizadas na *internet* para construir perfis de usuário e fazer recomendações de produtos personalizadas. Assim sendo, soluções aplicando o processamento de linguagem natural têm o potencial de resolver problemas relacionados ao *E-commerce* e, também, otimizar boa parte dos processos. A linha de pesquisa abordada é de estudo e aprimoramento de sistemas de inteligência artificial para *E-commerce*. Foram analisadas e desenvolvidas técnicas de classificação de dados não estruturados, considerando o problema enfrentado em plataformas de comércio *online*, já que novos produtos cadastrados podem ser classificados erroneamente enquanto suas classes ainda forem pouco representativas na base de dados. Essa é uma situação em que pode ser aplicada algoritmos de aprendizado *one/few-shot learning*, no qual um classificador deve aprender informações relevantes à classificação das amostras utilizando uma ou algumas amostras de uma classe durante seu treinamento. A quantidade de ferramentas eficientes para lidar com tal situação é limitada, pois os métodos de classificação convencionais não conseguem aprender e estabelecer relações significativas a partir de poucos dados de treinamento. Neste trabalho é proposto o uso de um classificador com redes neurais siamesas para classificar classes novas num problema de *E-commerce*. Foram testadas diferentes topologias para a rede interna da rede siamesa, assim como diferentes abordagens para a escolha do representante usado como referência pela rede, sendo proposta a escolha aleatória, com centroide e com *ensemble* de representantes. O classificador proposto com escolha de representante feita com centroide obteve 98% de acurácia ao lidar com um problema de 6 classes e menos de 400 amostras. Para uma base de dados de aproximadamente 4000 amostras e 452 classes o modelo de rede siamesa com estrutura da rede interna de três camadas utilizando a técnica de *DropOut* em uma das camadas e o representante sendo o centroide obteve melhor resultado dentre as opções testadas, tendo 90,31% de acerto, contra 83,62% do modelo de rede siamesa com representante aleatório e 81,81% do K-Nearest Neighbors (KNN). Como trabalhos futuros podem ser estudadas estratégias para melhorar o desempenho do modelo, tais como a formação de pares de treino maximizando as diferenças entre as classes, ao invés da combinação aleatória das amostras. Além disso, podem ser desenvolvidos diferentes extratores de características para os dados das plataformas de vendas *online* pois um extrator que gere características com menor dimensão contribui para a redução da complexidade do classificador, o que pode levar a uma economia no uso de servidores.

Palavras-chave: Mineração de dados. One shot learning. Redes neurais siamesas. Extração de características. Processamento de linguagem natural. Classificação. Aprendizado de máquina. E-commerce.

ABSTRACT

The number of companies making their products available for purchase online has increased, causing new offers to appear all the time. However, there is no pattern between the description of products provided by sellers, which can lead to a product being placed in a different category from the one to which it belongs and generating a poor shopping experience. E-commerce companies can use the large volume of data generated in the various transactions carried out on the Internet to build user profiles and make personalized product recommendations. Therefore, solutions applying natural language processing have the potential to solve problems related to E-commerce and also to optimize a good part of the processes. The issue addressed in this project is the study and improvement of artificial intelligence systems for E-commerce. Unstructured data classification techniques were analyzed and developed, considering the problem faced in online commerce platforms, since new registered products can be misclassified, while their classes are still unrepresentative in the database. This is a situation where one/few-shot learning algorithms can be applied, in which a classifier must learn information relevant to the classification of samples using one or a few samples of a class during its training. The amount of efficient tools to deal with such a situation is limited, as conventional classification methods cannot learn and establish meaningful relationships from a few training data. In this work, it is proposed to use a classifier with Siamese neural networks to classify new classes in an E-commerce problem. Different topologies were tested for the internal network of the Siamese network, as well as different approaches for choosing the representative sample used as a reference for each class, being proposed the random choice, with centroid and with ensemble of representatives. The proposed classifier with representative choice made with the centroid calculation obtained 98% accuracy when dealing with a problem of 6 classes and less than 400 samples. For a larger database, with approximately 4000 samples and 452 classes, the model with a three-layer internal network structure using the DropOut technique in one of the layers and the representative being the calculated centroid the Siamese network obtained the best result among the tested options, with 90.31% of correct answers, against 83.62% of the random representative sample and 81.81% using the K-Nearest Neighbors (KNN) algorithm. as future works, strategies can be studied to improve the performance of the model, such as the formation of training pairs that maximize the differences between classes, instead of randomly combining samples. Different feature extractors for data from online sales platforms can also be developed, since an extractor that delivers features with a smaller dimension contributes to the reduction of the classifier's complexity, which can result in savings in server usage.

Keywords: Data mining. One shot learning. Siamese neural networks. Feature extraction. Natural language processing. Classification. Machine learning. E-commerce.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

SUMÁRIO

1 INTRODUÇÃO

A área de Inteligência Artificial conhecida como Processamento de Linguagem Natural (PLN) é o núcleo da tradução automática e estuda a capacidade e as limitações de uma máquina em entender a linguagem dos seres humanos (ZONG; HONG, 2018). É uma área de pesquisa que desperta o interesse das empresas, pois as ferramentas que são desenvolvidas para realizar a análise de textos e extrair informações podem ser utilizadas para automação de seus processos e também para entender seus clientes e melhorar o relacionamento entre cliente e empresa.

Os impactos da pandemia causada pelo novo coronavírus (COVID-19) foram sentidos em todas as áreas. Embora seja um vírus com baixa letalidade, apresenta alta taxa de transmissão (ZHU et al., 2020). Entre as medidas para contenção do avanço do vírus encontra-se o isolamento social. No trabalho de Kim (2020) é relatado como a pandemia acelerou o crescimento do *E-commerce* (ou comércio eletrônico, em português). Um estudo conduzido com 1990 pessoas nos EUA revelou que 9% das pessoas fizeram uma compra online pela primeira vez durante a pandemia e 62% já realizavam compras online anteriormente (CONSULT, 2020).

Com a tendência das empresas disponibilizarem seus produtos para compra online, o número de ofertas tem aumentado consideravelmente. Por não haver um padrão entre a descrição dos produtos fornecida pelos vendedores e os termos buscados pelos consumidores, as experiências de compra e venda podem ser ruins, levando a resultados desagradáveis para ambas as partes. Além disso, a não padronização das descrições pode levar um produto a ser colocado em uma categoria diferente daquela a que ele pertence.

Outro problema encontrado é a categorização dos itens que fazem parte do catálogo de produtos, porém com pouca representatividade na base de dados. Isso gera dificuldades para que os algoritmos de classificação aprendam a diferenciá-los corretamente.

Buscando melhorar o processo de compras *online*, o aprendizado de máquina começou a ser aplicado para realizar o entendimento semântico de textos para construir perfis de usuário e descobrir suas preferências, na expectativa de melhorar a recomendação de produtos e gerar uma melhor experiência de compra (CHEN; WANG, 2013).

No artigo de Edosio (2014) é apresentado um estudo de caso de diversas empresas que utilizam plataformas de *E-commerce* e como isso aumenta a vantagem competitiva. Considerando o grande volume de dados relacionados ao comércio eletrônico, vê-se a importância do estudo do processamento de linguagem natural aplicado ao segmento de *E-commerce*. Neste contexto, o entendimento semântico de textos é de suma importância, o que pode permitir a

adequada categorização de produtos, assim como uma interpretação mais acertada dos desejos do consumidor.

As plataformas de comércio eletrônico enfrentam um desafio para classificação de seus produtos nos nichos adequados, pois existem muitos produtos nas bases de dados e podem ocorrer algumas classes com poucas amostras. O treinamento de classificadores tradicionais nesses casos é uma tarefa complexa, pois esses classificadores devem acertar também a classificação dessas classes com pouca representatividade na base de dados. Além disso, quando há a inclusão de novas classes, o modelo precisa ser retreinado para contemplá-las, o que é um processo computacionalmente custoso. Nesse contexto, são interessantes as abordagens de treinamento e classificação *Few Shot Learning* (FSL).

O problema *one/few-shot learning* é um tipo de problema de categorização, no qual o classificador deve aprender informações relevantes à classificação das amostras utilizando apenas uma ou algumas amostras de uma classe durante seu treinamento. O número de ferramentas eficientes para lidar com tal situação é limitado, pois os métodos de classificação convencionais não conseguem aprender e estabelecer relações significativas entre as amostras a partir de poucos dados de treinamento.

Diante do exposto, vê-se a importância do estudo do processamento de linguagem natural aplicado ao segmento de *E-commerce* no sentido de torná-lo cada vez mais automático e cognitivo. A linha de pesquisa abordada neste projeto é de estudo e aprimoramento de sistemas de inteligência artificial sendo necessário um levantamento e estudo das técnicas atuais de processamento de linguagem natural e suas aplicações no comércio eletrônico. Considerando que o problema *one/few-shot learning* é frequente na área de *E-commerce*, também podem ser desenvolvidos novos métodos de classificação especialistas inspirados neste problema.

1.1 Objetivos

O objetivo geral deste trabalho é o estudo e desenvolvimento de sistemas de inteligência artificial utilizando o processamento de linguagem natural para análise dos dados provenientes do comércio eletrônico. Esse sistema, inspirado nos modelos *one/few-shot learning*, será utilizado para classificação do produto no nicho correto, independentemente da categoria originalmente atribuída pelo anunciante.

O classificador desenvolvido é voltado para a resolução dos problemas encontrados na classificação de produtos em sistemas de *E-commerce*. Os objetivos específicos incluem:

a) o estudo e análise de algoritmos de *machine learning* para desenvolvimento de um classificador baseado nas redes siamesas para classificação de novos produtos em categorias com poucas amostras para treinamento.

b) a análise do efeito de diferentes configurações da rede interna da rede neural siamesa utilizada no classificador.

c) o estudo de estratégias para escolha do melhor representante de cada classe.

1.2 Justificativa

O número de ofertas *online* tem crescido, assim como a procura por essa modalidade de compra. Entretanto, não há um padrão de descrição para os anunciantes, fazendo com que o mesmo produto possa ser anunciado de forma diferente por mais de um vendedor em plataformas de *E-commerce*. Isso dificulta o processo de categorização e padronização de produtos, impactando diretamente nas vendas e também na manutenção desses sistemas. Para essas plataformas de venda, é interessante a implantação de um sistema que classifique o produto no nicho correto, independentemente do nicho originalmente atribuído pelo vendedor no momento da criação do anúncio na plataforma de vendas.

Um *marketplace* [1] pode ter um classificador já treinado que faça essa categorização dos novos produtos. Entretanto, se um novo nicho for criado, é necessário que esse classificador seja retreinado com a inclusão da nova classe (nicho), para que essa classe passe a ser considerada durante o processo de classificação. O retreino do modelo é computacionalmente custoso, além de ser inviável sua realização a cada novo nicho criado. Além disso, o classificador pode não ter um bom desempenho, pois terá poucos exemplares da nova classe e muitos exemplares de outras classes na fase de treinamento, ocasionando um problema de classes desbalanceadas que pode levar a uma classificação tendenciosa e errada dos novos produtos.

1.3 Hipótese

Considerando o contexto e justificativa expostos, propõe-se o uso de um classificador baseado nas redes siamesas inspirado no modelo *one/few-shot learning*, aplicado ao *E-commerce*. As lojas virtuais apresentam um modelo dinâmico, em que novas categorias podem ser criadas

[1] O *marketplace* é uma espécie de loja virtual mediado por uma empresa, que permite que lojistas se cadastrem e vendam seus produtos.

a todo momento. Por isso, buscam-se alternativas para a classificação de novos produtos nessas novas categorias, sem que seja necessário o treinamento de novos modelos a cada nova categoria criada. Também é necessário que o modelo tenha um bom desempenho ao classificar novas ofertas de classes que tenham poucas amostras já rotuladas.

1.4 Estrutura do trabalho

Este trabalho está organizado da seguinte forma: o referencial teórico, no Capítulo 2, aborda questões relacionadas ao processamento de linguagem natural, processamento de dados, algoritmos de aprendizado de máquina, além da apresentação de alguns trabalhos relacionados. A metodologia utilizada é descrita no Capítulo 3, com detalhes sobre o banco de dados, ferramentas que foram utilizadas e experimentos que foram feitos. Os resultados obtidos com os experimentos realizados, bem como as análises desses resultados são apresentados no Capítulo 4. Uma síntese do trabalho com os principais resultados, conclusões e sugestões de continuidade dos experimentos descritos neste trabalho são apresentados no Capítulo 5. Além disso, o Apêndice A contém artigos publicados e submetidos durante o desenvolvimento deste trabalho.

2 REFERENCIAL TEÓRICO

Neste capítulo são apresentados os conceitos básicos para o entendimento do trabalho, assim como os passos necessários preliminares à análise (coleta de dados, o pré-processamento textual e o tratamento dos dados).

2.1 *E-commerce*

A palavra *Ecommerce* é a abreviação de *electronic commerce* (comércio eletrônico, em português). Com a popularização do acesso à *internet*, e os benefícios da compra *online*, como variedade de produtos, facilidade de compra e preço competitivo, tem surgido novas plataformas de compra e venda de produtos e serviços. No *marketplace*, uma empresa faz a mediação das transações de compra e venda, permitindo que vários lojistas se cadastrem, anunciem e vendam seus produtos. O consumidor pode, em uma só compra, adquirir produtos de diferentes lojistas e efetuar somente um pagamento para a empresa administradora do *marketplace*, que efetuará o pagamento para cada vendedor.

Os impactos da pandemia causada pelo novo coronavírus (COVID-19) foram sentidos em todas as áreas. Embora seja um vírus com baixa letalidade, apresenta alta taxa de transmissão (ZHU et al., 2020). Entre as medidas para contenção do avanço do vírus encontra-se o isolamento social, o que diminuiu o número de pessoas frequentando lojas, *shoppings* e supermercados para comprar itens de que necessitam.

Um estudo conduzido com 1990 pessoas nos Estados Unidos (EUA) revelou que somente 5% dos entrevistados ainda consideravam frequentar um *shopping*, dadas as restrições de deslocamento, e 9% das pessoas fizeram uma compra online pela primeira vez durante a pandemia, sendo que 62% dos entrevistados já realizavam compras online anteriormente (CONSULT, 2020). Além disso, 27% dos entrevistados pretendiam gastar mais em compras *online* de mantimentos e refeições e optar pela opção de *delivery*, tendo suas compras entregues em casa, como forma de reduzir a necessidade de sair de casa, diminuindo também as chances de contaminação pelo novo coronavírus.

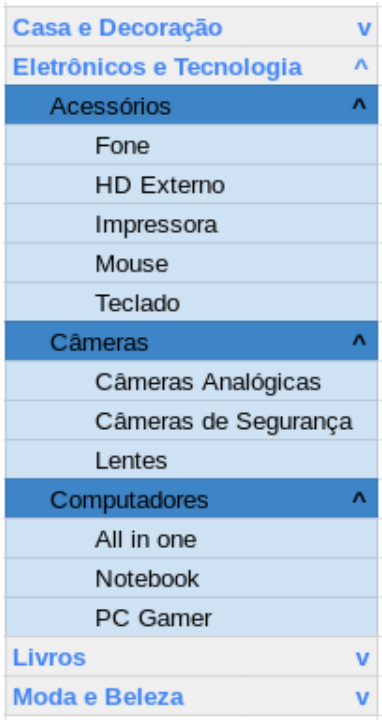
Tanto novos compradores que iniciaram suas experiências no mundo digital, quanto aqueles que já compravam *online* antes da pandemia relatam que pretendem manter compras digitais caso a experiência de compra *online* seja positiva, pois fornece uma maneira relativamente segura de fazer compras durante a pandemia (KIM, 2020). Em seu trabalho, Kim (2020) relata como a pandemia acelerou o crescimento do comércio eletrônico nos Estados Unidos,

uma vez que muitas empresas precisaram se adaptar e levar seus negócios para o meio digital, como forma de manter suas vendas.

Com a tendência das empresas disponibilizarem seus produtos para compra *online*, a demanda de ofertas tem aumentado consideravelmente. Entretanto, não há um padrão entre a descrição dos produtos fornecida pelos vendedores e os termos buscados pelos consumidores. Por isso, as experiências de compra e venda podem ser ruins, levando a resultados desagradáveis para ambas as partes.

Dentro de uma plataforma de vendas pode existir a chamada árvore de categorias ou árvore de produtos, que é uma estrutura semelhante à mostrada na Figura 2.1 para categorização dos produtos dentro da loja e permite que o usuário encontre um produto navegando pelas categorias e subcategorias existentes, sem precisar usar o campo de pesquisa de produtos. A não padronização das descrições pode levar um produto a ser colocado em uma categoria diferente daquela a que ele pertence na árvore de produtos da loja virtual. Isso é problemático pois, quando os produtos são colocados em categorias diferentes daquelas à que pertencem, o consumidor pode não encontrar o produto que procura e simplesmente desistir de comprar naquela loja virtual.

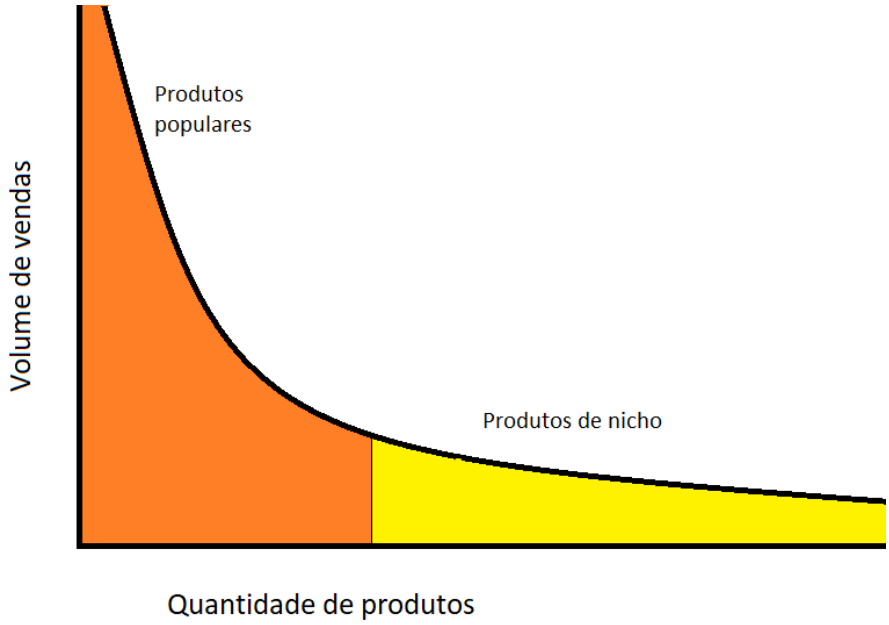
Figura 2.1 – Exemplo de árvore de produtos de uma loja virtual.



Fonte: Do Autor (2022)

Outro problema encontrado em lojas virtuais é a categorização dos itens que fazem parte do catálogo de produtos, porém com pouca representatividade na base de dados causando o chamado problema de classificação *long tail* ou problema da cauda longa (ANDERSON, 2004). A Figura 2.2 ilustra o que ocorre quando a loja dispõe de uma grande variedade de itens, porém esses itens tem pequenas quantidades. A região laranja representa os produtos de topo, que são mais populares, mas não conseguem ocupar todo o mercado. A região amarela do gráfico é onde ocorre a cauda longa, com os chamados produtos de nicho, que apesar de terem uma menor quantidade de vendas por produto, apresentam grande variedade de opções, tendo um total de vendas competitivo com os produtos de topo. Esses tipos de produto, geram dificuldades para que os algoritmos de classificação aprendam a diferenciá-los corretamente.

Figura 2.2 – Ilustração do problema *long tail* para uma loja virtual.



Fonte: Do Autor (2022)

Considerando um produto anunciado em uma loja *online*, um consumidor pode procurar por diversas características em um produto antes de efetivamente realizar a compra. O mais comum é a pesquisa pelo nome do produto, porém podem ser feitas pesquisas considerando as variações que o produto pode apresentar, como cor, tamanho, capacidade, volume, tempo de bateria, acessórios incluídos, fabricante, etc. Além disso, vários vendedores podem oferecer o mesmo produto, mas com especificações diferentes, o que pode causar mudanças no valor do produto ofertado. Outro ponto de atenção é em relação ao frete que pode ser cobrado pelo vendedor para fazer a entrega do produto no local escolhido pelo consumidor, pois diferentes

regiões podem apresentar diferentes valores de acordo com a transportadora escolhida para a entrega.

Em seu artigo, Schafer, Konstan e Riedl (2001) discute como os sistemas de recomendação ajudam os sites de comércio eletrônico a aumentar as vendas, fazendo a análise dos sistemas de recomendação em seis sites líderes de mercado. Segundo Schafer, Konstan e Riedl (2001) os sistemas de recomendação aumentam as vendas eletrônicas de três formas: convertendo visitantes em compradores, aumentando as vendas cruzadas e construindo lealdade dos compradores. Muitos compradores visitam vários *sites* antes de realmente fazer uma compra. Os sistemas de recomendação podem ajudar o comprador a encontrar o produto que ele realmente quer comprar. Além disso, tais sistemas aumentam as vendas cruzadas ao sugerir novos produtos baseados em um produto que o comprador esteja olhando no momento. E a construção de lealdade do comprador é feita pela sugestão de produtos úteis, considerando itens que já estão selecionados para compra e criando interfaces mais agradáveis, aprendendo sobre as necessidades do consumidor e criando um relacionamento entre o consumidor e o vendedor (ou loja) que faça com que ele queira repetir o processo de compra futuramente.

O grande volume de dados gerado no comércio eletrônico requer a aplicação de ferramentas de análises de *big data* e as diversas tecnologias que possibilitam a análise de dados do consumidor. Outra ferramenta que pode ser usada para análise de dados do comércio eletrônico é a análise preditiva, que é o uso de dados, obtidos anteriormente, para prever tendências futuras. No trabalho de (EDOSIO, 2014) é apresentado um estudo de caso de empresas como *Amazon* e *Walmart* que utilizam plataformas de *E-commerce* aliadas à análise de dados e como isso aumenta a vantagem competitiva.

Outras estratégias das plataformas de compra para atrair consumidores podem envolver melhorias relacionadas com a apresentação dos produtos na plataforma, a interface dos *sites* e o suporte oferecido aos clientes. Com o aumento das ofertas de produtos em plataformas de compra *online*, o aprendizado de máquina está sendo aplicado para realizar o entendimento semântico de textos para construir perfis de usuário e descobrir suas preferências, na expectativa de melhorar a recomendação de produtos e gerar uma melhor experiência de compra (CHEN; WANG, 2013).

2.2 Processamento de Linguagem Natural

A área de Inteligência Artificial conhecida como Processamento de Linguagem Natural (PLN) estuda a capacidade e as limitações de uma máquina em entender a linguagem dos seres humanos (ZONG; HONG, 2018). Com os estudos de PLN, as máquinas são capazes de ler, interpretar e entender significados contidos nas linguagens humanas. O volume de dados gerado diariamente é enorme e é possível encontrar tais dados em diversos formatos (texto, áudio, vídeo, imagens, etc).

A linguagem natural é a forma mais comum de comunicação, que pode ser escrita ou falada, e possui regras, nuances e ambiguidades. Tais regras variam de acordo com o idioma e representa um desafio no desenvolvimento dos sistemas digitais, que precisam ter a capacidade de entender tal linguagem e suas características. O processamento de linguagem natural também deve levar em conta os seguintes aspectos da linguagem natural: o som, a estrutura e o significado.

O som é a parte ligada à fonologia, que é a área da linguística que estuda as características sonoras de um idioma. Aqui são estudadas a função que os sons tem em cada língua, e como os fonemas se organizam e sua contribuição na formação do significado de uma palavra ou frase.

A estrutura está relacionada com a morfologia e sintaxe da língua. A morfologia é o estudo da formação, estrutura e classificação das palavras em uma das classes gramaticais (substantivo, artigo, adjetivo, numeral, pronome, verbo, advérbio, preposição, conjunção e interjeição). A sintaxe é o estudo da função de cada palavra dentro da frase, das frases nas orações e nas sentenças. Também estuda a relação entre cada palavra ou entre cada frase.

Já o significado é relacionado com a semântica e pragmática. A semântica está ligada ao sentido das palavras e a interpretação das sentenças e enunciados. A pragmática está ligada aos objetivos da comunicação e analisa a palavra ou frase de acordo com o contexto e as diferentes interpretações que podem surgir com a variação do contexto.

Considerando todas essas características que influenciam no significado de uma mensagem, um grande desafio é extrair informação relevante de dados não estruturados. A Figura 2.3 mostra um exemplo de dado não estruturado e a estruturação desse dado de acordo com as categorias relevantes. Neste tipo de dado, as informações são apresentadas de maneira dispersa, os dados relevantes para a análise não são categorizados e não se encaixam na estrutura tradicional de linhas e colunas dos bancos de dados relacionais.

Um formato bem comum de uma base de dados textual não estruturada é aquele em que cada amostra de texto é representada em uma linha, sem nenhum tipo de organização ou separação de informações de acordo com sua relevância. Para extração de informação desses dados é necessário realizar um pré-processamento, para que os dados sejam colocados em um formato de fácil entendimento pela ferramenta de análise de texto utilizada.

Figura 2.3 – Exemplo de transformação de um texto não estruturado em texto estruturado.

Dados não estruturados	**Dados estruturados**	
Geladeira inverter frost free Panasonic NR-BT55PV2 inox com freezer 483L 127V. Possui freezer superior. Eficiência energética A. Conta com iluminação interior. Dimensões: 695 mm de largura, 1900 mm de altura e 758 mm de profundidade. Possui porta-latas e porta-ovos	**Atributo**	**Valor**
	Produto	Geladeira
	Fabricante	Panasonic
	Modelo	NR-BT55PV2
	Capacidade	483 L
	Tensão	127 V
	Cor	Inox

Fonte: Do Autor (2022)

A mineração de texto (*text mining*, em inglês) é uma área da mineração de dados em que são desenvolvidos técnicas e processos para extração de informação e conhecimento de dados textuais não estruturados. Um problema encontrado na análise textual é lidar com vocabulários muito grandes, além da compreensão e manipulação de linguagens morfologicamente ricas. No seu artigo, Ling et al. (2015) propuseram uma modificação na arquitetura de tradução neural do codificador/decodificador padrão para usar combinações de vocabulário ilimitado baseado em caracteres.

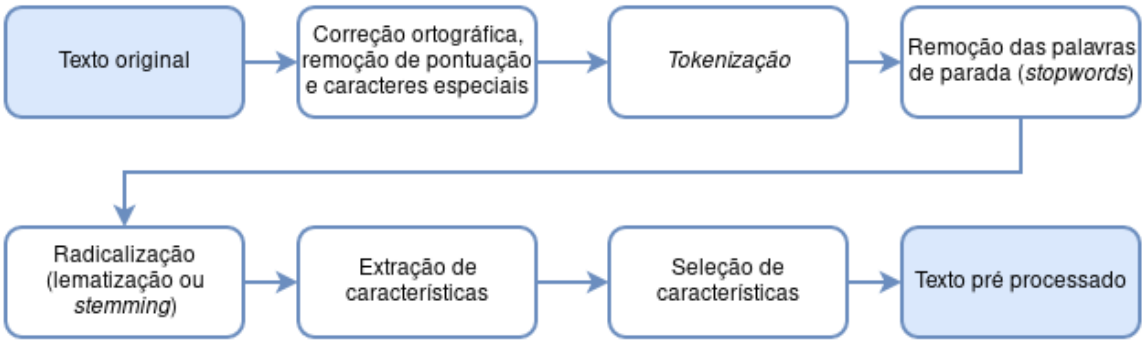
O aprendizado de máquina (*machine learning*) pode ser utilizado para resolver problemas de mineração de dados, como no trabalho de Matsumoto et al. (2017), onde é proposto um *framework* que trata tanto informações numéricas quanto textuais. Com o crescente uso das mídias sociais na Internet, a análise textual e mineração de opiniões se tornou uma abordagem essencial para analisar tantos dados, sendo utilizada em aplicações como precificação de produtos, previsão de mercado, previsão de eleições, inteligência competitiva, entre outras (SUN; LUO; CHEN, 2016).

2.3 Pré-processamento

O pré processamento é uma parte importante na análise de dados textuais, pois os dados brutos contém muita informação que não é aproveitada pelo algoritmo de *machine learning*

utilizado. A Figura 2.4 mostra as etapas do pré processamento necessárias para transformar um texto não estruturado, como o da Figura 2.3, em um tipo de dado próprio para o modelo.

Figura 2.4 – Etapas do pré processamento do texto não estruturado.



Fonte: Do Autor (2022)

A primeira etapa do pré processamento é a correção ortográfica, e é feita para eliminar erros de digitação ou de ortografia contidos no texto. Essa verificação é feita usando dicionários da língua analisada junto com um corretor ortográfico. Também pode ser realizada a remoção de caracteres especiais e sinais de pontuação que podem não ser entendidos pela ferramenta escolhida. Com isso, evita-se a perda de palavras-chave, pois as palavras que estavam escritas erradas, ou com símbolos especiais são corrigidas e podem ser usadas como informação para o algoritmo que irá analisar o texto.

A próxima etapa do pré processamento é a *tokenização*, que é a divisão de cada amostra em *tokens* para o processo de análise e mineração de texto. A formação dos *tokens* é feita quebrando o texto em frases ou palavras, dependendo do limite estabelecido para essa separação. Esse limite é escolhido de acordo com as características e o tamanho do texto a ser analisado. O limite pode ser um espaço, no caso de palavras, ou um sinal de pontuação, caso o *token* seja uma frase. Pode ser feito, ao fim da *tokenização*, a técnica de *lower case*, que consiste em colocar todas as letras da palavra ou frase em letra minúscula.

A etapa seguinte do pré processamento é a remoção das palavras de parada (*stopwords*, em inglês), e é feita removendo palavras desnecessárias para a análise, como artigos e preposições. Ainda podem ser removidas palavras usadas para representar conceitos generalizados que não são interessantes para a análise, assim como palavras de tom ofensivo. É possível encontrar listas prontas com essas palavras, como as listas utilizadas pelo *Google, Facebook, YouTube*, ou construir uma lista personalizada com as palavras de acordo com a aplicação.

A próxima etapa é a radicalização dos termos, e é feita reduzindo cada palavra ao seu radical, com a remoção dos prefixos, sufixos e outros modificadores da palavra. Isso é necessário para aumentar a chance dessa palavra ser encontrada no dicionário nas etapas de análise do texto. A radicalização pode ser feita usando a lematização ou o *stemming*. Na lematização, as palavras são transformadas no seu radical, sendo o infinitivo dos verbos e o masculino singular dos substantivos e adjetivos. No *stemming*, os prefixos e sufixos de uma palavra são removidos, reduzindo a palavra à sua raiz, que não precisa ser um radical válido gramaticalmente. A radicalização diminui a quantidade de palavras que serão analisadas nas próximas etapas.

A etapa seguinte do pré processamento é a extração de características, que é feita para diminuir a dimensionalidade dos dados, fazendo com que o texto analisado seja reduzido a algumas características que representem todo seu sentido. Existem diversas ferramentas que podem ser utilizadas para extração de características. Cada uma delas tem suas vantagens e desvantagens e pode apresentar melhor desempenho dependendo do problema no qual é aplicada e do formato do resultado esperado. Para a extração de características de dados em formato textual podem ser usadas diversas ferramentas, como *word2vec* (AL-AMIN; ISLAM; UZZAL, 2017), (TIAN; LI; LI, 2018), *doc2vec* (KARVELIS et al., 2018), (BILGIN; SENTURK, 2017) e *seq2seq* (WU et al., 2016), sendo que a escolha depende das características da base de dados escolhida e do resultado esperado.

As ferramentas de extração de características permitem que o texto seja transformado em um vetor que contenha a informação do texto original, porém de maneira simplificada. A etapa de seleção de características pode ser realizada em sequência para selecionar, dentre aquelas características extraídas, aquelas que melhor representam a amostra utilizada. As características selecionadas podem, então, ser utilizadas em algoritmos de aprendizagem, classificação, seleção, etc.

2.4 Métodos para *Word Embeddings*

O termo *Word Embeeding* pode ser traduzido livremente como representação de palavras e propõe que palavras com significado semelhante tem uma representação semelhante e serão representadas em espaços próximos. Existem algumas métricas, utilizadas no PLN, para medida de similaridade entre palavras e obtenção de informações que são utilizadas para análises de textos, como a Distância de *Levensthein* e a *Bag-of-words* (BOW).

A Distância de *Levensthein*, também conhecida como distância de edição, é uma técnica utilizada em aplicações que precisam determinar quão semelhantes duas sequências de caracteres são (LEVENSHTEIN, 1966). Essa distância é dada pelo número mínimo de operações necessárias para transformar uma sequência na outra. As operações podem ser a inserção, remoção ou substituição de um carácter.

Já a métrica *Bag-of-words* é usada para identificar as palavras mais frequentes em um documento e também para extração de características para classificação de documentos. O modelo BOW contém um vocabulário de palavras conhecidas e uma medida da presença dessas palavras conhecidas no texto em que se está analisando. Não são coletadas informações sobre posicionamento ou estrutura das palavras, somente a quantidade de vezes em que a palavra aparece no texto. O modelo BOW trabalha com uma pontuação da frequência de palavras, o que pode gerar um problema pois palavras muito frequentes podem não conter conteúdo relevante para a análise do texto, e palavras que aparecem com menos frequência, mas são específicas do tema do texto podem fornecer informação mais acertada.

Uma abordagem que resolve esse problema do BOW é a *Term Frequency – Inverse Document Frequency* (TF-IDF), que permite calcular a importância de uma palavra. A métrica TF-IDF é obtida pela multiplicação de dois valores, o Frequência do Termo (TF) e o Inverso da Frequência no Documento (IDF). O valor TF mede a frequência com que um termo ocorre num documento. O valor IDF mede o quão importante um termo é no contexto de todos os documentos que estão sendo analisados.

As representações geradas pelos métodos de *word embeedings* são vetores densos, nos quais as palavras são representadas por números reais em um espaço dimensional de tamanho pré-definido e o conjunto de características extraídas do texto é representado por um vetor neste espaço (MIKOLOV et al., 2013). Alguns dos métodos mais utilizados para o processamento de linguagem natural e a extração de características são o *word2vec*, o *seq2seq* e o *Glove*.

2.4.1 Método *word2vec*

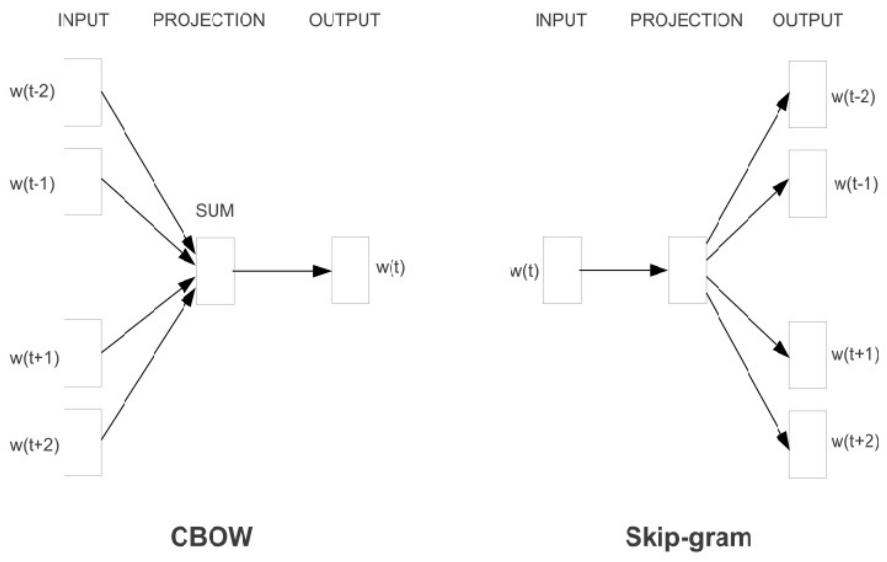
A abordagem utilizada no método *word2vec* foi proposta por uma equipe de pesquisadores da *Google* (MIKOLOV et al., 2013a), e transforma cada palavra do conjunto de frases em um vetor numérico que a represente semanticamente. Além disso, esse método permite capturar relações entre as palavras de uma frase e definir a similaridade entre textos.

Nesse método, os autores utilizaram *word embeeding*, que é um conjunto de técnicas de aprendizado no processamento de linguagem natural, nas quais as palavras ou frases são mapeadas em vetores não lineares de múltiplas dimensões. Esses vetores de números reais carregam informações sobre as palavras e, ao mesmo tempo, possibilitam cálculos matemáticos como o cálculo da distância entre vetores. Esse cálculo é importante pois uma proximidade entre os vetores pode indicar também uma proximidade linguística entre as palavras.

A abordagem do *Word2Vec* é treinar uma rede neural e obter os pesos da camada oculta para representar os vetores de palavras (MIKOLOV et al., 2013a). Na arquitetura do modelo *Word2Vec* são utilizados dois tipos de rede neural, o *Continuous Bag-of-Words* (CBOW) e o *Continuous Skip-Gram* (MIKOLOV et al., 2013a), cada um contendo uma camada de entrada de dados, uma camada oculta e uma camada de saída.

O modelo CBOW é utilizado para descobrir a palavra central de uma sentença, baseado nas palavras que a cercam. A camada de entrada recebe os dados pré-processados e a saída será uma palavra que representa a maior probabilidade para a palavra alvo. Já o *Skip-Gram* é utilizado para descobrir as palavras mais prováveis para a vizinhança a partir da palavra central. Na camada de entrada, tem-se apenas a palavra alvo e na saída tem-se diferentes opções de palavras de contexto possíveis, com uma probabilidade associada. As duas redes são usadas sequencialmente, como mostrado na Figura 2.5.

Figura 2.5 – Arquitetura do *word2vec*: CBOW e *Skip-gram*



Fonte: (MIKOLOV et al., 2013a)

Considerando uma janela de tamanho 2, significa que só serão utilizadas as 2 palavras antes e depois da palavra alvo, como entradas para a rede CBOW. A rede projeta as palavras em um espaço multidimensional e faz a previsão da palavra central (do instante *t*), independente da ordem das palavras recebidas. Já a rede *Skip-gram*, faz a operação contrária, prevendo as palavras de contexto. O *Skip-gram* tenta maximizar a classificação de uma palavra com base em outra palavra na mesma frase, aprendendo representações vetoriais de palavras a partir de grandes quantidades de dados de texto não estruturados (MIKOLOV et al., 2013a).

2.4.2 Método sequência a sequência (*Seq2seq*)

Já o método *Seq2Seq* foi introduzido pela *Google* em aplicações de tradução, com o Sistema de Tradução Automática Neural do *Google* (GNMT, do inglês *Google Neural Machine Translation system*), que utiliza técnicas de treinamento de ponta para obter melhores resultados em termos de qualidade de tradução automática (WU et al., 2016). Nesse método, uma sequência de *tokens* de entrada, em um domínio específico, é transformada em uma sequência de *tokens* de saída em outro domínio (SUTSKEVER; VINYALS; LE, 2014).

O método *Seq2Seq* foi proposto no trabalho de Sutskever, Vinyals e Le (2014) em que os autores demonstram o uso de redes *Long short-term memory* (LSTM) para aplicações de tradução de máquina, como a tradução "de ponta a ponta". A técnica de aprendizado sequência a sequência usa um modelo que converte sequências de um domínio para sequências em outro domínio. Esse é o método utilizado pelo tradutor automático do *Google* [1] , que é capaz de traduzir instantaneamente palavras, frases ou páginas da Web para mais de 100 idiomas.

No artigo de Sutskever, Vinyals e Le (2014), por exemplo, é feita a tradução de frases do inglês para o francês. O artigo propõe o uso de duas redes profundas com LSTM, com um modelo Codificador/Decodificador. A primeira rede LSTM é o codificador e é responsável por mapear a entrada em um vetor de dimensão fixa. A segunda rede LSTM é o decodificador e é responsável por mapear o vetor fixo para uma sequência de saída.

Para um melhor desempenho, o modelo inverte a sequência de entrada ao mapear para a sequência de saída, o que facilita estabelecer a relação entre entrada e saída. Isso ocorre pela minimização da distância entre as palavras geradas e as palavras fonte, o que introduz dependências de curto prazo e torna o problema de otimização mais simples. Essa técnica

[1] *https://translate.google.com.br/*

otimiza os resultados não só nas sentenças curtas, mas gera um bom resultado também em sentenças mais longas, ao contrário de outros métodos.

O modelo *Seq2Seq* utiliza as LSTMs para aprender a mapear uma sentença de entrada de comprimento variável em uma representação vetorial de dimensão fixa. Juntamente com a inversão da ordem de entrada, essa técnica ajuda a LSTM decodificadora a encontrar representações de frases que capturem seu significado, já que frases com significados semelhantes estão próximas umas das outras, enquanto significados de frases diferentes estão distantes (SUTSKEVER; VINYALS; LE, 2014).

2.4.3 Método Global Vectors (GloVe)

O algoritmo Global Vectors (GloVe) gera representações vetoriais para palavras usando aprendizagem não supervisionada (PENNINGTON; SOCHER; MANNING, 2014). Neste algoritmo, as estatísticas globais do conjunto de textos são capturadas diretamente pelo modelo e o treinamento é realizado usando um modelo específico de mínimos quadrados ponderados com as estatísticas globais e a matriz de co-ocorrência de palavras, que indica a frequência com que as palavras do *corpus* são correlacionadas entre si, dentro de um contexto.

O GloVe usa a fatoração de matriz global (*global matrix factorization*) e a janela de contexto local (*local context window*). Os métodos de fatoração de matriz são usados para gerar representações de palavras de baixa dimensão, uma vez que permite decompor matrizes grandes em matrizes menores. Já a janela de contexto local aprende a representação das palavras usando palavras próximas.

O GloVe é um modelo log-bilinear que usa pesos com mínimos quadrados ponderados. Durante o treinamento o algoritmo aprende vetores de palavras, de forma que o produto escalar seja igual ao logaritmo da probabilidade de co-ocorrência das palavras (PENNINGTON; SOCHER; MANNING, 2014). Como o logaritmo de uma razão é igual à diferença entre os logaritmos, pode-se associar as razões de probabilidades de co-ocorrência com as diferenças vetoriais da representação das palavras no espaço vetorial pré definido. Já que os valores de probabilidades de co-ocorrência palavra-palavra podem conter algum significado, esses significados também são codificadas como diferenças de vetor. O algoritmo produz vetores de palavras que possuem faixas horizontais marcadas que ficam mais evidentes à medida que a frequência da palavra aumenta.

2.5 Algoritmos e conceitos de aprendizado de máquina

O aprendizado de máquina ou *machine learning*, em inglês, é uma área da ciência da computação que estuda reconhecimento de padrões e aprendizado computacional em inteligência artificial. Segundo Mohri, Rostamizadeh e Talwalkar (2012), o aprendizado de máquina pode ser definido como métodos computacionais que, dada uma tarefa, aprendem, por meio da experiência (dados obtidos no passado e fornecidos ao modelo computacional), a melhorar sua performance e fazer previsões mais certeiras.

Os algoritmos de aprendizado supervisionado aprendem usando dados rotulados para o treinamento, ou seja, todas as amostras utilizadas possuem uma classe associada à elas. Na tarefa de classificação de um modelo de treinamento supervisionado, o modelo faz a divisão das amostras nas classes e seu desempenho é medido comparando o valor conhecido (valor real das amostras) com o valor predito pelo modelo. Já nos algoritmos de aprendizado não supervisionado, o conjunto de dados usado no treinamento não é rotulado, ou seja, as classes não são conhecidas e o algoritmo agrupa as amostras em grupos (*clusters*), usando as características semelhantes que aprendeu a identificar nos dados para colocá-los nos *clusters* corretos. Alguns algoritmos permitem definir o número de *clusters* desejado e o algoritmo então agrupa os dados da melhor maneira de acordo com o número de grupos definido.

A aprendizagem profunda, do inglês *deep learning*, é um ramo de aprendizado de máquina que usa algoritmos de redes neurais que podem ter várias camadas, trazendo a ideia de profundidade da rede, cada camada com inúmeros neurônios, sendo que cada neurônio da rede atribui um peso para os dados que entram, determinando o quão correto ou incorreto ele é, de acordo com a tarefa que está sendo executada e a saída final da rede é determinada pelo total desses pesos (LECUN; BENGIO; HINTON, 2015). O *deep learning* usa algoritmos de aprendizado de máquina como *Support Vector Machine* (SVM), *K-Nearest Neighbor* (KNN) e Rede Neural Artificial (RNA) para aprender usando os dados recebidos e fazer uma classificação ou predição, de acordo com a forma que o algoritmo foi programado. O uso de redes de aprendizado profundo tem sido aplicado em diversas áreas como reconhecimento de fala, visão computacional e processamento de linguagem natural.

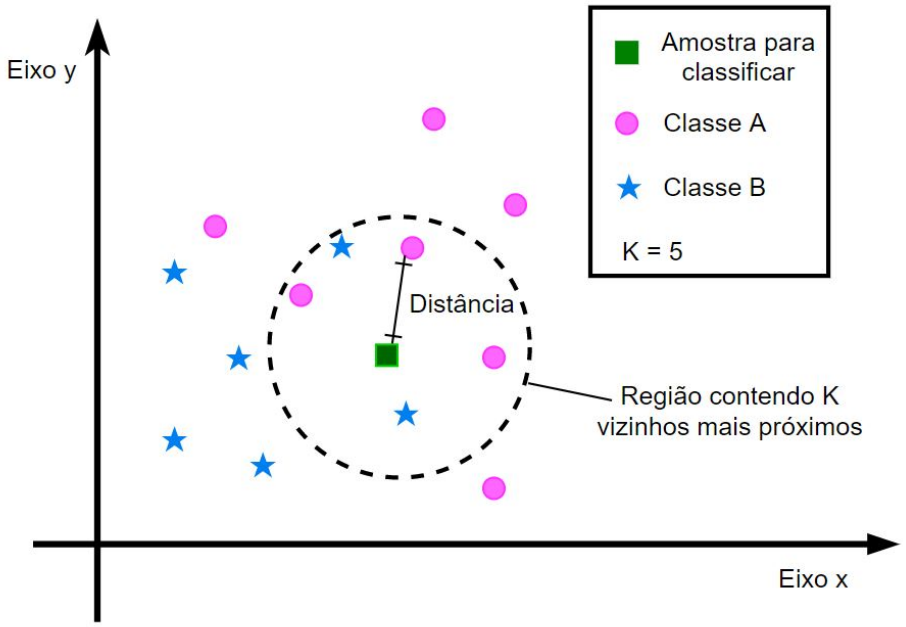
2.5.1 *K-Nearest Neighbors*

O algoritmo *K-Nearest Neighbors* (KNN) ou K-Vizinhos Mais Próximos, foi proposto no trabalho de Fukunage e Narendra (1975) e é um algoritmo de aprendizado supervisionado

que pode resolver problemas de classificação e regressão. O objetivo do algoritmo KNN é determinar a classe de uma amostra baseado nas amostras vizinhas de acordo com um conjunto de treinamento.

Os parâmetros que devem ser determinados são a métrica de distância e o valor de k utilizados pelo classificador. O valor de k é o número de amostras vizinhas que será considerado para a classificação e sua escolha varia de acordo com a base de dados. O valor da distância é a distância entre a amostra que deve ser classificada e as amostras vizinhas. A Figura 2.6 ilustra os conceitos de distância e do número de vizinhos aplicados no KNN. Neste exemplo, o quadrado verde representa a amostra a ser classificada nas classes representadas pelo círculo rosa ou estrela azul, o número de vizinhos (K) é igual a 5 e o círculo pontilhado indica a região contendo os K vizinhos escolhidos para classificação da amostra.

Figura 2.6 – Conceitos do classificador KNN



Fonte: Do Autor (2022)

O algoritmo KNN pressupõe que objetos semelhantes estão próximos uns dos outros. Para cada nova amostra, é feita a comparação com uma amostra já existente, usando uma medida de distância. Para $K=1$, a classe da nova amostra será determinada de acordo com a classe da amostra que tiver a menor distância até ela. Para $K>1$ a classe mais frequente nos K vizinhos será a classe da amostra que se deseja classificar.

A medida de distância pode ser uma forma de medir a similaridade entre as amostras analisadas. O cálculo da distância pode ser feito com diversas fórmulas, como a distância ponto a ponto, distância Euclidiana ou a distância Manhattan.

Considerando os pontos n-dimensionais $P = (p_1, p_2, ..., p_n)$ e $Q = (q_1, q_2, ..., q_n)$, a distância ponto a ponto é dada pela Equação 2.1 para P e Q em suas respectivas dimensões, tendo como resultado um vetor com *n* dimensões, tal como as dimensões dos pontos.

$$\text{Distância ponto a ponto (P, Q)} = ((p_1 - q_1),\ (p_2 - q_2),\ ...,\ (p_n - q_n)) \tag{2.1}$$

De forma semelhante, para os pontos n-dimensionais $P = (p_1, p_2, ..., p_n)$ e $Q = (q_1, q_2, ..., q_n)$, a distância Euclidiana é dada pela Equação 2.2 para P e Q em suas respectivas dimensões.

$$\text{Distância Euclidiana (P, Q)} = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + ... + (p_n - q_n)^2} \tag{2.2}$$

Já a distância Manhattan é dada pela Equação 2.3, considerando os mesmos pontos n-dimensionais $P = (p_1, p_2, ..., p_n)$ e $Q = (q_1, q_2, ..., q_n)$.

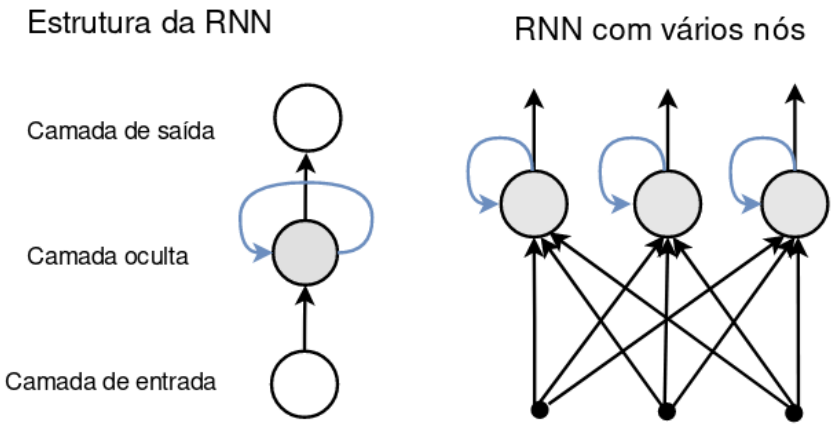
$$\text{Distância Manhattan (P, Q)} = |p_1 - q_1| + |p_2 - q_2| + ... + |p_n - q_n| \tag{2.3}$$

2.5.2 Rede Neural Recorrente

A Rede Neural Recorrente (RNN, do inglês *Recorrent Neural Networks*) é baseada no trabalho de Rumelhart, Hinton e Williams (1986) e é uma classe de rede neural usada para reconhecer padrões e fazer processamento de dados sequenciais. Esse tipo de rede neural usa informação sequencial e apresenta conexão entre dois nós de modo que formem um ciclo dentro de uma rede, passando informação do estado atual de um nó para o outro, como mostrado na Figura 2.7. As camadas da RNN implementam um loop para iterar os estados de uma sequência e mantém uma memória que codifica informações sobre os estados já conhecidos pela rede. Ou seja, esse tipo de rede possui estados ocultos que funcionam como uma memória, armazenando as saídas anteriores, que serão utilizadas juntamente com novas entradas para determinar sua nova saída.

As RNNs podem usar sua memória para processar uma sequência de entradas. Ela executa a mesma operação para todas as entradas de dados, mas a saída da entrada atual vai de-

Figura 2.7 – Neurônios em um Rede Neural Recorrente



Fonte: Do Autor (2022)

pender também do valor de saída anterior. Cada saída é copiada e enviada de volta como entrada da rede (essa operação é representada pela seta azul na Figura 2.7). Para tomar uma decisão e determinar a nova saída, a rede considera a entrada atual e a saída que aprendeu da entrada anterior. Essa saída é o estado oculto da rede e possui informações sobre as representações dos estados anteriores.

As RNNs são muito utilizadas em tarefas como reconhecimento de fala, transcrição de fala em texto, tradução automática e modelos de linguagem (MIKOLOV et al., 2013b; SUTSKEVER; VINYALS; LE, 2014; GRAVES; JAITLY, 2014; MIKOLOV et al., 2013a). Nessas aplicações, a rede lê a sequência de entrada por algum tempo e depois começa a gerar a sequência de saída. No entanto, as RNNs podem apresentar um desempenho ruim em aplicações que dependam do aprendizado de dependências de longo alcance, como a relação entre as palavras que estão distantes em uma frase, devido ao problema de dissipação do gradiente.

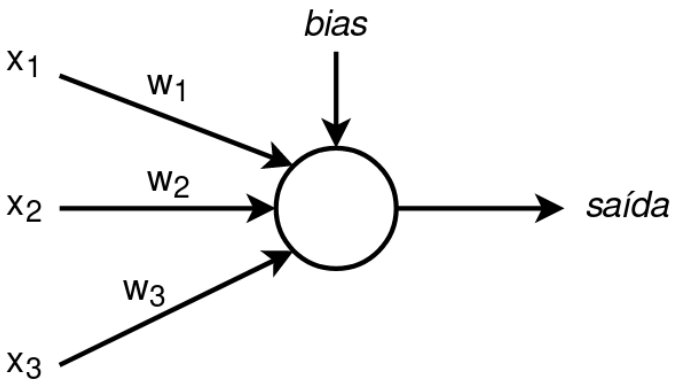
Simplificando, o gradiente é o valor usado para atualizar os pesos de uma rede neural. O problema de dissipação do gradiente ocorre quando o valor do gradiente se torna muito pequeno, não contribuindo para o aprendizado da rede e desaparecendo ao longo do tempo.

2.5.3 *Perceptron* multi camadas

A rede *Perceptron* multi-camadas (MLP, do inglês *multilayer Perceptron*) com uma camada de entrada, uma camada de saída, e pode ter inúmeras camadas intermediárias (camadas ocultas), cada camada contendo um ou mais neurônios (ROSENBLATT, 1958). Um *Perceptron* é um modelo matemático que pode receber várias entradas (x_i), como mostrado na Figura 2.8, e produzir uma saída binária. Pode-se usar um único neurônio recebendo as entradas e produ-

zindo uma saída, assim como também é possível conectar vários neurônios em várias camadas e formar uma rede de neurônios.

Figura 2.8 – Representação de uma rede neural *Perceptron*



Fonte: Do Autor (2022)

Para determinar a saída de cada neurônio (Equação 2.4) são utilizados as entradas (x_i), os pesos (w_i), que são números reais que quantificam a importância de cada entrada para a saída daquele neurônio, e o viés (*bias*) do neurônio. O viés pode ser entendido como um valor que representa quão fácil é para o *Perceptron* produzir o valor 1 na sua saída.

$$saída = \begin{cases} 0, \ se \ w \cdot x + b <= 0 \\ 1, \ se \ w \cdot x + b > 0 \end{cases} \tag{2.4}$$

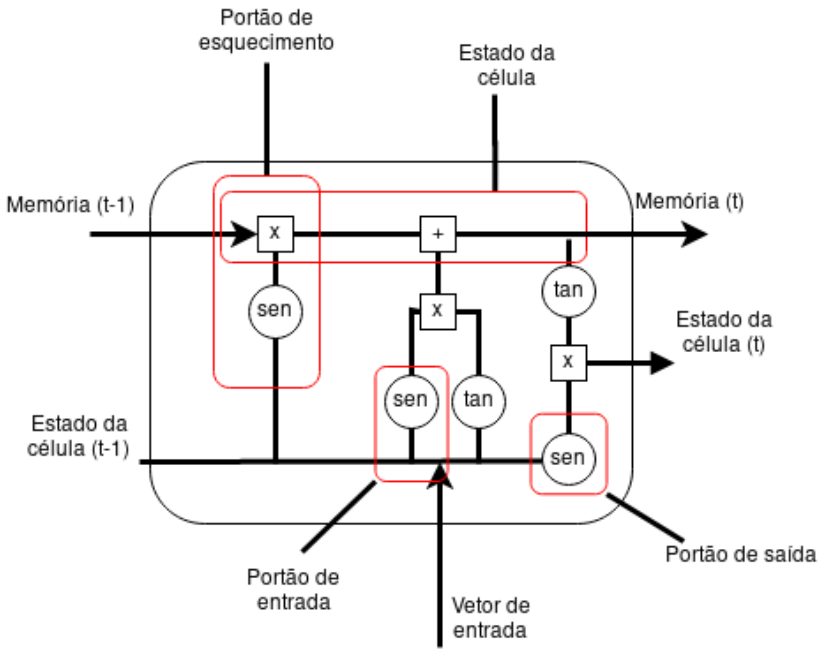
O *Perceptron* é uma rede neural de camada única e, quando são construídas várias camadas de *Perceptron*, tem-se uma Rede Neural Artificial (RNA). Em uma rede *Perceptron*, cada neurônio das camadas intermediárias toma uma decisão analisando os resultados da camada anterior, formando um processo de tomada de decisão complexo. A rede MLP é do tipo *feed forward*, ou seja, as conexões entre os nós da rede não formam um ciclo pois as entradas são recebidas pelo neurônio, que faz o processamento e gera uma saída. O treinamento da rede é feito usando a técnica de aprendizado supervisionado chamada *backpropagation*, que atualiza os pesos aplicados às entradas de cada neurônio, com o objetivo de fazer o modelo aprender os valores ideais de pesos e *bias* para que o erro entre a saída do sistema e a resposta desejada seja minimizado. Depois que o treinamento é concluído, pode-se apresentar novos dados de entrada para o modelo e ele irá prever uma saída.

2.5.4 Redes *Long Short-term Memory*

A rede de memória de longo prazo (LSTM, do inglês *Long Short-term Memory*) é um tipo de arquitetura da rede RNN usada para *deep learning*. A LSTM foi proposta no trabalho de Hochreiter e Schmidhuber (1997) e funciona de maneira semelhante à RNN, porém trata o estado oculto de forma diferente. Essa arquitetura foi construída com o intuito de resolver o problema da dissipação de gradiente, que faz com que a RNN, em sua arquitetura padrão, não consiga aprender de maneira eficiente as dependências de longo prazo (HOCHREITER; SCHMIDHUBER, 1997).

A estrutura da LSTM é bi-direcional, ou seja, ela considera os estados antes e depois do estado atual. A Figura 2.9 representa a estrutura interna de uma LSTM, mostrando o posicionamento dos portões e células. Os círculos com as letras sen e tan representam as funções seno e tangente.

Figura 2.9 – Estrutura interna de uma LSTM



Fonte: Do Autor (2022)

Tal estrutura também permite que ela lembre as relações e consiga prever sequências após 1000 intervalos, mesmo com a presença de ruído. A estrutura é organizada em cadeia, com redes neurais, diferentes blocos de memória (células) e portões (*gates*) (HOCHREITER; SCHMIDHUBER, 1997). A informação é armazenada pelas células e os portões controlam o fluxo de informação, determinando qual informação será passada para a célula.

Os portões são compostos de uma camada de rede neural sigmoide e um operador de multiplicação por pontos. A função sigmoide é usada para determinar o peso de cada informação e o quanto ela é importante para a rede. Ela gera números entre zero e um, descrevendo quanto de cada componente deve ser liberado, sendo que quanto mais próximo de zero, menos importante é essa informação para a rede, e menos dela será passada para a célula.

Ainda segundo o artigo de Hochreiter e Schmidhuber (1997), o portão de esquecimento (*forget gate*) é usado para determinar quais partes da informação são importantes e remover as informações que não são mais úteis no estado da célula. O portão de entrada (*input gate*) é usado para adicionar novas informações no estado da célula. Essas informações são uma combinação das informações da memória de curto prazo com as novas informações recebidas naquele instante de tempo. Depois que o estado da célula é atualizado, o portão de saída (*output gate*) é usado para extrair informações úteis do estado da célula atual, utilizando a memória de curto prazo, e apresentá-las como uma saída da rede.

2.5.5 *Few/One-shot learning*

O treinamento de classificadores, normalmente, é feito com muitas amostras das classes que se deseja classificar, para que o classificador aprenda bem o modelo que diferencia as classes e alcance bons resultados, com seu desempenho avaliado de acordo com diferentes métricas. Porém, existem problemas em que algumas classes possuem poucas amostras, devido à dificuldade de conseguir amostras para todas as classes, ou ao fato de existirem muitas classes para o problema em questão e nem todas possuírem o mesmo grau de representatividade na base de dados. Entretanto, independente do número de amostras de cada classe, deseja-se que essas amostras sejam classificadas corretamente. O processo de aprendizagem usando poucas amostras das classes envolvidas no problema é uma tarefa complexa tanto para as redes neurais tradicionais quanto para as redes neurais profundas (JADON; GARG, 2020).

Em casos em que se tem poucas amostras rotuladas de uma ou mais classes para treinar o modelo, é conhecido como problema *few-shot learning* e, no caso em que só existe uma amostra rotulada da classe, é chamado de problema *one-shot learning*. Nesta abordagem, o modelo desenvolve uma capacidade de generalização pelo aprendizado de informações sobre as classes a partir de uma ou poucas amostras, o que torna interessante sua aplicação em problemas de diversas áreas do conhecimento.

Segundo Jadon e Garg (2020), um algoritmo de *few/one-shot learning* requer que o modelo tenha filtros treinados e uma arquitetura pré-definida, uma base de dados bem balanceada e com amostras das diferentes classes rotuladas corretamente, e por fim alguma forma de analisar e classificar as informações coletadas e armazenadas pelo modelo. Durante o treinamento, uma base de dados não balanceada, com muitas amostras de uma única classe pode causar o *overfitting* da rede neural. Uma das estratégias que podem ser usadas para balancear a base de dados é o aumento de dados (*data augmentation*, em inglês), que adiciona variações aos dados, na forma de algum tipo de ruído e tem como resultado o aumento dos dados disponíveis e o balanceamento da base de dados.

Existem diferentes algoritmos que podem ser aplicados para o problema *few/one-shot learning*. O algoritmo KNN pode ser utilizado para pequenos conjuntos de dados e ter um resultado satisfatório, pois seu desempenho depende da métrica de distância escolhida (JADON; GARG, 2020). Entretanto, à medida em que o volume de dados e o número aumentam, o custo computacional para o cálculo da distância se torna elevado e seu desempenho pode ser afetado, tornando interessante o uso de outros algoritmos. São encontrados alguns trabalhos com a aplicação de redes siamesas para *one-shot learning*, já que elas podem ser aplicadas em problemas de classificação com poucas amostras para treinamento e obter resultados satisfatórios, em problemas de diversas áreas (SOUZA et al., 2019; KOCH; ZEMEL; SALAKHUTDINOV, 2015).

Assim, considera-se o potencial dessas redes para solução do problema descrito neste trabalho. A aplicação das redes siamesas com *one-shot learning* será feita para classificação de novas amostras de produtos que tenham poucas amostras conhecidas e não são conhecidas pelo classificador em operação, uma vez que retreinar o classificador já existente para incluir essas novas classes é um processo muito custoso e são buscadas alternativas para a classificação dessas novas classes.

2.5.6 Redes neurais siamesas

A Rede Neural Siamesa pode ser usada para medir a similaridade entre duas entradas, e pode ser usada para determinar se um par de amostras pertence a uma mesma classe ou não. Essa rede foi apresentada no trabalho de Bromley et al. (1994), fazendo a comparação de assinaturas escritas em um *tablet*. Existem diversos relatos de estudos envolvendo aplicações utilizando redes siamesas para determinar a similaridade entre duas entradas, principalmente na

análise de imagens, porém sua aplicação não é restrita a esse tipo de entrada (BENAJIBA et al., 2019; BROMLEY et al., 1994; KOCH; ZEMEL; SALAKHUTDINOV, 2015; MUELLER; THYAGARAJAN, 2016; SOUZA et al., 2019).

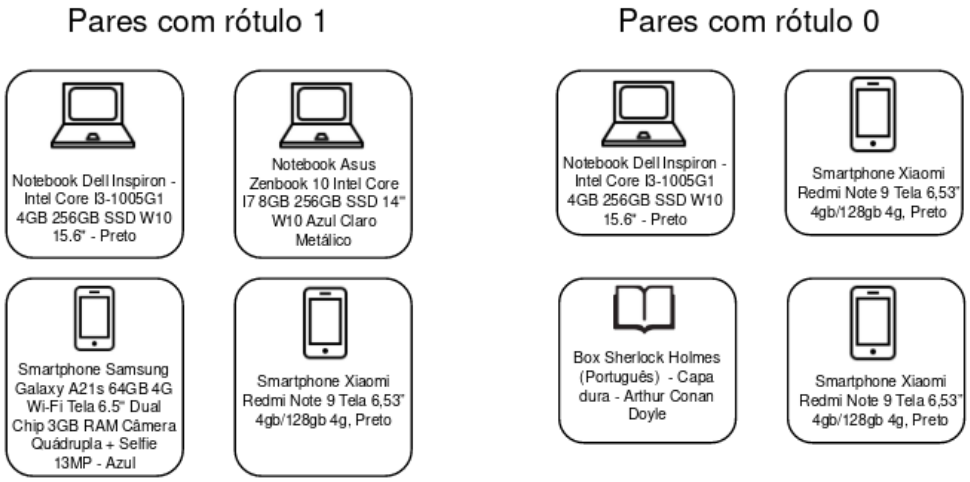
O uso de redes siamesas é uma boa alternativa para resolver problemas com muitas classes e poucos exemplos por classe ou problemas que necessitem incluir/remover classes do modelo constantemente. O modelo aprende uma função para medir a similaridade entre as entradas e gera *features* (características) em um espaço de similaridade semântica. Essas características obtidas na saída podem ser utilizadas em classificadores posteriormente, para tarefas de classificação, por exemplo.

O treinamento das redes siamesas é feito fornecendo pares de amostras que são conhecidamente da mesma classe e pares de amostras de classes distintas. A base de dados usada no treinamento deve passar por processos adequados de mineração de dados e pré-processamento. Um dos métodos para treinar siamesas é pela aprendizagem usando a medida de similaridade das amostras do par. O treinamento da rede siamesa é feito utilizando pares de amostras. Os pares positivos são amostras que pertencem à mesma classe. Os pares negativos são amostras que não pertencem à mesma classe.

Uma das maneiras de obter os pares de amostras para o treinamento é descrita a seguir. Para formação um par positivo, é escolhida uma amostra de uma classe. Em seguida, randomicamente, é escolhida outra amostra da mesma classe. Esse par é rotulado com o valor 1, indicando que as amostras são da mesma classe. Para formar um par de classe diferente, é escolhida uma amostra de qualquer classe. Em seguida, as amostras dessa classe são retiradas do conjunto e é excluída, randomicamente, uma amostra de uma das classes restantes. Esse par negativo é rotulado com o valor 0. A Figura 2.10 mostra exemplos de pares com os rótulos 0 e 1.

A rede siamesa é formada por duas sub-redes idênticas, que possuem a mesma estrutura e são configuradas com os mesmos pesos e parâmetros. A rede neural utilizada pode ser escolhida de acordo com as características do problema e é representada pela função f. Ambas as sub-redes recebem as entradas 1 e 2 ($x1$ e $x2$) e produzem as saídas 1 e 2 ($h1 = f(x1)$ e $h2 = f(x2)$), fazendo a extração de características das entradas, em um espaço multidimensional de similaridade semântica, criado pela rede siamesa. Esse espaço tem a mesma dimensão do número de neurônios de saída das sub-redes. É importante enfatizar que, do ponto de vista da implementação do código, as sub-redes representadas na Figura 2.11 são a mesma rede neural

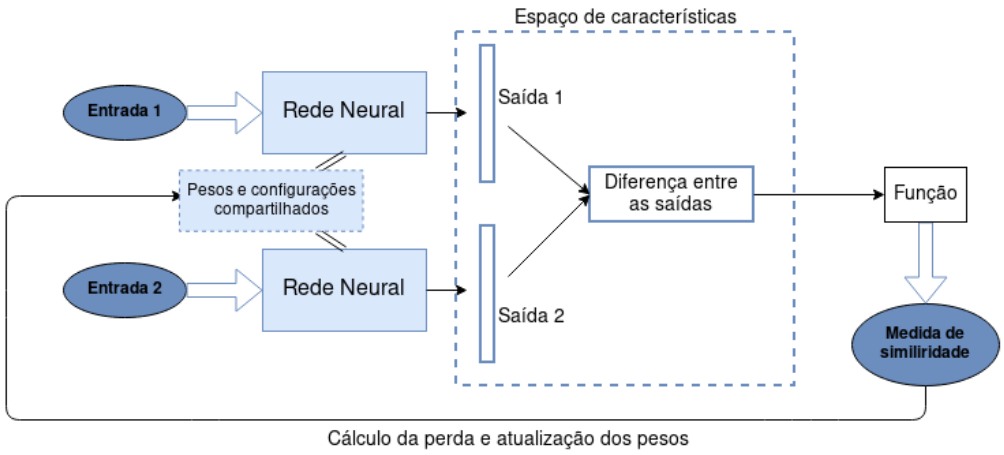
Figura 2.10 – Exemplos de pares para a rede siamesa



Fonte: Do Autor (2022)

(função f) para a extração de características, que recebe os elementos do par de entrada, um por vez.

Figura 2.11 – Arquitetura de uma rede siamesa



Fonte: Do Autor (2022)

O nome "rede siamesa"é inspirado em gêmeos siameses, que é o termo usado na área médica para quando dois irmãos tem uma parte de seus corpos conectados. De forma semelhante, é possível imaginar que cada rede é um 'corpo' da rede siamesa (a parte representada pelas entradas e a rede neural com a função f) e a 'cabeça' da rede siamesa é representada pela parte do cálculo da distância até a obtenção do valor de similaridade. Assim, cada sub-rede teria seu próprio 'corpo', mas estariam unidas pela 'cabeça'.

Segundo o trabalho de Roy et al. (2019), a rede neural siamesa leva os dados de entrada para um espaço de similaridade semântica, no qual os padrões relacionados são próximos uns dos outros e padrões não relacionados são distantes uns dos outros. Com isso, a rede desenvolve uma capacidade de generalização e pode determinar se novos dados pertencem ou não a uma classe. Para impedir que a rede tenha um *overfitting* e perca sua capacidade de generalização, as bases de dados fornecidas para treinamento precisam estar balanceadas, com amostras dos pares de classes iguais e diferentes bem representados.

Depois que as características são extraídas, é calculada a diferença entre as saídas $h1$ e $h2$. O neurônio que faz a junção dos vetores de saída das sub-redes no espaço de similaridade também é responsável por medir ou quantificar a distância entre eles. Para calcular a distância entre as amostras, podem ser utilizados diferentes métodos, como o cálculo da distância ponto a ponto (Equação 2.1) ou a distância Euclidiana (Equação 2.2). O resultado desse cálculo é o vetor $z = |h1 - h2|$ que é a diferença entre os dois vetores de características. O objetivo da rede siamesa é aprender uma função tal que ela consiga maximizar a distância entre os pares de classes diferentes e minimizar a distância entre os pares da mesma classe nesse novo espaço de similaridade.

O vetor z é então transformado em um valor escalar e processado por uma função que normaliza os dados no intervalo de 0 a 1. Uma das funções que pode ser utilizada para a normalização é a função sigmoide, dada pela equação 2.5. O valor normalizado dado por essa função é o valor de similaridade entre as amostras e pode ser usado para definir se as amostras são de uma mesma classe ou não, de acordo com os limiares estabelecidos para que as amostras sejam consideradas similares. Se as amostras do par analisado são da mesma classe o valor de similaridade $sim(x1,x2)$ deverá estar mais próximo de 1, e se são de classes diferentes, os valores estarão mais próximos de 0.

$$f(x) = \frac{1}{1+e^{-x}} \ , \ para\ todo\ x\ real \tag{2.5}$$

Depois de obtido o valor de similaridade, é calculado a perda da rede, pela diferença entre o valor de similaridade $sim(x1,x2)$ e o valor do alvo (também chamado de *label* ou *target*) previamente definido na montagem dos pares. Esse valor de perda é utilizado para atualização dos pesos das sub-redes, utilizando *backpropagation*, como mostrado na Figura 2.11.

No trabalho de Chopra, Hadsell e Lecun (2005) o treinamento da rede siamesa é feito utilizando a função de Perda Contrastiva (*Contrastive Loss*) como função de custo. Assim, o

espaço euclidiano construído tem os pares que são da mesma classe próximos um dos outros e os pares que são de classes diferentes vão sendo distanciados. A função de perda contrastiva é dada pela Equação 2.6, onde M é o valor de margem de similaridade que determina se os pares são genuínos ou impostores; Y_a é o valor da similaridade previsto entre os elementos do par e Y_p o valor associado aos pares previamente, sendo 1 para pares genuínos e 0 para pares impostores (MARTIN et al., 2017).

$$Perda(sim(x_1,x2),\ Y_p(x_1,x_2)) = Y_p * \frac{1}{2} * Y_a^2\ +\ (1-Y_p) * \frac{1}{2} * [max(0,M-Y_a)]^2 \tag{2.6}$$

Se as amostras do par pertencem à mesma classe, suas representações vetoriais devem ser semelhantes. O valor $Y_p = 1$ e o segundo termo desaparece da equação. O valor Y_a deve ser minimizado, pois a perda irá diminuir à medida que o valor Y_a diminuir. Se as amostras são de classes diferentes, $Y_p = 0$ e o primeiro termo desaparece da equação, portanto, o resultado da operação $(max(0,M-Y_a))$ deve ser maximizado, pois a perda será diminuída à medida em que a distância entre as saídas aumentar. O uso da margem M é interessante pois, quando as amostras já são suficientemente diferentes (Y_a é maior que M), a perda será 0 e não poderá ser minimizada. Logo, o modelo irá rejeitar esse par, ou seja, não tentará separá-lo ainda mais, e irá focar em outros pares de entrada.

A perda contrastiva é calculada como a soma das perdas individuais para pares genuínos e impostores. O primeiro termo da soma penaliza os pares genuínos que estão muito distantes. O segundo termo penaliza os pares impostores que estão dentro do valor aceitável de distância.

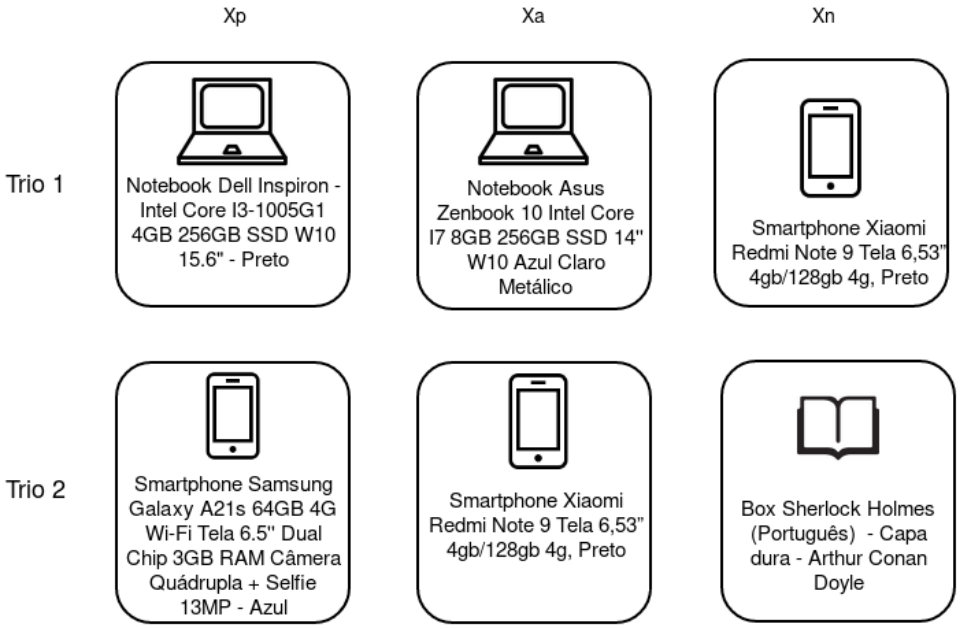
Outra função de perda que pode ser utilizada é a função de entropia cruzada (*cross entropy*), dada pela Equação 2.7. Considerando que neste problema os alvos (*labels*) são 0 e 1, Y é o *label* atribuído à amostra (valor real) e P é a probabilidade da amostra pertencer à classe analisada.

$$Perda(sim(x_1,x2),\ Y(x_1,x_2)) = -Y * log(P)\ -\ (1-Y) * log(1-P) \tag{2.7}$$

A cada época de treinamento o peso das redes siamesas é atualizado via *backpropagation* e a penalização dos pares faz com que os pares genuínos sejam atraídos e os pares impostores vão se distanciando.

Outra forma de treinar as redes siamesas é mostrada no trabalho de Weinberger e Saul (2009), que utiliza o conceito de *triplet loss* (perda trigêmea, em tradução livre). Nesta abordagem, os autores criaram trios de amostras, ao invés de pares de amostras para o treinamento da rede. O trio de treinamento no formato (Xa, Xp, Xn) é composto pela amostra rotulada Xa, chamada de ponto de ancoragem, Xp, uma amostra da mesma classe, e Xn, uma amostra de classe diferente, conforme mostrado na Figura 2.12.

Figura 2.12 – Exemplo de trio para rede siamesa com perda *triplet loss*



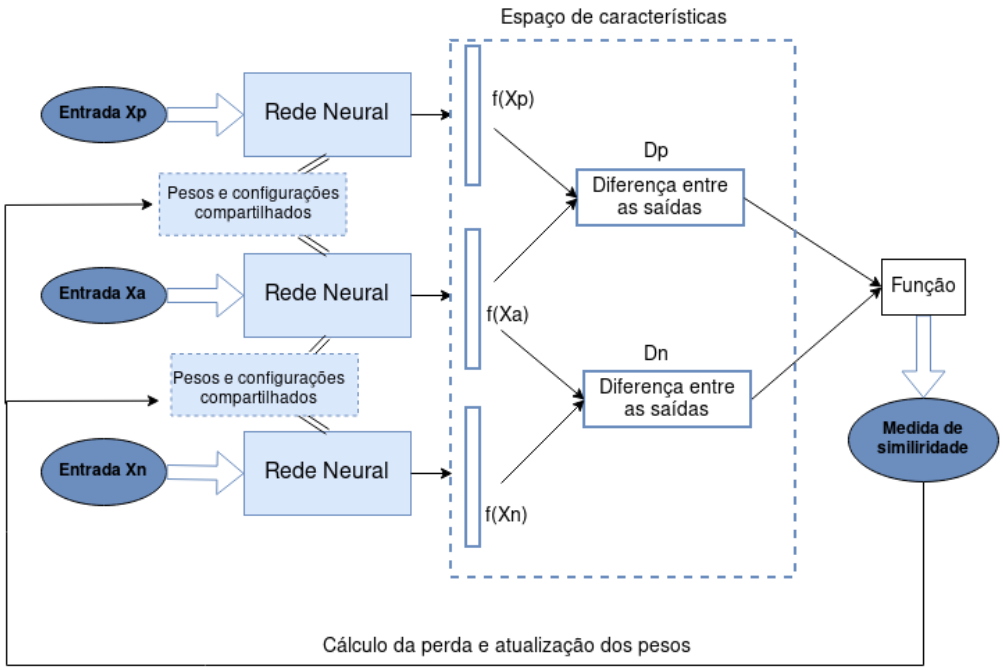
Fonte: Do Autor (2022)

A Figura 2.13 ilustra a estrutura e funcionamento da rede utilizando a função de perda *triplet loss*. Importante ressaltar que, apesar de representadas três redes neurais na Figura 2.13, só existe uma rede neural f. As amostras (Xa, Xp, Xn) passam pela função f da sub-rede neural. Essas amostras são mapeadas para o novo espaço de similaridade, sendo a saída da rede $f(X)$ para cada amostra. Em seguida, é calculada a distância d entre as amostras positiva e negativa e a amostra de ancoragem. Para o cálculo da distância entre $f(Xa)$, $f(Xp)$ e $f(Xn)$ é utilizada a norma Euclidiana, como mostrado na Equação 2.8 e Equação 2.9.

$$Dp = \|f(Xp) - f(Xa)\|_2^2 \tag{2.8}$$

$$Dn = \|f(Xa) - f(Xn)\|_2^2 \tag{2.9}$$

Figura 2.13 – Arquitetura de uma rede siamesa com perda *triplet*



Fonte: Do Autor (2022)

A Equação 2.10 apresenta a formulação da *triplet loss*, sendo que o valor da margem M deve ser maior que 0 e indica a distância necessária entre as amostras positivas e negativas para que a função de perda não aumente significativamente. Caso a distância Dn entre a amostra negativa Xn e a amostra de ancoragem Xa seja maior que a distância Dp entre a amostra positiva Xp e a amostra de ancoragem Xa mais a margem M, não há perda.

$$Perda(Xa, Xp, Xn) = max(0,\ (M\ +\ Dp\ -\ Dn)) \tag{2.10}$$

No caso da *triplet loss*, a amostra negativa força o aprendizado na rede e a amostra positiva funciona como um regularizador. Os trios de entrada trigêmeos devem ser montados de forma que não sejam muito fáceis (amostras muito parecidas), mas também não muito difíceis (amostras muito diferentes) para o aprendizado do modelo (WEINBERGER; SAUL, 2009).

2.6 *Ensembles*

Um *ensemble*, ou comitê de decisão, é entendido como a fusão de vários modelos treinados que, quando combinados, tem o objetivo de melhorar o desempenho de um modelo simples (GONZáLEZ et al., 2020). Usando as previsões combinadas, o modelo final faz previsões

melhores, conseguindo generalizar suas previsões e considerar informações vindas de partes específicas da base de dados. O uso de uma estrutura de *ensemble* pode ser feito em casos em que se deseja reduzir o viés do modelo.

Um tipo de *ensemble* é o *bagging*, em que diferentes modelos base são treinados em pequenas partes da base de dados e o resultado é composto pela média dos resultados de cada modelo no caso de problemas de predição (GENTLE; HARDLE; MORI, 2012). Outro tipo é o *boosting*, em que os modelos são treinados em sequência e fazem ajustes nos pesos aproveitando também as informações dos erros dos modelos anteriores (GENTLE; HARDLE; MORI, 2012).

Em casos de classificação, uma das estratégias para definição do resultado é a contagem por votação, em que a classe mais votada é atribuída à amostra. Já no tipo de *ensemble stacking*, é feito o treinamento de um algoritmo que combina a predição de vários outros algoritmos para fazer a predição final para o problema (GENTLE; HARDLE; MORI, 2012).

2.7 Estratégias de avaliação dos resultados

2.7.1 Bases de dados

As bases de dados utilizadas neste trabalho foram cedidas pela empresa parceira. Porém, as bases de dados listadas abaixo estão disponíveis na literatura e podem ser utilizadas, uma vez que possuem informações semelhantes às que foram necessárias para realização desta pesquisa.

a) *Amazon Customer Reviews Dataset*: Uma coleção de análises feitas no mercado online *Amazon.com* [2], de 1995 a 2015, e os metadados associados. Contém mais de 200 mil amostras de comentários feitas por usuários de 5 países sobre produtos em vários idiomas.

b) *eCommerce Item Data*: Contém códigos alfanuméricos usados por comerciantes com o objetivo de identificar tipos de produtos e variações, além das suas descrições no catálogo de produtos de uma marca de vestuário para uso externo.

c) *ECommerce Search Relevance*: Contém informações de URLs de imagem, classificação na página, descrição de cada produto, a pesquisa que leva a cada resultado, entre outros, para produtos em cinco principais sites de comércio eletrônico de idioma inglês.

[2] *https://www.amazon.com/*

d) *Dataset Mercado Livre Data Challenge 2019*: Cada linha do conjunto corresponde a um produto no Mercado Livre [3]. Cada amostra contém o título do anúncio, linguagem (português, inglês ou espanhol), qualidade e categoria do produto.

2.7.2 Métricas utilizadas

Uma das formas de avaliar os resultados obtidos é utilizando a matriz de confusão, que é uma ferramenta usada para avaliações de modelos de classificação em aprendizado de máquina, e as métricas calculadas a partir dos dados nela dispostos. Ela é apresentada na forma de uma tabela contendo as frequências de classificação para cada classe do modelo. O Quadro 2.1 ilustra uma matriz de confusão para duas classes genéricas A e B.

Quadro 2.1 – Matriz de confusão

		Reais	
		Classe A	Classe B
Preditas	Classe A	TP	FP
	Classe B	FN	TN

Fonte: Do Autor (2022)

Considerando que a classe de interesse, que se deseja prever, é a classe A, os valores de cada coluna correspondem aos valores de Verdadeiro Positivo (TP, do inglês *true positive*), que é quando a classe é prevista corretamente; Falso Positivo (FP, do inglês *false positive*), quando a classe foi prevista incorretamente, sendo que o modelo previu a amostra pertencente à classe A, porém a amostra pertence à classe B. O valor de Verdadeiro Negativo (TN, do inglês *true negative*) é relativo aos casos em que o modelo prevê a classe B (a classe que não está se buscando prever) corretamente e o Falso Negativo (FN, do inglês *false negative*) ocorre quando a classe foi prevista incorretamente, sendo que o modelo previu a classe pertencente à classe B, porém a amostra pertence à classe A.

Com os valores obtidos na matriz de confusão são calculados os índices de Precisão (Equação 2.12), Acurácia (Equação 2.11), *Recall* (Equação 2.13) e *F1-score* (Equação 2.14) para avaliação dos resultados obtidos e desempenho da solução proposta. Além desse índices, também será avaliado o índice de Cobertura (Equação 2.15) do classificador.

$$Acurácia = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.11}$$

[3] *https://www.mercadolivre.com.br/*

$$Precisão = \frac{TP}{TP+FP} \tag{2.12}$$

$$Recall = \frac{TP}{TP+FN} \tag{2.13}$$

$$F1 = \frac{2 * Precisão * Recall}{Precisão + Recall} \tag{2.14}$$

$$Cobertura = \frac{TP+FP}{TP+TN+FP+FN} \tag{2.15}$$

O índice de Acurácia indica qual a porcentagem de amostras que tiveram valores extraídos corretamente, considerando toda a base de dados. O índice de Precisão indica a confiabilidade do modelo, ou seja, qual a porcentagem de amostras do subconjunto que atendeu os índices de qualidade estabelecidos tiveram valores extraídos corretamente. O índice *Recall* indica a sensibilidade do modelo e é calculada pela porcentagem de amostras que foram corretamente extraídas. O valor *F1-score* é obtido usando a média harmônica dos valores de Precisão e *Recall*, gerando um número único que indica a qualidade geral do modelo. O índice de cobertura indica a porcentagem de amostras que obtiveram uma classificação que atende aos índices de qualidade determinados, considerando todas as amostras da base de dados.

2.7.3 Redução de dimensionalidade

Em alguns problemas os conjuntos de dados que serão analisados tem variáveis que são dependentes entre si e os dados tem dimensões elevadas, ou seja, possuem muitas características. A redução de dimensionalidade é uma estratégia utilizada na análise de dados e aprendizado de máquina para transformar os dados de um espaço de alta dimensão para um espaço de baixa dimensão, fazendo com que representação de baixa dimensão retenha as propriedades importantes dos dados originais em alta dimensão. Nesses casos, podem ser usados diversos métodos de redução de dimensão para identificar os conjuntos de variáveis que não são correlacionadas entre si e que ainda assim conservam a maior parte de informações relevantes para descrever os dados.

A capacidade de generalizar corretamente se torna exponencialmente mais difícil conforme a dimensionalidade do conjunto de dados de treinamento aumenta, pois o conjunto de

treinamento cobre uma fração cada vez menor do espaço de entrada. Os modelos também se tornam mais eficientes à medida que o conjunto reduzido de recursos aumenta as taxas de aprendizado e diminui os custos de computação ao remover recursos redundantes.

A Razão de Discriminação de Fisher (FDR, do inglês *Fisher Discriminante Radio (FDR)* é um método estatístico para encontrar uma combinação linear de características que caracterizem ou separem duas ou mais classes de objetos ou eventos (DUDA; HART; STORK, 2012). O FDR faz uma projeção dos dados maximizando a separação de classes e permitindo uma pequena variância dentro de cada classe, minimizando assim a sobreposição de classes (BISHOP, 2006). O FDR busca uma pequena variação dentro de cada uma das classes e uma grande variação entre as classes do conjunto de dados.

Já a Análise de Componentes Principais (PCA, do inglês *Principal Component Analysis*) é uma técnica estatística não paramétrica e não supervisionada, usada para redução de dimensionalidade no aprendizado de máquina. Segundo Duda, Hart e Stork (2012), usando o PCA, as características originais de um conjunto de dados são transformadas em um novo conjunto de *n* características chamado componentes principais. Após a transformação, o primeiro componente principal tem a maior variância possível e cada componente subsequente tem a menos variância e mais ruído que os componentes anteriores.

2.7.4 Método *Leave-one-out*

O *Leave-one-out* é um método de validação cruzada. A validação cruzada é uma técnica utilizada para avaliar a capacidade de generalização de um modelo, ou seja, para analisar qual vai ser o desempenho do modelo para um novo conjunto de dados. Nas técnicas de validação cruzada o conjunto de dados é dividido em subconjuntos mutuamente exclusivos, sendo um conjunto utilizado para treino e estimação de parâmetros e o outro conjunto é utilizado para o teste do modelo (DUDA; HART; STORK, 2012).

No método de validação cruzada *Leave-one-out*, cada amostra do banco de dados é utilizada como um "conjunto de teste"(AYUSO; SOLER, 2012). O modelo então é avaliado em relação ao seu desempenho em classificar essa amostra específica. O número de execuções do teste é igual ao número de amostras do banco. Para calcular a acurácia total do modelo é feita a média das acurácias de cada teste realizado.

No caso das redes siamesas, em cada execução do teste, o modelo recebe um par de amostras para avaliação. Um elemento do par de amostras será a amostra escolhida no *Leave-*

one-out (elemento n da base, começando pelo primeiro elemento, indo até o número de amostras disponíveis) e o outro elemento do par será uma das amostras restantes, até que todas as amostras restantes tenham sido utilizadas como o segundo elemento do par. O modelo é então avaliado com esses pares de teste. Na próxima rodada de teste, a próxima amostra da base (elemento $n+1$) será utilizada como o primeiro elemento do par de teste. O teste *Leave-one-out* é repetido até que todas as amostras da base tenham sido utilizadas como o primeiro elemento do par.

2.8 Trabalhos relacionados

Com o aumento das transações de compra e venda de produtos e serviços *online*, o número de ofertas encontradas na *internet* tem aumentado consideravelmente. O crescimento dessa modalidade de compras pode ser explicado pela comodidade que o comprador tem em poder fazer suas compras de qualquer lugar, necessitando apenas de uma conexão com a *internet*, e receber suas compras em casa ou em qualquer outro endereço que deseje, podendo até mesmo fazer o pagamento por cartões de crédito, sem ter a necessidade de ir até alguma agência física de um banco ou outras instituições para realizar o pagamento. Cada operação de compra e venda, ou até mesmo a busca por esses produtos, geram dados que podem ser usados pelas empresas, tendo cada vez uma maior quantidade e variedade de dados disponíveis, relativos à essas transações (EDOSIO, 2014).

As plataformas de compra e venda *online* permitem que os vendedores cadastrem seus produtos informando as características que desejarem e que classifiquem o produto na categoria que escolherem. Portanto, não há uma padronização da descrição dos produtos, fazendo com que o mesmo produto possa ser cadastrado com descrição ou categoria diferente, de acordo com o entendimento de cada vendedor. Diante disso, o entendimento semântico de textos é de suma importância, o que pode permitir a adequada categorização de produtos, assim como uma melhor interpretação dos desejos do consumidor.

O uso de ferramentas de processamento de linguagem natural pode auxiliar no processo de classificar o produto de acordo com o nicho correto, tendo como base a descrição fornecida pelo anunciante. Além disso, pode ser usado no mapeamento dos tipos de produto (entidades e atributos) nas categorias do *marketplace*, independentemente da categoria originalmente atribuída pelo vendedor. Tal categorização pode melhorar a experiência de compra para os usuários

da plataforma, pois os produtos estarão agrupados na categoria correta, facilitando a busca pelos mesmos.

Kannan et al. (2011) apresentam um sistema automatizado usando análise semântica para fazer a correspondência da descrição de ofertas não estruturadas recebidas no *Bing Shopping* [4] para descrições estruturadas do catálogo de produtos. O sistema foi implementado com um componente orientado a dados que aprende uma função de correspondência durante o treinamento e aplica essa função, em tempo de execução, para realizar a correspondência das ofertas de produtos. Para o treinamento do algoritmo foram utilizadas 20 mil amostras, sendo que cada classe de produto tinha uma média de 425 amostras de descrição de ofertas e cada oferta continha um texto não estruturado com diversas informações sobre os atributos do produto. O algoritmo de correspondência proposto é baseado na compreensão da semântica das descrições da oferta. Essa semântica é usada para ajudar a encontrar uma função de correspondência. Neste trabalho foi observado que em algumas categorias de produtos o algoritmo apresentava melhor desempenho que em outras categorias. Essa diferença foi atribuída às diferentes combinações de atributos presentes em cada categoria de produtos. Devido ao grande número de atributos, o algoritmo não aprende todas as relações possíveis entre eles, somente aquelas em que o número de exemplos disponíveis alcançou um determinado limite, o que prejudicou a obtenção da função de correspondência para algumas classes de produtos.

Essas correlações entre atributos são importantes quando se deseja mapear um perfil de usuário, como no trabalho de Ghani e Fano (2002), que desenvolveram um sistema de recomendação que analisa as descrições dos produtos que o usuário navega ou compra e infere automaticamente esses atributos semânticos para criar um perfil do usuário, permitindo entender seus gostos e recomendar outros itens da mesma classe de produtos que possam despertar o interesse do consumidor. Além disso, o sistema utiliza uma abordagem inovadora, com uma base de conhecimento customizada, contendo informações sobre os produtos e seus atributos, o que possibilita que o sistema também recomende outros produtos que combinam com o perfil do usuário, mas são de categorias diferentes.

Ghani et al. (2006) buscam representar cada produto como um conjunto de pares atributo-valor. Os autores apresentam resultados promissores em produtos de vestuário e artigos esportivos, mostrando que o sistema desenvolvido extrai pares de atributos e valores das descrições dos produtos com precisão. Já Kozareva (2015) propõe um mecanismo de catego-

[4] *https://www.bing.com/*

rização automática do produto que, para um determinado título de produto, atribui a categoria correta do produto a partir de uma taxonomia. Tal trabalho é importante pois cada plataforma organiza os produtos em diferentes taxonomias, tornando difícil e demorado para os vendedores categorizar mercadorias de acordo com cada plataforma de compras. Foi construído um algoritmo de classificação multi-classe que pode classificar o título do anúncio de um produto não rotulado em uma das 319 classes disponíveis.

Ristoski et al. (2018) apresentam uma abordagem que utiliza modelos de linguagem neural e técnicas de aprendizado profundo em combinação com abordagens de classificação padrão para correspondência e categorização de produtos. Também usam dados estruturados do produto para treinamento de modelos de extração de características textuais do produto. A solução proposta conta com duas etapas, a extração de características e a classificação dos produtos, e pode ser usada em aplicações de recomendação de produtos, recuperação de informação e processamento de consultas de pesquisa em várias áreas, desde que sejam feitas as adaptações necessárias de acordo com os dados disponíveis.

No trabalho de Tyler e Teevan (2010) é explorado como os mecanismos de pesquisa são usados para encontrar a resposta de uma consulta usando resultados de pesquisa encontrados anteriormente. Uma abordagem diferente é proposta por Bar-Yossef e Kraus (2011) com o algoritmo *NearestCompletion*, usando um novo método para medir a similaridade de consultas. Para medir essa similaridade, as consultas e contextos foram representados como vetores de termos ponderados de alta dimensão e também foi utilizada a similaridade de cosseno.

Em seu artigo, Li, Kok e Tan (2018) propõe um novo paradigma baseado na tradução automática, em que o idioma natural de um produto é traduzido em uma sequência de *tokens*, representando um caminho raiz-para-folha em uma taxonomia de produto. Esse paradigma apresentou um bom desempenho para as duas bases de dados em que foi testado, além de apresentar novos caminhos que podem ser úteis para o caso de inserção de novos produtos ou mudanças nos produtos já existentes. No modelo de tradução de máquina são usados os modelos *seq2seq* e *Transformers*, que são tipos de arquiteturas de redes neurais usadas para tradução de máquina e problemas de tradução sequencial.

Já Kiros et al. (2015) propuseram o modelo *skip-thoughts*, que estende a abordagem de *skip-gram* do *word2vec* do nível da palavra para o nível de frase. Nesta implementação, o codificador-decodificador RNN é alimentado com cada sentença de textos de livros e o modelo tenta reconstruir as sentenças imediatamente anteriores e seguintes àquela analisada. Os vetores

extraídos do modelo foram avaliados nas tarefas de relacionamento semântico, detecção de paráfrase, classificação de sentenças de imagem e classificação de tipo de pergunta. O codificador produziu representações de sentenças altamente genéricas com bom desempenho.

Tai, Socher e Manning (2015) utilizaram LSTMs, organizadas em forma de árvore, pois as propostas anteriores somente tratavam a organização em linha. A chamada *Tree-LSTM* supera os métodos com os quais foi comparada nas tarefas de prever a relação semântica de duas frases e classificação de sentimentos. Publicações mais recentes propõe o uso de LSTM juntamente com redes siamesas para resolver problemas de similaridade entre palavras e sentenças (MUELLER; THYAGARAJAN, 2016; BENAJIBA et al., 2019).

O uso das redes siamesas para comparação entre dois objetos foi proposto em 1994 para verificação de assinaturas escritas em um *tablet* com entrada por caneta. No trabalho de Bromley et al. (1994), a verificação da legitimidade de uma assinatura, ou seja, se uma amostra pertence a uma classe, é feita comparando um vetor de recurso extraído com um vetor de recurso armazenado para o assinante, sendo que o vetor armazenado pertence ao assinante. As assinaturas mais próximas, no espaço de similaridade semântica, a essa representação armazenada do que uma terceira opção escolhida são aceitas. As assinaturas distantes dessa representação armazenada são consideradas falsificações e são rejeitadas.

No trabalho de Koch, Zemel e Salakhutdinov (2015) as redes neurais siamesas foram usadas para resolver um problema de *One-shot Image Recognition*. Usando uma arquitetura convolucional, obtiveram resultados que superaram os de outros modelos de aprendizado profundo para resolução desse problema. Para o cálculo da distância entre as entradas no neurônio de saída foi utilizado a distância linear e as entradas foram unidas usando a função de ativação sigmoide.

O trabalho de Mueller e Thyagarajan (2016) propõe uma adaptação ao trabalho de Koch, Zemel e Salakhutdinov (2015), propondo um modelo implementado com redes siamesas e LSTM para análise de dados textuais rotulados, compostos por pares de sequências de comprimento variável. O modelo de aprendizado supervisionado proposto, chamado de *Manhattan LSTM (MaLSTM)*, foi aplicado para avaliar a semelhança semântica entre as frases, superando os resultados de sistemas de redes neurais anteriormente conhecidos.

Abdalhaleem, Barakat e El-Sana (2018) também se inspiraram no trabalho de Koch, Zemel e Salakhutdinov (2015) para medir pequenas mudanças nos estilos de escrita da mesma pessoa, analisando manuscritos digitalizados. Foram feitas mudanças na estrutura da rede pro-

posta anteriormente para evitar *overfitting*, por exemplo as mudanças do *Kernel* e do número de filtros por camada. Um dos desafios encontrados neste trabalho foi construir um conjunto de treinamento para a rede siamesa, já que o conjunto deve ser grande o suficiente para representar bem o espaço de entrada e também ser muito maior que o número total de pesos na rede, como forma de evitar o *overfitting*, ou seja, o desempenho do modelo alcança valores satisfatórios somente para o conjunto de treino, não tendo um desempenho bom para prever resultados de novos dados (ABDALHALEEM; BARAKAT; EL-SANA, 2018).

Em outra área de aplicação, o reconhecimento de voz, o trabalho de Velez, Rascon e Fuentes-Pineda (2018) apresenta uma nova abordagem para identificação de voz que foi desenvolvida usando uma arquitetura de rede neural convolucional siamesa, onde a rede aprende a verificar se dois sinais de áudio são da mesma pessoa. Os pesquisadores utilizaram um banco de dados externo de áudio gravado dos usuários e a identificação é feita verificando a nova entrada de fala com cada uma das entradas do banco. Se novos usuários fossem encontrados, seria necessário adicionar o áudio gravado ao banco de dados externo para poder ser identificado, sem retreino do modelo utilizado.

Em seu trabalho, Souza et al. (2019) utilizam as redes siamesas em uma aplicação *one shot learning* para autenticação e identificação de condutores como forma de superar a necessidade de retreinar os modelos a cada vez que um novo condutor precisa ser identificado. Foram utilizados os dados obtidos com a participação de 25 voluntários, sendo os dados de 13 condutores usados para treino e o restante para a validação do modelo proposto. Os resultados foram analisados usando a acurácia para as diferentes topologias de redes neurais propostas. Uma das limitações do trabalho desenvolvido, é analisar a capacidade de generalização de dados do modelo, caso fossem analisadas diferentes seções dos dados de direção de cada condutor.

Já Sreepada e Patra (2020) propõe o uso das redes siamesas para um sistema de recomendação de produtos de uma plataforma de vendas *online*, recomendando itens pouco populares para compradores específicos de acordo com as preferências já observadas para eles. Essas preferências foram agrupadas em diferentes *clusteres*, sendo eles: "produtos curtidos", "produtos não curtidos"e "produtos de cauda longa"e, em seguida, as redes siamesas foram usadas para aprender os interesses de cada usuário de acordo com seus *clusteres*. O modelo treinado é capaz de recomendar itens gerais e itens relevantes de cauda longa para cada usuário. Os autores também introduzem três métricas para avaliar o desempenho das recomendações em itens de cauda longa (*Long Tail Coverage* (LTC), *Weighted Long Tail Coverage* (WLTC) e *Tail of the*

Tail Coverage). Essas métricas e a abordagem usando redes siamesas para o problema de recomendação de produtos de cauda longa foram validadas nas bases de dados (*MovieLens 1M* e *Netflix*). A estrutura proposta no trabalho superou as abordagens tradicionais e as técnicas de recomendação de cauda longa existentes.

2.9 Conclusão

Foram apresentadas algumas aplicações de redes siamesas para o problema de *one shot learning* em diferentes áreas. De forma similar aos trabalhos apresentados, vê-se a aplicação de redes siamesas para solucionar o problema deste trabalho, de classificar ofertas em novas classes, sem que tenha que retreinar o classificador a cada classe nova identificada. Outro desafio do problema deste trabalho que pode ser resolvido com o uso das redes siamesas é o número limitado de amostras de cada classe, além do alto custo do retreino do classificador à cada vez que for adicionada uma amostra de uma nova classe à base de dados.

3 METODOLOGIA

Esse capítulo descreve os materiais utilizados, a metodologia e as técnicas que foram empregadas na pesquisa e desenvolvimento do trabalho, visando a implementação de um sistema com aprendizagem *few shot learning* para aplicação em problemas de classificação de produtos em uma plataforma de comércio eletrônico.

3.1 Acordo de parceria

O presente trabalho é desenvolvido pelo Acordo de Parceria 03/2020 PD&I (Pesquisa, Desenvolvimento e Inovação) firmado entre a Universidade Federal de Lavras (UFLA) e a empresa *Omnilogic* [1] com o objetivo de desenvolver estudos de ferramentas de aprendizado de máquina para aplicações em comércio eletrônico. O contrato de parceria tem vigência de 01/04/2020 até 01/05/2022.

A *Omnilogic* é uma empresa de inovação focada em tecnologias de inteligência de dados para operações digitais, oferecendo soluções de alto desempenho criadas para descoberta de inteligência coletiva, baseada nos padrões de comportamento de usuários e inteligência de produtos, tendo aplicações nos segmentos de varejo, mercado financeiro, turismo, manufatura, educação e saúde. Possui clientes reconhecidos no mercado de comércio eletrônico como B2W, Magazine Luiza, *Walmart, Avon, Naspers/Buscapé, Multiplus*, CVC, Submarino Viagens, entre outros.

A empresa foi fundada em 2009 e, atualmente, possui sede na cidade de Belo Horizonte (MG), no bairro Savassi. Possui cerca de 50 funcionários e financia equipes de pesquisadores dedicados na forma de convênios firmados com laboratórios na Universidade Federal de Minas Gerais (UFMG) desde 2018 e com o Laboratório do grupo de pesquisa de Automação e Inteligência Artificial (AIA), da UFLA, desde 2020.

3.2 Ferramentas e softwares

O equipamento utilizado para a pesquisa foi um computador com especificações que suportavam o processamento do alto volume de dados e execução dos softwares necessários para obtenção dos resultados. As especificações do notebook utilizado são: marca *Dell*, com processador *Intel Core i5-8250U CPU 1,60GHz* 8a geração, gráficos *Intel UHD Graphics 620*

[1] *https://www.omnilogic.com.br*

(Kabylake GT2), 8Gb de memória RAM, sistema operacional *Linux Fedora 31*. O computador, de valor aproximado de R$4.200,00, foi cedido pela *Omnilogic*, empresa parceira neste trabalho.

O processamento dos dados foi realizado inicialmente nos Laboratórios de Processamento de Dados I e II do Programa de Pós Graduação em Engenharia de Sistemas e Automação (PPGESISA). Para realização do processamento dos dados também foi disponibilizada pela *Omnilogic* uma conta para acesso à sua infraestrutura *cloud*, que possibilitou uma estrutura de processamento de dados compatível com a demanda do projeto. Foi disponibilizado o acesso à máquinas virtuais contendo GPU (*Graphics Processing Unit*, ou Unidade de Processamento Gráfico), possibilitando um processamento mais rápido.

Além disso, foi utilizada a plataforma *Google Colaboratory (Colab)*, uma ferramenta em nuvem que permite a escrita e execução de *scripts* direto no navegador, oferecendo acesso gratuito a GPUs, porém com limite de tempo de utilização de 12 horas. Também existem os limites de utilização da memória RAM (aproximadamente 12 GB) e de espaço em disco (100 GB). O *Colab* é um ambiente de fácil configuração e simplifica o compartilhamento dos *scripts*, no caso de um trabalho colaborativo.

Para implementação dos códigos foi utilizada a linguagem de programação *Python*, que tem se popularizado nas áreas relacionadas à análise de dados. Essa linguagem é gratuita e conta com bibliotecas que podem ser usadas para realizar algumas operações de análise de dados, como cálculos matriciais, processamento de dados e plotagem de imagens (COELHO; RICHERT, 2015). Por ser uma linguagem *open source*, desenvolvida de forma comunitária, são criadas novas bibliotecas constantemente, além da atualização das bibliotecas já existentes, com melhorias e correção de erros. As bibliotecas utilizadas neste trabalho foram as seguintes:

a) *Numpy*: Biblioteca com recursos para operações com matrizes e funções matemáticas de alto nível.

b) *Pandas*: Biblioteca com métodos que permitem filtrar, agrupar e combinar dados, fornecendo estruturas de dados de alto nível.

c) *Scikit-Learn*: Biblioteca com algoritmos para tarefas de aprendizado de máquina e mineração de dados, como classificação e regressão.

d) *Keras*: Biblioteca que pode ser usada junto com o *TensorFlow* para manipulação de redes neurais, permitindo experimentação rápida com redes neurais profundas.

e) *TensorFlow*: Biblioteca para criação e treinamento de redes neurais, que podem ser usadas para detectar e decifrar padrões e correlações entre os dados.

f) *MatplotLib*: Usada em conjunto com a biblioteca *Numpy* para plotagem de gráficos e visualização de dados.

g) *Seaborn*: É utilizada juntamente com a biblioteca *Matplotlib* para visualização de dados.

3.3 Banco de dados

A base de dados foi fornecida pela empresa *Omnilogic*, parceira no trabalho. Tal base de dados é formada por milhares de amostras contendo descrição de ofertas informadas por anunciantes e a *label* (classe) atribuída para essas ofertas. As amostras contidas neste banco de dados são pertencentes à classes ainda não utilizadas na operação pela empresa.

Cada amostra contém as seguintes colunas:

a) *id*: um código alfanumérico utilizado para identificação da amostra.

b) *input model*: é o texto original informado pelos anunciantes na plataforma de vendas após passar por um pré processamento. Nesse pré processamento são retirados, por exemplo, sinais de pontuação, conectores de frase, acentos, elementos de *html*, e todo o texto é colocado em letras minúsculas.

c) *intermed layer*: É um vetor numérico com 1200 elementos, que são a saída de uma rede neural profunda utilizada hoje pela *Omnilogic*, para a extração de características do texto pré processado (*input model*). Esse tamanho pode variar de acordo com o extrator de características utilizado.

d) *predict*: classe predita pelo classificador da *Omnilogic*, sendo que essa classe não fazia parte do conjunto de dados utilizado em seu treinamento. O classificador atribui uma das classes conhecidas durante o treinamento para a amostra.

e) *target*: classe real da amostra, definida por um funcionário da empresa após analisar o texto informado pelo anunciante.

A base de dados conta com mais de 446 mil amostras. Inicialmente, foi escolhido trabalhar somente com uma parte desses dados, por demandar um alto poder de processamento e

muito tempo para visualização dos resultados. Os dados selecionados pertencem ao nicho de "Utilidades domésticas", composto por 394 amostras e 34 classes. A distribuição de amostras por classe é mostrada na Tabela 3.1.

Tabela 3.1 – Distribuição das amostras em cada classe

Número de Classes	**Nome das Classes**	**Número de Amostras**
1	Suporte para Galão de Água	113
1	Pêndulo	60
1	Cesta de Supermercado	51
1	Defumador de Ambiente	49
1	Limpador Magnético	32
1	Pedra Esotérica	20
1	Aspersório	8
1	Porta Canudo	6
3	Alça para Galão de Água, Conjunto Esotérico, Lenço *Furoshiki*	5
2	Adaga, Alça para Carregar Sacolas	4
3	Forma para Torre de Batata, Abafador para Narguilé, Base para Varal de Parede	3
5	Porta Máscara Cirúrgica, Degrau para Banheiro, Limpador de Lente, Abridor para Galão de Água, Adaptador para Bico de Confeitar	2
13	Base para Tocha, Extensor de Suporte de TV, Aplicador de Pérolas para Confeitaria, Lamparina, Turibulo, Alça para Garrafa, Capa para Faca, Porta Filtro de Café, Espátula Alisadora para Confeitaria, Redutor para Cachimbo, Tábua de Lavar, Anel de Vedação para Garrafa de *Chantilly*, Armador de Flor para Confeitaria	1

Fonte: Do Autor (2022)

As amostras foram obtidas utilizando o extrator de características desenvolvido e utilizado pela *Omnilogic*, sendo que as ofertas enviadas em linguagem natural são transformadas em um vetor com as características no formato de valores numéricos, que foram usadas nos classificadores *few shot learning* desenvolvidos neste trabalho.

Para reduzir a dimensão dos dados utilizados, foram aplicados os métodos de transformação das características com Análise de Componentes Principais e a seleção de características com Razão de Discriminação de *Fisher* combinado com correlação. Assim sendo, as bases de dados obtidas foram nomeadas da seguinte forma: a base de dados sem redução de dimensão, com 1200 características, foi chamada de Primeira Base de Dados, a base de dados reduzida

por PCA, com 307 características, foi chamada Base PCA e a base de dados reduzida por FDR com correlação, contando com 308 características, foi chamada Base FDR.

Para análise da capacidade do modelo proposto de lidar com um volume de dados maior, bem como com uma grande quantidade de classes, foi disponibilizada outra base de dados, que foi nomeada Segunda Base de Dados. Esta base conta com 4253 amostras distribuídas em 452 classes. Nesta base, o número máximo de amostras por classe é 10 amostras e cada amostra conta com 1000 características. A Tabela 3.2 mostra a relação entre a quantidade de amostras por classe e a quantidade de classes da base.

Tabela 3.2 – Número de classes e número de amostras por classe da Segunda Base de Dados

Número de classes	**Número de amostras**
2	2
3	3
4	5
8	6
15	7
27	8
79	9
314	10

Fonte: Do Autor (2022)

3.3.1 Preparação do conjunto de dados

A rede siamesa enfrenta o problema da escolha do representante de cada classe (amostra referência) pois compara uma amostra rotulada com uma amostra de classe desconhecida para definir a similaridade entre elas. Caso a amostra que represente a classe for um *outlier* ou não estiver bem posicionada no *cluster*, o classificador não terá um bom desempenho.

A abordagem usada como base de comparação da rede siamesa é aquela em que não há a escolha do representante, sendo todas as amostras de treinamento usadas como elemento de referência. As abordagens para escolha do melhor representante de cada classe foram as seguintes:

a) Escolha aleatória

b) Uso do centroide como representante da classe;

c) Escolha do elemento mais próximo do centroide de cada classe;

d) Uso de estrutura *ensemble* com *n* representantes de cada classe;

e) Uso de estrutura *ensemble* com *K-Means* para escolha dos *n* representantes de cada classe;

Por simplicidade, para os testes iniciais de cada abordagem proposta para escolha do melhor representante foram utilizadas somente as classes com mais de 20 amostras. As classes selecionadas foram: Suporte para Galão de Água, Pêndulo, Cesta de Supermercado, Defumador de Ambiente, Limpador Magnético, Pedra Esotérica. Essas classes foram divididas em base de treino e teste seguindo a proporção 70/30, em que 70% das amostras foram usadas no treino e 30% foram usadas no teste. Assim sendo, com o total de 325 amostras, a base de treino ficou com 227 amostras e a base de teste ficou com 98 amostras.

Na etapa de treinamento da rede siamesa é necessário que a rede receba as características extraídas do par de amostras que será analisado e o valor do alvo. Caso as amostras sejam da mesma classe, o alvo será 1, caso sejam de classes diferentes, o alvo será 0.

A formação dos pares de treino enviados para a rede siamesa foi feita de maneira semelhante ao trabalho de Koch, Zemel e Salakhutdinov (2015), em que a escolha dos elementos dos pares é feita de maneira aleatória. Considerando a base de dados de treino do modelo, uma amostra foi escolhida como o primeiro elemento do par. Em seguida, uma outra amostra da mesma classe foi escolhida aleatoriamente, dentre as amostras restantes daquela classe, para ser o outro elemento do par. O alvo deste par foi 1, já que as amostras são da mesma classe. Para manter o equilíbrio nos dados de treinamento enviados para a rede siamesa e evitar que o modelo crie algum viés, também foi gerado um par de classe diferente para a mesma amostra, tendo como alvo 0. O elemento foi escolhido aleatoriamente dentre as amostras de classe diferente da amostra do primeiro elemento do par. Assim sendo, cada amostra da base de treino tinha um par da mesma classe e um par de classe diferente usados no treinamento da rede siamesa.

Os pares de amostras são formados pela amostra de referência (rotulada) e a amostra monitorada (a ser testada). Nos testes preliminares para validação da rede siamesa como método *few shot learning* (Abordagem *baseline* siamesa) e das outras abordagens adotadas, os pares de teste foram montados escolhendo o representante da classe de forma aleatória dentre as amostras usadas no treinamento. A amostra monitorada do par foi uma amostra da base de dados de teste.

Para os testes da Abordagem simples, os pares de teste foram montados de forma que a amostra da base de teste foi considerada como a amostra de referência do par e cada amostra utilizada no treinamento foi considerada como a amostra monitorada.

De maneira semelhante, para a Abordagem com centroide, os pares de teste foram montados de forma que o centroide calculado para cada classe com as amostras disponíveis para treinamento foi considerado como elemento de referência e a amostra da base de teste foi considerada como amostra monitorada.

Já para a Abordagem com *ensemble*, os pares de teste foram montados de forma que cada representante escolhido dentre as amostras de treinamento para cada classe foi considerado como elemento de referência do par e a amostra de teste foi considerada como elemento monitorado.

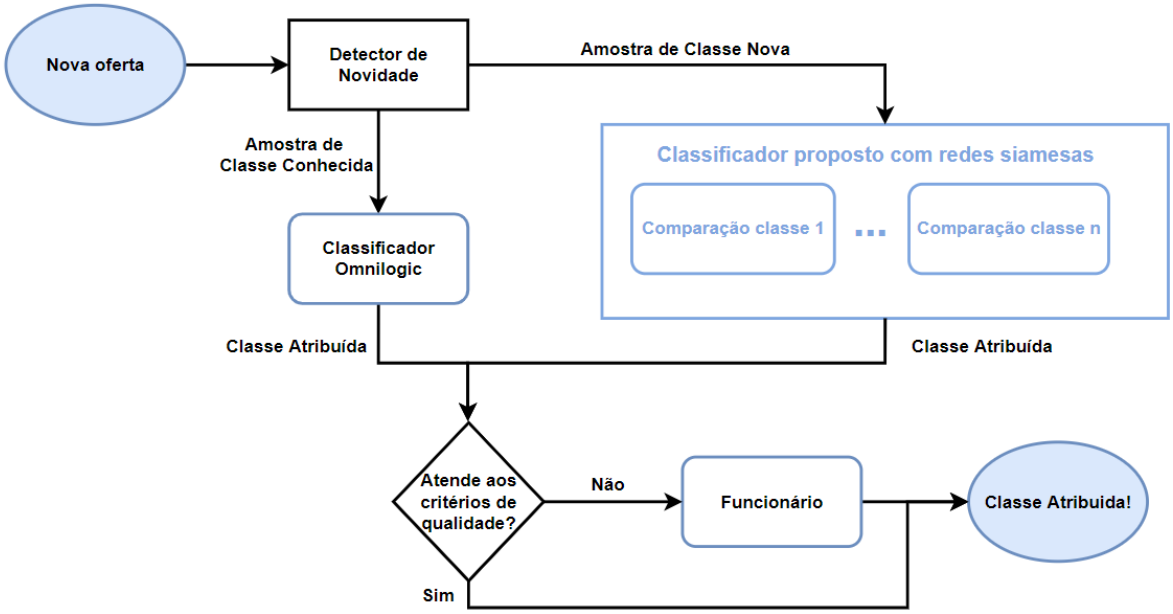
3.4 Solução proposta

A *Omnilogic* já possui um classificador para classificação de novas ofertas cadastradas em mais de 400 classes. Porém, caso seja criada uma classe nova, é necessário o reprojeto (treino, teste e validação) desse classificador, o que é um processo custoso, já que o modelo usado é complexo dado o número de classes envolvidas e demanda muito tempo de processamento. Diante disso, a solução proposta para evitar o reprojeto do classificador da empresa a cada oferta de uma nova classe envolve o uso de outro classificador em paralelo ao classificador já existente, para classificação das amostras de novas classes que não sejam conhecidas do classificador já implementado. A estrutura considerada para a solução proposta está representada na Figura 3.1.

Um ponto importante é que, neste trabalho, não foi necessário implementar a etapa para detecção de novidade, pois a *Omnilogic* já desenvolveu seu próprio detector de novidade. Assim sendo, todas as amostras que passarão pelo classificador proposto são, certamente, de alguma nova classe desconhecida pelo classificador da empresa já treinado. A extração de características do texto dos anúncios cadastrados é feita por um processo conhecido como *transfer learning*, aproveitando a estrutura neural já implementada pela rede profunda da empresa.

O processo ocorre da seguinte forma: uma nova oferta é cadastrada no *marketplace* e recebida pelo modelo da *Omnilogic*. São extraídas as características desta oferta pelo extrator em operação. Caso o detector de novidades indique que se trata de uma oferta de uma classe que não é conhecida pelo classificador da empresa, essa oferta é enviada para os classificadores aqui propostos. No caso de ser uma oferta de uma classe conhecida previamente pelo classificador da empresa durante seu treinamento, essa oferta é classificada, e o classificador também retorna um valor que representa a confiabilidade da classificação, que é um dos índices de qualidade

Figura 3.1 – Estrutura proposta para implantação dos classificadores



Fonte: Do Autor (2022)

utilizados. Caso sejam atingidos os níveis de qualidade especificados para a classificação ser considerada correta, a oferta é rotulada automaticamente.

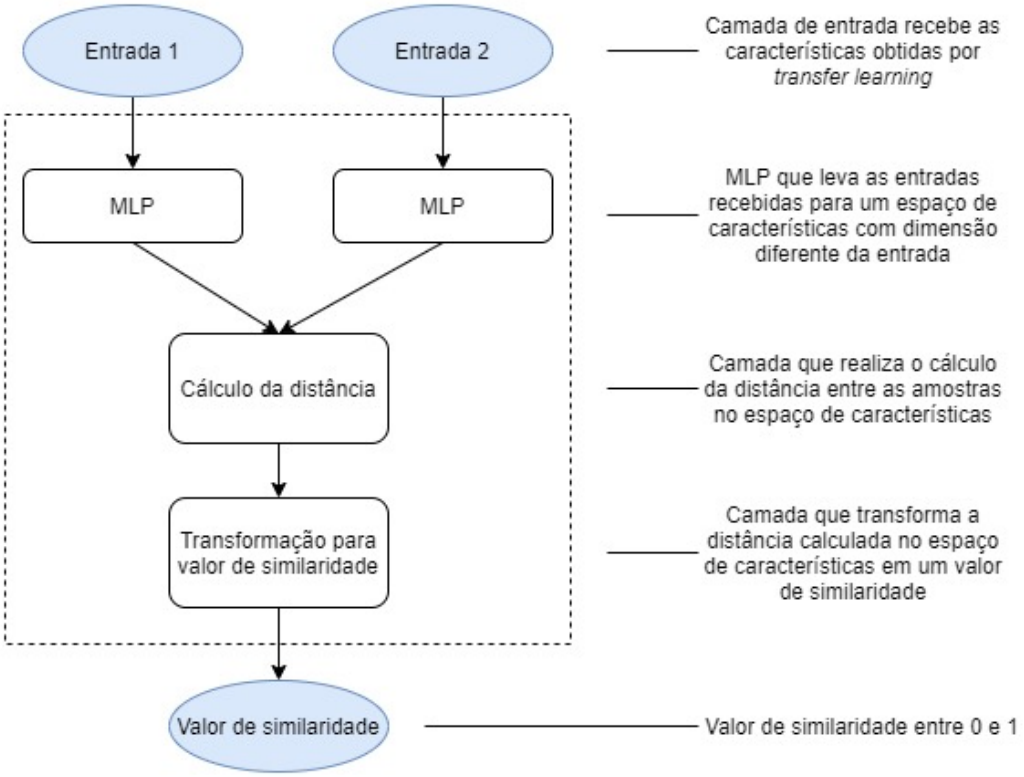
A confiabilidade da classificação é avaliada após a classificação automática pelo classificador da empresa ou pelos classificadores aqui propostos. Caso os níveis de qualidade não sejam atingidos, a oferta é enviada para um funcionário da empresa, que realiza manualmente a classificação das ofertas. Todo esse processo é ilustrado na Figura 3.1. Caso o funcionário detecte que a oferta pertence a uma classe ainda não cadastrada no *marketplace*, nem conhecida pelo classificador aqui proposto, essa amostra deve ser rotulada e incluída no banco de dados de treinamento do classificador proposto. Dessa forma, será necessário adicionar mais um classificador para reconhecer as próximas amostras desta nova classe.

3.4.1 Arquitetura do classificador proposto

Inicialmente, as redes siamesas foram projetadas com uma estrutura simples, composta por uma rede do tipo *Multi-Layer Perceptron* (MLP) como mostrado na Figura 3.2, que ilustra as camadas da rede e o comportamento de cada camada.

A camada de entrada recebe as características, cujo número pode variar de acordo com a base de dados utilizada. Em seguida, tem-se a MLP, que inicialmente foi feita com uma camada intermediária, cujo número de neurônios foi variado. O número de camadas da MLP também

Figura 3.2 – Camadas da rede siamesa e suas funções



Fonte: Do Autor (2022)

foi variado, com mais camadas de diferentes números de neurônios para analisar o desempenho da rede. A próxima camada da rede siamesa faz o cálculo da distância vetorial entre as saídas das duas MLPs. A saída dessa camada alimenta uma camada composta por um único neurônio que calcula o valor de similaridade entre as entradas.

Para o cálculo da similaridade, é considerado o vetor de distância calculado no espaço de características, que é então transformado para o intervalo de 0 a 1. O valor de saída dessa camada corresponde à similaridade entre as amostras de entrada, sendo que quanto mais próximo de 1, mais similares são as amostras.

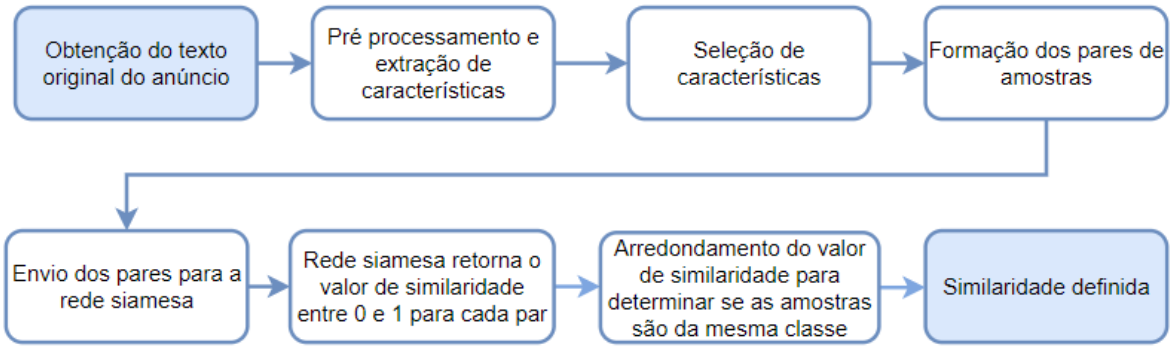
3.4.2 Etapas da classificação utilizando redes siamesas

Para o treinamento do modelo, todas as amostras da base de dados passaram por uma etapa de pré processamento, onde o texto original do anúncio foi transformado em características (valores numéricos) que representam toda a informação contida no texto. Essas características podem ser selecionadas usando os métodos descritos na Seção 2.7.3. Em seguida, são formados os pares de amostras, sendo que todo elemento da base será usado ao menos uma vez

como o primeiro elemento do par, e serão formados dois pares, sendo um par em que o segundo elemento será uma amostra da mesma classe e outro par em que o segundo elemento será uma amostra de classe diferente, escolhida aleatoriamente.

Os pares de amostras são enviados para a rede siamesa, que determina o valor de similaridade, no intervalo de 0 a 1, para as amostras de cada par. Esse valor é arredondado de acordo com um limite estabelecido para que as amostras sejam consideradas da mesma classe e o valor de similaridade é definido como 0 ou 1, indicando que as amostras são da mesma classe ou não. O limite usado neste trabalho foi de 0,5. As etapas descritas para a definição de similaridade durante o treinamento do modelo estão dispostas no diagrama da Figura 3.3.

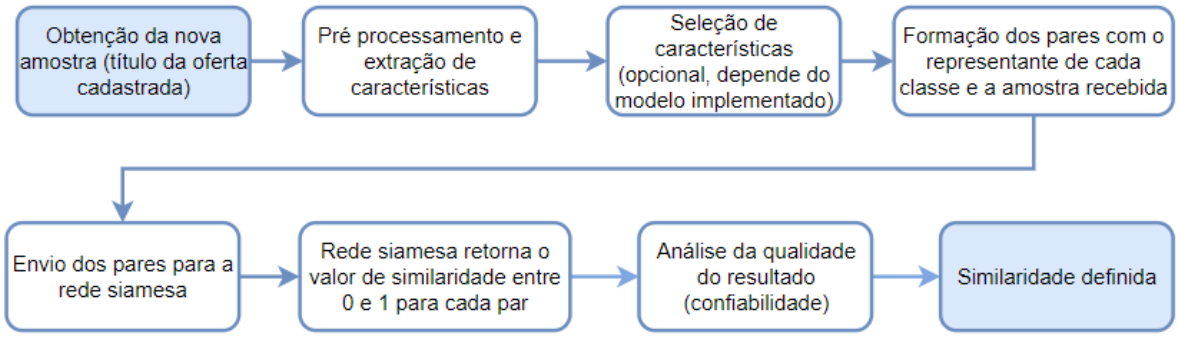
Figura 3.3 – Etapas para a definição de similaridade binária no treinamento do modelo



Fonte: Do Autor (2022)

Depois de ter o modelo treinado e implementado segundo a configuração mostrada na Figura 3.1, o processo para determinar a similaridade entre as amostras sofre algumas modificações, conforme mostrado na Figura 3.4. Ao receber um novo anúncio, o texto é extraído e passa pela etapa de pré processamento. Pode ser feita a seleção de características, caso esteja implantando um modelo que utilize as entradas neste formato. A formação dos pares de amostras que serão enviados para a siamesa é feita de maneira que a nova amostra seja pareada com o representante de cada classe disponível para análise. Esses pares são enviados para a rede siamesa, que calcula o valor de similaridade entre os elementos do par. O valor de similaridade é arredondado de acordo com o limite definido (valor de confiabilidade) e é definido se as amostras do par pertencem ou não a uma mesma classe.

Figura 3.4 – Etapas para a definição de similaridade binária no teste do modelo



Fonte: Do Autor (2022)

3.5 Abordagens utilizadas para escolha do representante

Após a rede siamesa ser treinada, ela pode ser usada como classificador para indicar se o par de amostras recebidos é similar ou não, ou seja, se pertencem à mesma classe. Durante a etapa de treinamento, não houve nenhuma forma de seleção do representante de cada classe, já que cada amostra foi utilizada como representante e o segundo elemento do par foi escolhido aleatoriamente.

Para as etapas de validação e teste do modelo, foram estudadas diferentes formas para selecionar o melhor representante, já que a rede siamesa faz a comparação de uma amostra de classe conhecida (amostra de referência ou representante da classe) com uma nova amostra que ainda não foi rotulada (amostra monitorada). Essa amostra de classe conhecida, quando escolhida aleatoriamente, pode não representar bem a classe, caso seja uma amostra que esteja nas bordas do *cluster* ou até mesmo um *outlier* da classe.

Na Abordagem *baseline* siamesa e Abordagem simples, a escolha do representante de cada classe foi feita de forma aleatória. Na Abordagem com centroide, o elemento utilizado como representante foi o centroide calculado para cada classe com os dados de treinamento. Ao utilizar o centroide, elimina-se o problema da escolha de representante, pois o centroide é calculado para cada classe usando os dados de treinamento. Uma alternativa seria considerar o representante da classe como a amostra mais próxima do centroide calculado para cada amostra.

Outra abordagem para o problema da escolha do melhor representante é o uso de uma estrutura de *ensemble* de representantes. São escolhidos n representantes para cada classe dentre as amostras da base de dados de treinamento. Assim é possível escolher mais de uma amostra que represente bem a classe. A rede siamesa avaliará a classe de amostra desconhecida, com-

parando com os representantes de cada classe e a similaridade do *ensemble* é decidida por voto majoritário. Os representantes podem ser escolhidos aleatoriamente ou usando algum algoritmo como o *k-Means*.

3.6 Análise de complexidade computacional do algoritmo

A complexidade computacional é definida em Desurvire (2009) como o número de operações matemáticas necessárias para a execução de um algoritmo. Considerando a estrutura proposta na Figura 3.2, com uma camada oculta, foi calculada a complexidade computacional do algoritmo utilizado na implementação das redes siamesas. Foi utilizada a notação a seguir: Ne é o número de entradas da rede, que pode variar de acordo com o número de características analisadas e Nn é o número de neurônios na camada oculta (primeira camada), que também pode ser variado.

O cálculo foi iniciado pela quantidade de multiplicações necessárias. Na primeira camada, os neurônios aplicam um peso para cada entrada, sendo assim, cada neurônio fará uma multiplicação para cada entrada recebida. Como é feito uma vez para cada elemento do par de entradas, esse valor é multiplicado por dois. Logo, na primeira camada são necessárias $Ne * Nn * 2$ multiplicações. Na segunda camada da estrutura, é feito o cálculo de distância ponto a ponto, não sendo feita nenhuma multiplicação. Na terceira camada da estrutura os neurônios aplicam os pesos em cada entrada, sendo que a entrada tem dimensão Nn. Logo, serão Nn multiplicações na terceira camada. No total são $(2 * Nn * Ne) + Nn$ multiplicações.

O mesmo raciocínio se aplica em relação a quantidade de adições necessárias. Na primeira camada, cada neurônio executa $Ne - 1$ somas mais 1 soma, que se refere ao *bias* adicionado. Isso é multiplicado por dois, já que é feito uma vez para cada elemento do par de entradas. Logo, na primeira entrada tem-se que são executadas $(Ne - 1 + 1) * Nn * 2$ adições. Simplificando, são $Ne * Nn * 2$ adições. Na segunda camada, são executadas Nn adições, pois é feita a subtração ponto a ponto dos dados. Na terceira camada são executadas $Nn - 1$ somas, mais 1 adição, que se refere ao *bias*. No total são $(2 * Nn * Ne) + Nn + Nn$ adições. Simplificando, são $2 * Nn * (1 + Ne)$ adições necessárias.

Na primeira camada oculta é utilizada a função tangente hiperbólica como função de ativação. Como são necessárias duas estruturas, são necessárias $2 * Nn$ tangentes hiperbólicas. Na última camada a função de ativação é a sigmoide. Como é somente um neurônio, é neces-

sária somente 1 sigmoide. As fórmulas para o cálculo da quantidade de operações totais para a estrutura apresentada na Figura 3.2 são apresentadas na Tabela 3.3.

Tabela 3.3 – Custo computacional do algoritmo de acordo com a operação realizada

Adições	**Multiplicações**	**Tangente Hiperbólica**	**Sigmóide**
$2 * Nn * (1 + Ne)$	$2 * Nn * Ne$	$2 * Nn$	1

Fonte: Do Autor (2022)

3.7 Avaliação dos resultados

A avaliação do desempenho dos classificadores foi feita utilizando as métricas apresentadas na Seção 2.7.2. Para desenvolvimento do(s) classificador(es) proposto(s) foi estudada a aplicação de técnicas de resolução de problemas de classificação *one/few shot learning*.

Além disso, foi usado *transfer learning*, uma vez que o pré processamento e a extração das características do texto do anúncio publicado no *marketplace* foi feito pelo extrator desenvolvido pela *Omnilogic*.

Foram variados a arquitetura da rede interna com diferentes números de camadas, e os parâmetros da rede siamesa, impactando diretamente na performance da solução proposta. Também foi analisada a influência dos diferentes representantes de cada classe, sendo que a escolha do representante de maneira aleatória foi considerada como a *baseline* de comparação. Para escolha do melhor representante de cada classe foram utilizadas as opções do cálculo do centroide de cada classe e estrutura de *ensembles* de representantes e variações dessas duas abordagens.

Foi realizado o teste de validação cruzada *leave one out* para a melhor abordagem, considerando toda a base de dados. Para isso, a base de dados foi dividida de forma que, em cada iteração do teste uma amostra fosse utilizada como base de teste, e todo o resto das amostras fosse utilizado como base de treino. O processo se repetiu até que todas as amostras da base tivessem sido utilizadas como base de teste.

O classificador KNN foi utilizado como *baseline* para comparação com os outros algoritmos utilizando o método de validação cruzada *leave one out*. A escolha do KNN foi feita considerando que é um algoritmo que dispensa treinamento, já que seu desempenho depende da métrica de distância utilizada (JADON; GARG, 2020).

4 RESULTADOS

4.1 Redução de características

Para visualizar os dados com altas dimensões as características foram transformadas usando o algoritmo *t-Distributed Stochastic Neighbour Embedding* (t-SNE) (MAATEN; HINTON, 2008). O objetivo do t-SNE é representar um conjunto de pontos que está no espaço de alta dimensão em um espaço de menor dimensão. Para isso, o algoritmo realiza diversas transformações dos dados em diferentes regiões.

Segundo Maaten e Hinton (2008), a primeira etapa é analisar a distância de cada ponto em relação aos demais. Essa análise é feita com uma distribuição de probabilidade que representa a similaridade entre os vizinhos, de forma que amostras semelhantes têm alta probabilidade de serem selecionadas. Em seguida, o t-SNE distribui os pontos no novo espaço de dimensão reduzida e calcula a distribuição de probabilidade para esse espaço. A próxima etapa é o cálculo e minimização da Divergência de *Kullback–Leibler* (KULLBACK; LEIBLER, 1951), que indica quão diferente são duas distribuições de probabilidade. Se a distribuição de probabilidade para os pontos no espaço de baixa dimensão for próxima à distribuição de probabilidade na alta dimensão, é possível obter *clusters* bem definidos.

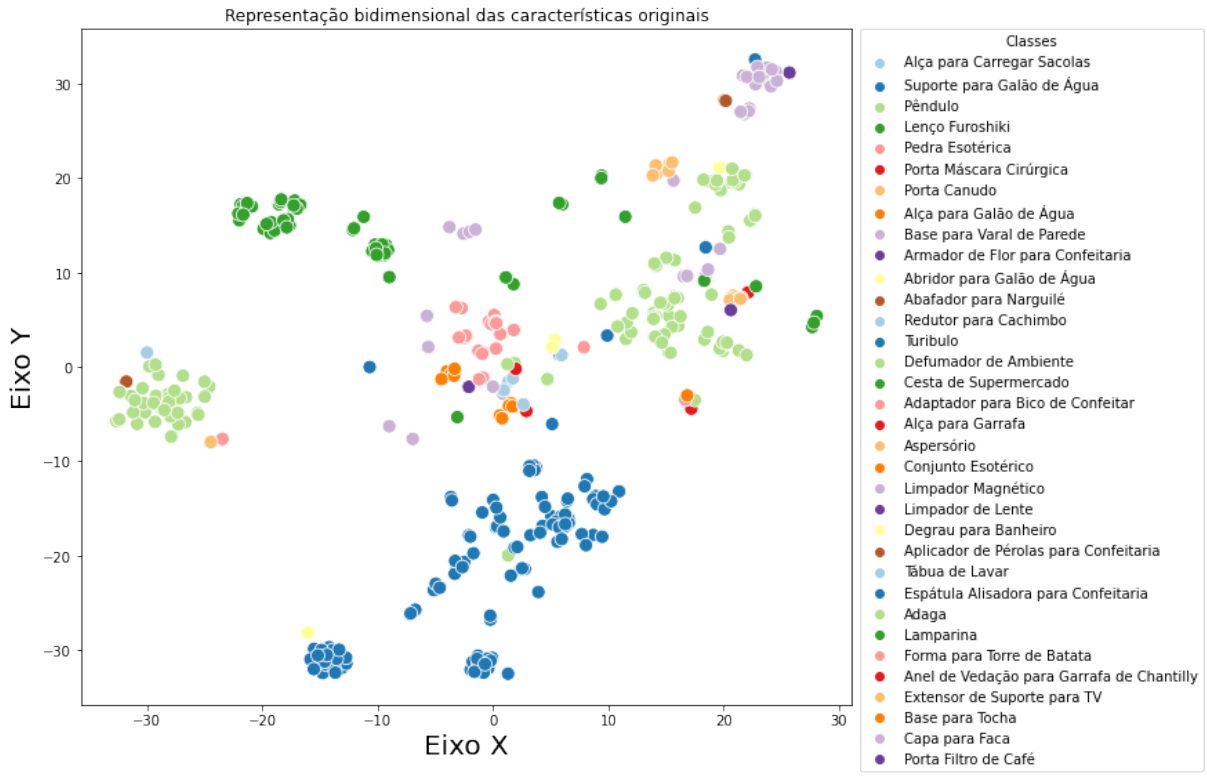
A Figura 4.1 mostra a distribuição das amostras da Primeira Base de Dados, em que cada amostra conta com 1200 características, em um espaço bidimensional.

Analisando a distribuição das amostras é possível perceber os *clusters* de classes espalhados pelo espaço, sendo alguns *clusters* mais definidos, como o da classe Suporte para Galão de Água (pontos azuis na parte inferior da imagem) e, em alguns casos, ocorre a sobreposição de *clusters*, mas as amostras de uma mesma classe aparecem próximas. Por exemplo, na classe Defumador de Ambiente (*cluster* verde claro à esquerda), apesar de ter sobreposição de algumas amostras de outras classes, as amostras da própria classe aparecem próximas uma das outras. Isso indica que as características obtidas por *transfer learning* isolam algumas classes mas outras não, mostrando que é um problema complexo de classificação.

A fim de explorar a possibilidade de redução de dimensionalidade da base de dados, foi feita a redução das características utilizando os métodos FDR combinado com correlação e PCA.

A seleção feita pelo PCA transformou as 1200 características originais em 307 características, mantendo mais de 99% de variância dos dados e gerando uma redução de 74,41% da

Figura 4.1 – Distribuição das amostras com 1200 características no espaço bidimensional



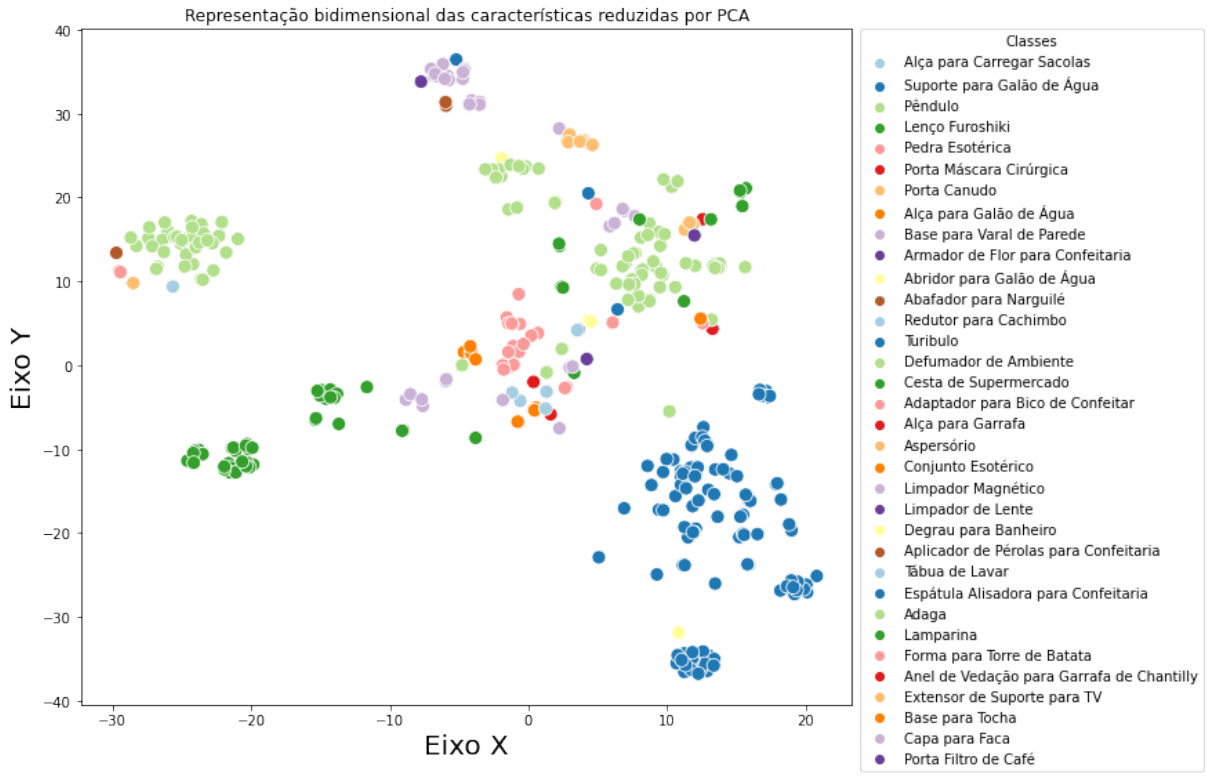
Fonte: Do Autor (2022)

dimensão dos dados analisados, o que mostra que há bastante redundância de características. A Figura 4.2 mostra a distribuição das amostras no espaço bidimensional após a redução de características feita pelo PCA.

Na abordagem com aplicação do FDR, foram selecionadas as características mais relevantes. Em seguida foi aplicada a correlação, para eliminar a redundância das características, o que causou uma redução de 60% nas características já selecionadas. A redução de características com o FDR e correlação levou a 308 características usando um limiar de correlação de 0,7. Portanto, para pares de variáveis com correlação superior a 0,7 (em módulo), a variável com menor FDR foi descartada. A Figura 4.3 mostra a distribuição das amostras no espaço bidimensional após a redução de características feita pela aplicação do FDR e correlação.

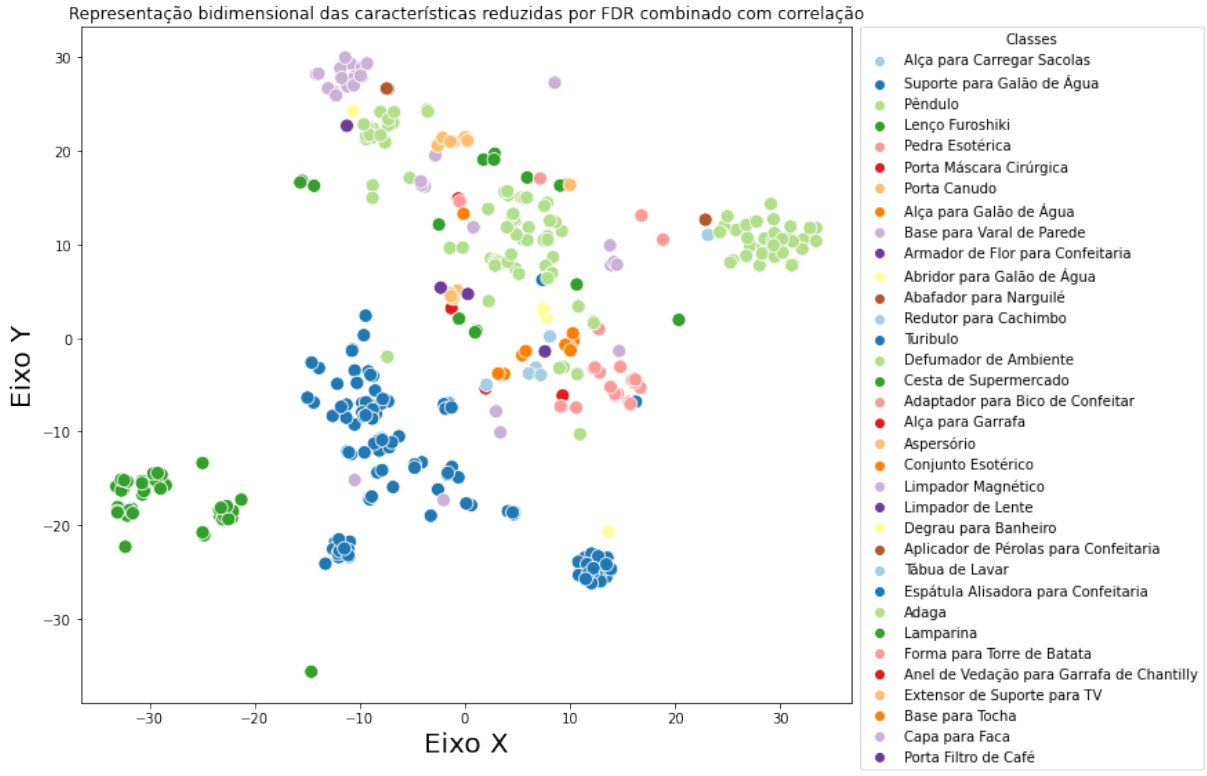
Considerando que a PCA efetua uma nova transformação de características a cada nova amostra da base de dados, essa pode ser uma operação computacionalmente custosa em uma base de dados com um grande número de amostras. Já o FDR combinado com análise de correlação é mais simples do ponto de vista computacional, já que é aplicado apenas na fase de projeto do método, que é executada *offline* e não requer processamento na fase operacional. Uma vez que a redução de dimensão manteve a boa separação dos *clusters*, as técnicas de

Figura 4.2 – Distribuição das amostras com 307 características no espaço bidimensional



Fonte: Do Autor (2022)

Figura 4.3 – Distribuição das amostras com 308 características no espaço bidimensional



Fonte: Do Autor (2022)

redução de características podem ser utilizadas em bases de dados maiores, como forma de diminuir a complexidade e tempo do treinamento dos modelos de classificação.

4.2 Projeto da rede siamesa

Durante o projeto do modelo, os parâmetros para a rede siamesa foram definidos experimentalmente usando a Primeira Base de Dados. Por simplicidade, foram utilizadas somente as classes com 20 ou mais amostras por classe, sendo elas: Suporte para Galão de Água (113 amostras), Pêndulo (60 amostras), Cesta de Supermercado (51 amostras), Defumador de Ambiente (49 amostras), Limpador Magnético (32 amostras) e Pedra Esotérica (20 amostras). A base de dados foi dividida em conjunto de treino e conjunto de teste, seguindo a proporção 70/30, sendo que o conjunto de treino ficou com 227 amostras e o conjunto de teste ficou com 98 amostras.

A base de dados de treinamento foi usada para o treinamento e ajuste dos parâmetros do modelo. Para validar o desempenho do modelo no treino da mesma forma que é analisado o desempenho na escolha de representantes, foi criado o conjunto de dados de validação, que foi composto por 50% das amostras da base de treino. As amostras utilizadas durante o teste são desconhecidas para o modelo treinado, porém as classes são conhecidas.

Inicialmente, o número de épocas foi variado no intervalo de 10 a 1000 épocas para analisar os efeitos de tal variação no desempenho do modelo. Foram testados todos os otimizadores da biblioteca *Keras* e aqueles que obtiveram os melhores resultados foram o Adam e o Adamax, que é um otimizador baseado no otimizador Adam. Para o otimizador Adam, o valor da taxa de aprendizagem (*learning rate*) foi variado experimentalmente de 0,0001 a 0,001 e o melhor valor foi de 0,0008, Com este valor, conseguiu-se melhores valores de acurácia e a rede aprendeu mais rápido a diferenciar as classes.

A estrutura adotada para a rede interna da rede siamesa foi de uma camada oculta, que será chamada estrutura MLP Camada Única (MLPCU), que foi a estrutura considerada como *baseline* de comparação. O número de neurônios da rede interna também foi variado de 5 a 600, tendo o melhor resultado com 20 neurônios na camada.

Ao diminuir a complexidade do modelo, o custo computacional também diminui, como pode ser observado ao analisar a Tabela 3.3. Os parâmetros que proporcionaram o melhor resultado para a rede siamesa, com a estrutura MLPCU estão detalhados na Tabela 4.1.

Tabela 4.1 – Parâmetros de configuração da rede siamesa

Parâmetros	**Valores otimizados**
Número de neurônios	20
Número de épocas	100
Taxa de aprendizagem do otimizador Adam	0,0008
Tempo de treinamento	58 minutos
Funções de ativação	tangente hiperbólica e sigmoide

Fonte: Do Autor (2022)

Após a definição dos melhores parâmetros para a rede siamesa com uma única camada oculta, foram iniciadas as alterações na estrutura da rede interna.

Inicialmente, foram adicionadas mais duas camadas ocultas na MLP com função de ativação *relu*. Cada camada reduziu pela metade o número de neurônios da camada anterior. Essa abordagem foi identificada como estrutura MLP Três Camadas (MLPTC). Ao alterar a estrutura foi necessário aumentar também o número de neurônios, pois a rede não conseguiu aprender bem com tão poucos neurônios como na estrutura MLP Camada Única. O número de neurônios foi escolhido para gerar um espaço de características com dimensão semelhante ao da MLPCU. Assim sendo, a configuração que gerou os melhores resultados foi com a primeira camada com 100 neurônios, a segunda camada com 50 neurônios e a terceira camada contando com 25 neurônios.

Em seguida, foi adicionada uma camada de *Dropout* antes da primeira camada da estrutura MLPTC e a taxa de aprendizagem foi aumentada para 0,001, como feito no trabalho de Sreepada e Patra (2020). Essa abordagem foi identificada como estrutura MLP *Dropout* na Camada Visível (MLPDCV). A camada de *Dropout* elimina, aleatoriamente, a contribuição de alguns neurônios ocultos na rede, sem modificar os neurônios de entrada e saída (SREEPADA; PATRA, 2020).

Outra abordagem foi adicionar uma camada de *Dropout* entre a primeira e a segunda camada oculta da estrutura MLPTC, e a taxa de aprendizagem usada foi de 0,001. Esta abordagem foi identificada como estrutura MLP *Dropout* na Camada Oculta (MLPDCO).

Também foi testada a estrutura MLP Quatro Camadas (MLPQC), com quatro camadas ocultas na MLP, sendo que a primeira camada ficou com 100 neurônios, a segunda camada com 50 neurônios, a terceira e quarta camadas contaram com 25 neurônios para manter a dimensão da saída compatível com as outras configurações.

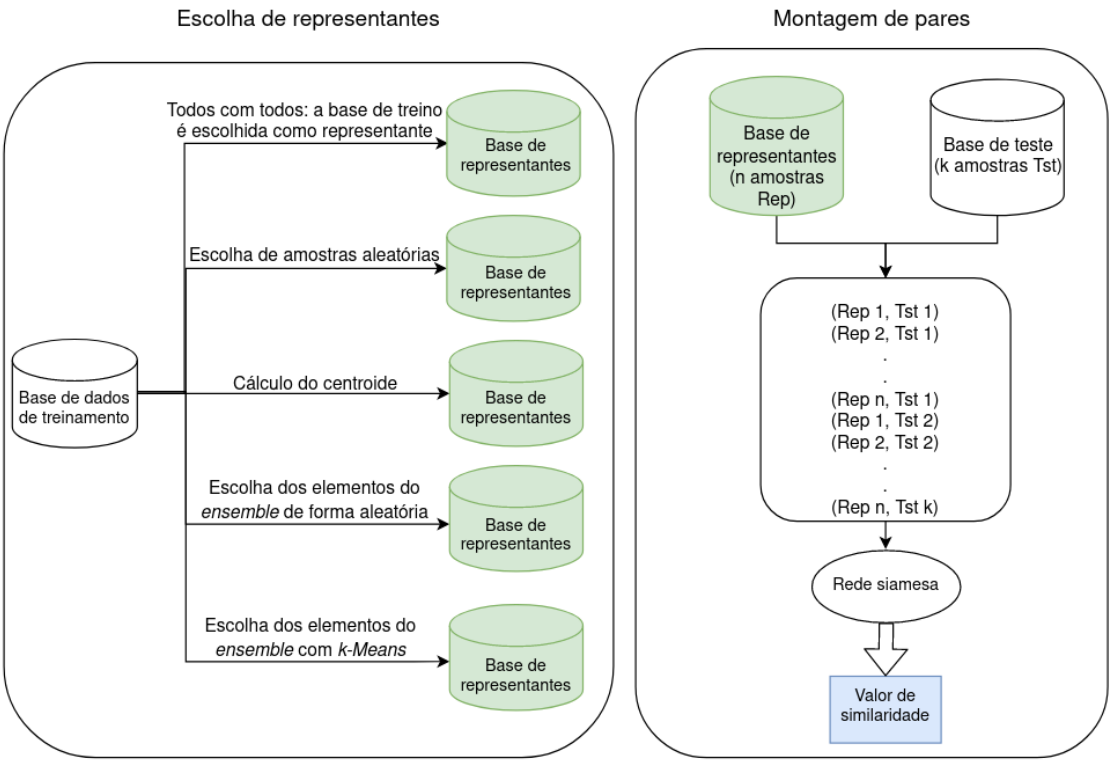
4.3 Escolha do melhor representante de cada classe

Um dos desafios no uso de redes siamesas consiste na escolha do representante de cada classe que será usado como base para comparação das novas amostras. Caso seja escolhido um elemento que se comporte como um *outlier* da classe, estando distante das outras amostras, ou que não represente bem os demais elementos da sua classe, o modelo pode ter um resultado precário. Nesta dissertação foram propostas algumas estratégias para escolha do melhor representante de cada classe, visando melhorar o desempenho do modelo proposto.

O modelo foi treinado previamente e testado com os diferentes métodos de escolha do melhor representante para avaliar a importância desta escolha no desempenho da rede siamesa. Cada uma das estruturas apresentadas na Seção 4.2 foram treinadas e, após ter o modelo treinado, foram feitos a validação e o teste com os diferentes métodos de escolha para representantes.

Os pares de validação e teste foram montados de forma que, para cada representante (elemento de referência), cada amostra da base de teste ou validação foi colocada como elemento monitorado. Nas seções a seguir são mostradas opções de escolhas de representantes para a rede siamesa. A Figura 4.4 ilustra as escolhas de representantes e a montagem dos pares.

Figura 4.4 – Diferentes métodos da escolha de representantes e montagem dos pares



Fonte: Do Autor (2022)

O algoritmo KNN foi escolhido para construção de *baseline* para comparação com a implementação *few-shot learning* das redes neurais siamesas. A utilização do algoritmo KNN foi feita considerando que ele não necessita de um treinamento prévio e possibilita resultados rápidos, além de apresentar bons resultados de acurácia para a tarefa de classificação de novos produtos com poucas amostras. O algoritmo KNN apresentou acurácia de 92,85%, sendo usada a base de teste como elementos a serem rotulados e a base de treino como os vizinhos disponíveis para escolha.

4.3.1 Todos com todos

O primeiro teste foi feito de forma que cada amostra do conjunto de treino foi usada como elemento de referência (representante) uma vez, sendo todas as outras amostras usadas como elemento monitorado. O processo foi repetido até que todas as amostras fossem usadas como elemento de referência. A acurácia média do modelo é medida observando se a rede siamesa previu corretamente o valor de 0 ou 1, tendo como alvo o valor calculado para o par de teste montado e como limite de arredondamento 0,5. Os resultados estão dispostos na Tabela 4.2 para cada estrutura testada e o melhor resultado se encontra destacado em negrito.

Tabela 4.2 – Resultados do teste com diferentes estruturas para representante escolhido usando método de teste todos contra todos

Estrutura	Acurácia Validação (%)	Acurácia Teste (%)
MLPCU	95,22 +- 6,09	96,97 +- 4,42
MLPTC	97,00 +- 4,11	97,79 +- 2,90
MLPDCV	96,04 +- 5,22	97,54 +- 3,19
MLPDCO	**97,06 +- 3,29**	**98,17 +- 1,86**
MLPQC	96,45 +- 3,94	97,70 +- 2,31

Fonte: Do Autor (2022)

Com os resultados acima é possível perceber que o modelo tem um bom resultado e aprendeu a definir se uma amostra pertence ou não a uma determinada classe. Porém, não é feita a escolha de representante, já que cada amostra foi usada uma vez como elemento de referência. Apesar de ter um resultado com quase todas as estruturas acima de 97%, esta não é uma abordagem viável para a aplicação tratada neste trabalho, uma vez que o classificador aqui proposto lidará com alto volume de dados. As escolhas de representantes a seguir trazem um ganho ao diminuir o custo computacional já que são necessárias menos comparações e análises pela rede siamesa.

4.3.2 Escolha aleatória

A primeira estratégia, e a mais simples, foi a escolha aleatória de uma amostra de cada classe da base de treinamento para ser a representante usada como elemento de referência no par de amostras enviado para a rede siamesa. Os pares de amostras foram montados de forma que todas as amostras da base de teste fossem usadas como elemento monitorado do par.

Para este caso, os testes foram repetidos 20 vezes, sendo que em cada vez foram escolhidos novos representantes (elemento de referência) de forma aleatória. A acurácia de validação e teste apresentadas na Tabela 4.3 são a média e desvio padrão da acurácia de todas as execuções.

Tabela 4.3 – Resultados com diferentes estruturas para representante escolhido aleatoriamente

Estrutura	Acurácia Validação (%)	Acurácia Teste (%)
MLPCU	88,97 +- 3,77	93,88 +- 1,98
MLPTC	91,44 +- 3,51	96,35 +- 0,95
MLPDCV	**93,19 +- 3,14**	**97,70 +- 0,41**
MLPDCO	92,22 +- 2,84	97,07 +- 0,41
MLPQC	91,53 +- 2,50	96,32 +- 0,63

Fonte: Do Autor (2022)

Os resultados obtidos com a escolha de representantes aleatória superam os resultados obtidos com o KNN. Entretanto, é necessário cautela pois a estratégia de escolha aleatória dos representantes não garante que os representantes escolhidos são aqueles que melhor representam cada classe. Por exemplo, caso seja escolhido um representante que esteja na borda do *cluster*, o modelo pode ter um desempenho ruim. Além disso, é importante ressaltar que para estes testes é possível que a mesma amostra tenha sido escolhida aleatoriamente como representante mais de uma vez.

4.3.3 Centroide

Considerando que as amostras estão distribuídas em *clusters*, uma estratégia para a escolha do melhor representante foi o uso do centroide de cada classe. Para cada classe, as amostras utilizadas como representante foram construídas pelo cálculo da média das características das amostras de cada classe usadas no treinamento da rede siamesa. Para as classes que só possuíam uma amostra na base de treinamento, essa amostra foi considerada como sendo o centroide da classe.

Os resultados para a validação e teste com o método do centroide estão na Tabela 4.4.

Tabela 4.4 – Resultados com diferentes estruturas para representante gerado com o cálculo do centroide

Estrutura	Acurácia Validação (%)	Acurácia Teste (%)
MLPCU	96,93	98,29
MLPTC	**98,29**	**98,46**
MLPDCV	97,27	98,29
MLPDCO	97,61	98,12
MLPQC	96,59	97,78

Fonte: Do Autor (2022)

A alternativa de usar o centroide de cada classe como o melhor representante é, na verdade, uma maneira de evitar a escolha de uma amostra da classe como representante, pois essa abordagem não envolve nenhuma escolha e, sim, a criação de uma nova amostra para ser usada como representante. Porém, pode ser utilizada uma abordagem em que a amostra escolhida como representante da classe será aquela que está mais próxima do centroide calculado com os dados de treinamento do modelo.

Para usar as amostras disponíveis na base de treinamento, o representante foi escolhido como sendo a amostra mais próxima do centroide de cada classe. O centroide foi calculado com os dados de treinamento do modelo. A amostra mais próxima foi escolhida dentre as amostras do treinamento usando o KNN com um vizinho ($k = 1$). A montagem dos pares foi feita de forma que o elemento de referência foi a amostra mais próxima do centroide. Os elementos monitorados foram as amostras da base de validação e, posteriormente, da base de teste. Os resultados foram colocados na Tabela 4.5 e os melhores resultados se encontram destacados em negrito.

Tabela 4.5 – Resultados com diferentes estruturas para representante mais próximo do centroide de cada classe

Estrutura	Acurácia Validação (%)	Acurácia Teste (%)
MLPCU	95,57	97,44
MLPTC	96,59	97,61
MLPDCV	**97,61**	**98,46**
MLPDCO	96,93	97,95
MLPQC	97,61	98,12

Fonte: Do Autor (2022)

É possível perceber que os valores de acurácia para o caso do centroide como representante são maiores, estando acima de 97%, o que mostra que a escolha do representante influencia

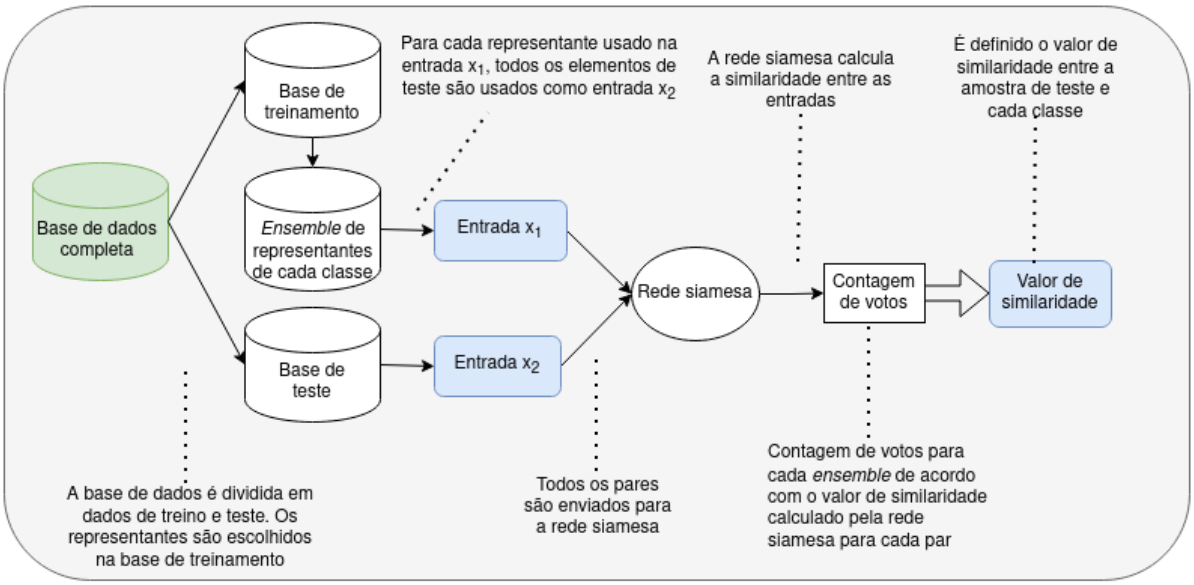
diretamente no desempenho do modelo. Vale ressaltar também que para classes com distribuição gaussiana o centroide será um excelente representante, porém isto não pode ser garantido para outras distribuições.

4.3.4 *Ensembles* de representantes

Outra estratégia foi o uso de uma estrutura de *ensembles* para lidar com o problema da escolha do melhor representante de cada classe. Utilizando um *ensemble* é possível escolher n representantes para cada classe e ter uma representação melhor da classe, considerando diversos pontos de sua distribuição no espaço.

No caso do *ensemble* de representantes, primeiro foram escolhidas aleatoriamente, na base de treinamento, as n amostras que seriam colocadas como representantes de cada classe e foi formado o *ensemble* para cada uma. Para cada representante do *ensemble*, a rede siamesa determina um valor de similaridade, que foi arredondado para 0 ou 1 segundo o limite de arredondamento de 0,5. Em seguida, foi feita a contagem de votos, sendo que a determinação do valor de similaridade para o *ensemble* foi feito por voto majoritário, ou seja, o valor de similaridade atribuído foi aquele que alcançou a maioria absoluta (metade do número de representantes n mais 1) de votos para o *ensemble*. A Figura 4.5 ilustra o processo de divisão da base, formação dos pares de entrada, cálculo da similaridade entre as entradas e contagem de votos para cada *ensemble*.

Figura 4.5 – Etapas para obter a similaridade usando estrutura de *ensembles*



Fonte: Do Autor (2022)

Foram executadas 20 rodadas de teste para observação da influência da escolha aleatória dos representantes de cada classe e os resultados da acurácia média e desvio padrão estão dispostos nas Tabelas 4.6 e 4.7, sendo os melhores resultados destacados em negrito. Foram considerados os casos com 3 e 5 representantes de cada classe.

Tabela 4.6 – Resultados do teste com diferentes estruturas para 3 representantes escolhidos aleatoriamente

Estrutura	Acurácia Validação (%)	Acurácia Teste (%)
MLPCU	95,32 +- 1,39	97,12 +- 1,05
MLPTC	96,61 +- 0,89	97,87 +- 0,47
MLPDCV	**97,39 +- 0,69**	**98,06 +- 0,52**
MLPDCO	96,71 +- 0,48	97,96 +- 0,24
MLPQC	96,15 +- 0,81	97,53 +- 0,44

Fonte: Do Autor (2022)

Tabela 4.7 – Resultados do teste com diferentes estruturas para 5 representantes escolhidos aleatoriamente

Estrutura	Acurácia Validação (%)	Acurácia Teste (%)
MLPCU	95,76 +- 0,80	97,62 +- 0,51
MLPTC	96,75 +- 0,65	97,93 +- 0,40
MLPDCV	**97,85 +- 0,63**	**98,43 +- 0,40**
MLPDCO	96,85 +- 0,58	98,04 +- 0,27
MLPQC	96,22 +- 0,74	97,72 +- 0,39

Fonte: Do Autor (2022)

Com o uso de *ensemble* o modelo alcança acurácias maiores do que com a escolha de representante aleatória, o que pode ser explicado pelo fato de que com mais representantes é maior a chance de escolher uma amostra que represente bem a classe. Em relação aos resultados obtidos com o uso do centroide, os resultados das Tabelas 4.6 e 4.7 são inferiores, porém são bem próximos aos melhores resultados encontrados utilizando o centroide. Porém, é importante lembrar que o centroide como representante é mais indicado no caso de classes com representação gaussiana, o que não pode ser garantido na aplicação proposta.

Em relação ao número de representantes, é possível perceber que a acurácia do modelo aumenta com o aumento do número de representantes. Entretanto, é necessário cuidado ao determinar n, o número ideal de representantes pois a base de dados é desbalanceada e a escolha de um número de representantes muito alto fará com que não seja possível aplicar esta solução

para as classes com menos de *n* elementos sem uma estratégia para criar novas amostras dessas classes que possam ser utilizadas como representantes.

4.3.5 *Ensembles* e *k-Means*

Mais uma estratégia foi usar um algoritmo de clusterização para escolher os *n* representantes que seriam usados no *ensemble*. O algoritmo escolhido foi o *k-Means* (MACQUEEN et al., 1967), que divide os dados em *n clusters*. Para cada agrupamento, é calculado um centroide considerando todos os pontos daquele grupo. O centroide de cada *cluster* foi considerado como um dos elementos de referência do *ensemble*. A ideia principal dessa estratégia é tentar gerar representantes que captem melhor o espalhamento dos dados no espaço de características e, portanto, os represente melhor.

Foram executadas 20 rodadas de teste, sendo calculados os representantes via *k-Means* em cada uma delas. Os modelos foram testados com 3 e 5 representantes. Os resultados foram colocados nas Tabelas 4.8 e 4.9, sendo que o desvio padrão foi desconsiderado neste caso por ter um valor muito próximo de zero.

Tabela 4.8 – Resultados do teste com diferentes estruturas para *ensemble* de 3 representantes escolhidos usando *k-Means*

Estrutura	Acurácia Validação (%)	Acurácia Teste (%)
MLPCU	93,19	95,74
MLPTC	95,57	97,44
MLPDCV	**97,95**	**98,46**
MLPDCO	97,27	98,29
MLPQC	96,25	97,61

Fonte: Do Autor (2022)

Tabela 4.9 – Resultados do teste com diferentes estruturas para *ensemble* de 5 representantes escolhidos usando *k-Means*

Estrutura	Acurácia Validação (%)	Acurácia Teste (%)
MLPCU	93,87	96,25
MLPTC	96,59	97,95
MLPDCV	**97,95**	**98,63**
MLPDCO	97,27	98,29
MLPQC	96,59	97,95

Fonte: Do Autor (2022)

Em comparação com os resultados obtidos com a escolha de representantes aleatórios para o *ensemble* com *k-Means*, os resultados mostrados nas Tabelas 4.8 e 4.9 foram superiores com a maioria das estruturas independente do número de representantes. Apesar da pouca diferença entre os resultados, o uso do *ensemble* combinado com *k-Means* pode ser uma boa opção de escolha de representantes, principalmente quando o número de amostras disponíveis para a escolha do representante for alto, já que um alto número de candidatos diminuiria a probabilidade de que a escolha pelo método aleatório resultasse em uma amostra que represente bem a classe.

4.3.6 Considerações

O KNN é o algoritmo mais simples testado pois dispensa treinamento. É possível perceber que a rede siamesa com o melhor representante de cada classe obtido de forma aleatória já superou os resultados obtidos com o KNN, de 92,85%. No pior cenário a escolha aleatória apresentou 93,88% de acurácia, uma melhora de 1,03% e, no melhor caso, com a estrutura MLPDCV, superou o KNN em 4,85%. Vale ressaltar que o KNN desempenha bem para poucas classes, enquanto que em um problema com mais classes, que é mais comum em comércio eletrônico é esperado uma redução no desempenho do KNN. No entanto, para poucas classes, o KNN continua sendo um bom candidato.

Diante disso, e pensando em formas de melhorar o desempenho da rede siamesa, foram testadas diferentes opções de representantes, sendo elas o centroide e o *ensemble* de representantes. O centroide como melhor representante teve o desempenho superior a 98% na maioria das estruturas testadas.

As estruturas com as camadas de *DropOut* apresentaram os melhores resultados, sendo que a MLPDCV teve resultados melhores com o uso do centroide e na maioria das implementações com *ensemble*.

A estrutura MLPQC apresentou um desempenho inferior em diversos experimentos, o que levanta a discussão sobre a melhor estrutura da rede interna. Não é garantido que com mais camadas ocultas haverá um ganho de desempenho e ainda pode haver um gasto maior com servidores, uma vez que aumentando a complexidade computacional, o algoritmo pode necessitar de um tempo maior de treinamento.

Vê-se a importância da escolha de representantes para serem usados como elemento de referência, nas estruturas de *ensemble*, que são bem próximos ao centroide e superaram

o KNN independente da estrutura utilizada. Em relação ao número de representantes, houve um aumento da acurácia ao aumentar de 3 para 5 representantes escolhidos aleatoriamente. Porém, ao utilizar uma métrica para escolha de representantes como o algoritmo *k-Means*, não foi verificado o mesmo aumento de desempenho em todas as estruturas. Isso pode ser causado pelo fato de as características extraídas terem boa qualidade e a rede siamesa fazer uma boa separação do espaço entre as classes, não sendo necessários mais do que 3 representantes por classe para alcançar o mesmo resultado.

4.4 *Leave one out*

Para determinar a capacidade de generalização do modelo foi executado o teste *leave one out*, em que, em cada iteração, uma amostra foi retirada da base de dados para ser usada como teste. A rede siamesa foi então treinada com pares montados com todos os outros dados. Para o teste, os pares foram montados de forma que a amostra que foi retirada da base de dados foi o elemento monitorado e o representante ou elemento de referência foi escolhido da mesma maneira que descrito na Seção 4.3, considerando a base de dados de treino em cada iteração.

Foram utilizadas somente as classes com mais de 3 amostras por classe da Primeira Base de Dados para implementar a escolha de representantes. Para os casos do *ensemble* com 5 representantes, foram usadas somente as classes com mais de 5 amostras por classes, já que o modelo necessita que tenha mais de n amostras, sendo n o número de representantes escolhido.

A estrutura utilizada foi a de redes siamesas MLPDCV, que foi a estrutura que apresentou os melhores resultados para os diferentes métodos de escolha de representantes. Em cada iteração, foi calculada a similaridade entre os pares de teste e contabilizada a acurácia do modelo. Ao fim das execuções, foram calculadas a média e desvio padrão considerando a acurácia de todas as execuções do modelo, que foram colocados na Tabela 4.10, em que RS se refere à Rede Siamesa.

Também são exibidos os resultados para o teste *leave one out* usando o KNN, que foi feito de forma que as amostras do conjunto de treino foram as amostras consideradas como os vizinhos. As amostras do conjunto de teste foram as amostras que precisavam ser rotuladas.

Tabela 4.10 – Resultados do teste *leave one out* com diferentes métodos para estrutura MLPDCV

Algoritmo	**Número de amostras**	**Acurácia (%)**
KNN	362	96,06 +- 19,44
RS Representante Aleatório	362	85,99 +- 13,07
RS com Centroide	362	**98,11 +- 11,61**
RS *Ensemble* 3 Representantes	362	94,00 +- 20,88
RS *Ensemble k-Means* 3 Representantes	362	97,84 +- 23,67
RS *Ensemble* 5 Representantes	339	97,50 +- 15,92
RS *Ensemble k-Means* 5 Representantes	339	98,02 +- 27.07

Fonte: Do Autor (2022)

Analisando os resultados dispostos na Tabela 4.10 é possível perceber que a rede siamesa com o centroide como melhor representante de cada classe alcançou o melhor resultado dentre as abordagens testadas, sendo o resultado destacado em negrito. O alto desvio padrão observado em todos os resultados pode ser explicado por ter diferentes modelos em cada iteração e cada modelo ter sido treinado com pares gerados de forma aleatória e chegado aos melhores pesos para aquele conjunto de dados. Além disso, como todas as amostras são usadas como representantes, o desempenho do modelo pode cair, já que nem toda amostra representa bem a sua classe. O fato dos representantes serem escolhidos de maneira aleatória em algumas das abordagens (escolha aleatória e *ensemble* escolhido de maneira aleatória) também contribui para que o desvio padrão seja elevado.

A abordagem com *ensemble* de 3 representantes não superou os resultados obtidos com o KNN, porém as abordagens com 3 e 5 representantes superaram os resultados da rede siamesa com o melhor representante escolhido de forma aleatória. Isso pode ser causado pelo fato de que algumas classes podem estar próximas a outras no espaço de características utilizado e, ao escolher aleatoriamente diversos pontos para o *ensemble* de representantes, alguns elementos ficam mais próximos dos representantes de outras classes do que dos representantes de suas próprias classes.

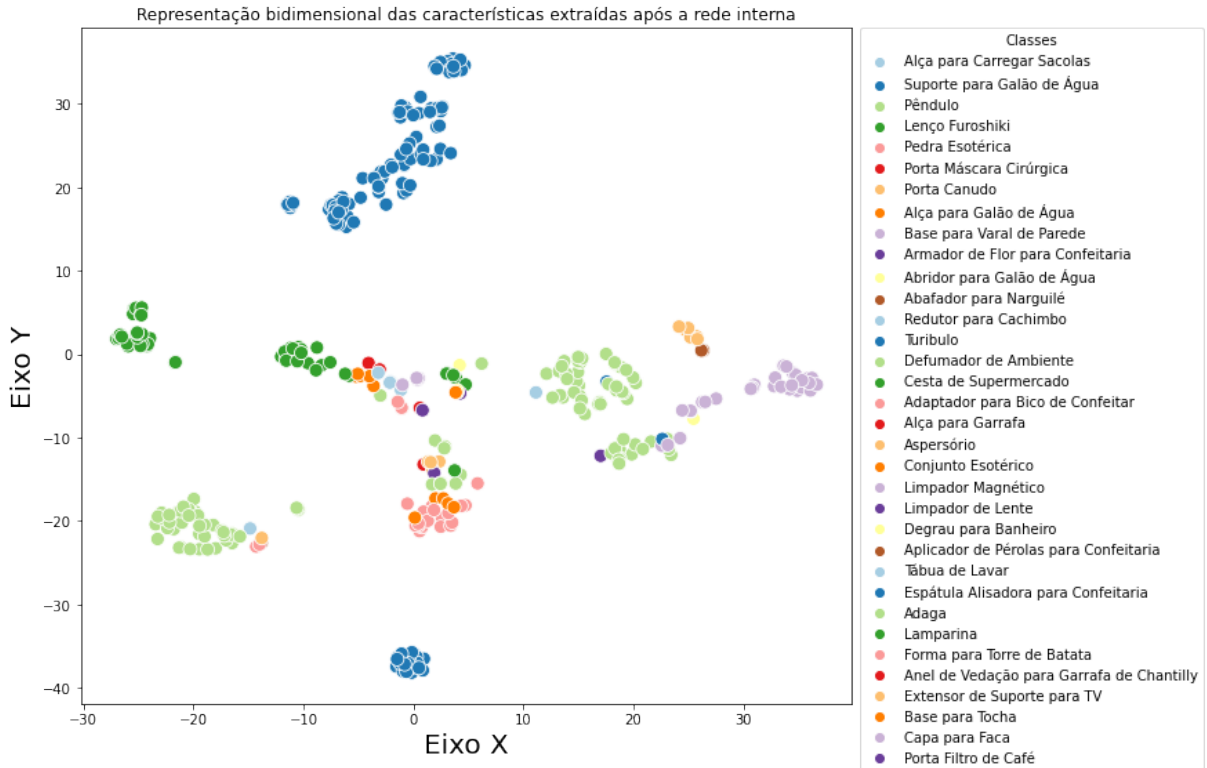
Nas abordagens de *ensembles* com 5 representantes, foram usadas menos classes do que nos cenários anteriores. A abordagem com *ensemble* com a escolha de representantes com *k-Means* obtiveram melhores resultados já que o algoritmo *k-Means* realiza a divisão das classes em *clusters*, e os centroides de cada *cluster* foram usados como representantes. O aumento do número de representantes melhorou o desempenho do modelo, se aproximando do desempenho registrado para o centroide.

4.5 Análise de features geradas pela rede interna

A rede interna da rede siamesa leva as amostras recebidas em suas entradas para um novo espaço de características com dimensão igual à da última camada de neurônios da rede interna. Neste novo espaço, as *features* são transformadas de forma que as amostras de mesma classe são aproximadas e as amostras de classes diferentes são distanciadas.

Para a estrutura MLPCU encontrada, o novo espaço de características é composto por 20 características. A Figura 4.6 mostra a representação destas novas características no espaço bidimensional obtido usando o método *t*-SNE.

Figura 4.6 – Distribuição das amostras após a rede interna MLPCU da rede siamesa no espaço bidimensional

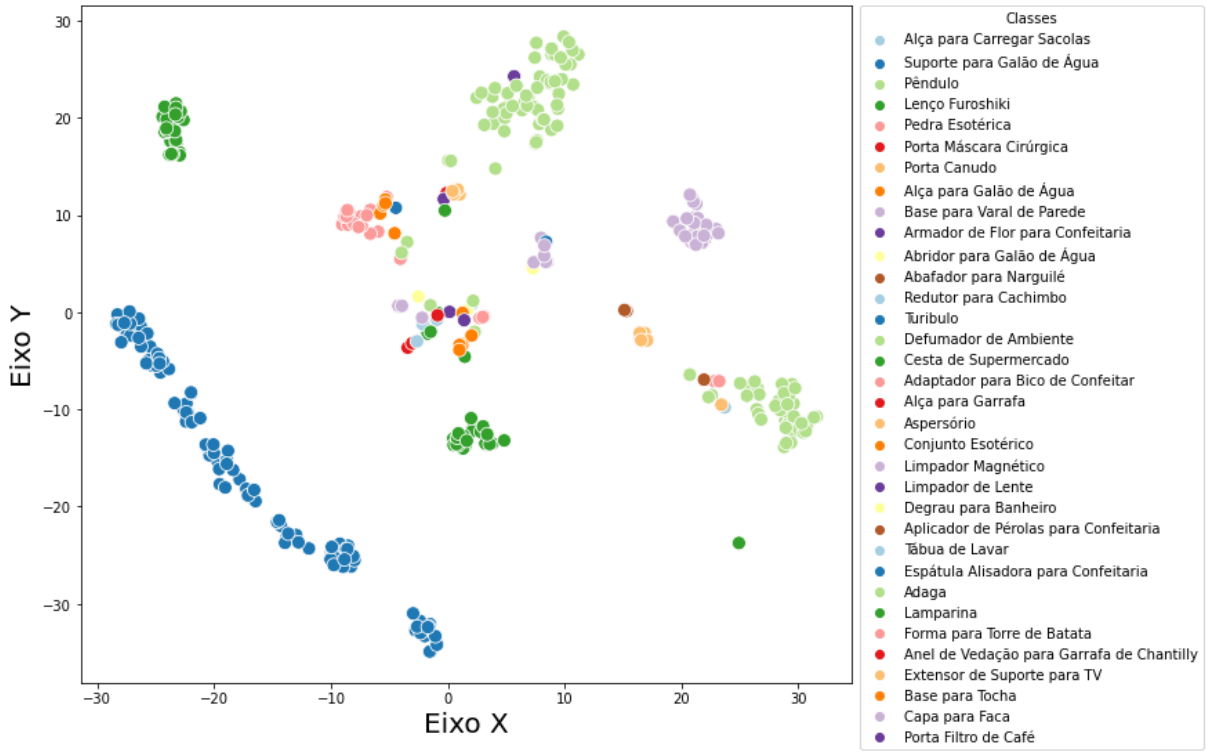


Fonte: Do Autor (2022)

Com a estrutura MLPDCV, que foi a estrutura que apresentou o melhor resultado nas comparações feitas na Seção 4.3, as 1200 características originais foram transformadas em 25 características. Estas foram então representadas no espaço bidimensional usando o algoritmo *t*-SNE e o resultado é mostrado na Figura 4.7.

Ao comparar as Figuras 4.6 e 4.7 com a Figura 4.1, que mostra as características originais no espaço bidimensional, é possível perceber que a rede interna é capaz de definir melhor os *clusters* de cada classe, fazendo com que fiquem com formatos mais homogêneos, o que faci-

Figura 4.7 – Distribuição das amostras após a rede interna MLPDCV da rede siamesa no espaço bidimensional



Fonte: Do Autor (2022)

lita o processo de classificação. Na Figura 4.7 as classes estão mais separadas e mais compactas do que na Figura 4.6, o que mostra o impacto da escolha da estrutura da rede interna, levando em conta a transformação do espaço de características.

Próximo ao ponto (0,0) do gráfico houve um aglomerado de amostras de classes diferentes, em sua maioria de classes que possuem somente uma amostra por classe. Isso pode ter sido causado pelo fato de, como não tem outras amostras da mesma classe, não foi possível realizar o cálculo de similaridade entre as amostras da classe e formar um *cluster* e por isso a amostra foi colocada próxima ao ponto de origem do gráfico.

4.6 Teste com a Segunda Base de Dados

Para análise da viabilidade da solução proposta em uma base de dados maior, com uma grande quantidade de classes, foi utilizada a Segunda Base de Dados. Nesta base de dados, há 452 classes, porém cada classe conta com no máximo 10 amostras. Além disso, as características não foram extraídas pelo mesmo extrator utilizado na Primeira Base de Dados, o que fez com que as amostras contassem com somente 1000 características.

Considerando que foi utilizado um extrator diferente e que as classes presentes são diferentes da Primeira Base de Dados, foi feito o treinamento do modelo novamente, para criar um modelo capaz de lidar com as características das amostras da base de dados em questão. De acordo com os testes feitos na Seção 4.3, a estrutura MLPDCV foi a que apresentou os melhores resultados e foi escolhida para os testes aqui descritos.

A Segunda Base de Dados foi dividida em conjunto de treino e teste seguindo a proporção 70/30, sendo usadas somente as amostras das classes que possuíam mais de 3 amostras, resultando em 2941 amostras para o treino e 1217 amostras para o teste. Foi testado o desempenho do KNN para criar uma base de comparação, sendo que o conjunto de dados de teste são os dados que se deseja rotular e os dados de treino são os dados disponíveis como vizinhos. O representante centroide foi calculado com a base de treinamento. Para a opção de representante escolhido aleatoriamente, por limitações de tempo de uso e recursos da plataforma *Colab*, os representantes aleatórios foram escolhidos somente 5 vezes.

Para o caso de 5 representantes, foram usadas somente as classes com mais de 8 representantes, sendo 2534 no treino e 853 no teste. Para os casos dos *ensembles* com representantes aleatórios, estes foram escolhidos somente 2 vezes. Para os casos em que o *ensemble* de representantes foi montado usando os centroides dos *clusters* encontrados com o algoritmo *k-Means*, o teste foi executado apenas uma vez pois foi constatado nas seções anteriores que o desvio padrão com esta abordagem é quase nulo. Os resultados dos testes com cada método de escolha de representantes são apresentados na Tabela 4.11, com o melhor resultado destacado em negrito.

Tabela 4.11 – Resultados do teste para Segunda Base de Dados com diferentes representantes para estrutura MLPDCV

Algoritmo	**Acurácia (%)**
KNN	81,81
RS Representante Aleatório	83,62 +- 7,41
RS com Centroide	**90,31 +- 1,10**
RS *Ensemble* 3 Representantes	84,71 +- 1,08
RS *Ensemble* 3 Representantes *k-Means*	87,33 +- 1,15
RS *Ensemble* 5 Representantes	85,33 +- 0,89
RS *Ensemble* 5 Representantes *k-Means*	86,70 +- 0,98

Fonte: Do Autor (2022)

Devido a limitação de memória RAM e tempo de uso da plataforma *Colab* e o fato de a Segunda Base de Dados só ter sido disponibilizada na etapa final dos experimentos, não foi

possível testar a capacidade de generalização do modelo com o teste *leave one out* assim como feito para a Primeira Base de Dados.

Ao lidar com um grande volume de dados, o modelo conseguiu realizar a distinção entre amostras da mesma classe e classes diferentes. O desempenho do modelo com rede siamesa foi superior ao KNN em todas as abordagens. A escolha de representante aleatória apresenta um alto desvio padrão, o que comprova a necessidade da aplicação de algum método para a escolha do melhor representante de forma a maximizar o desempenho do classificador.

O melhor resultado foi obtido usando o centroide como representante de cada classe. Os resultados de média das opções utilizando o *ensemble* de representantes não conseguiram superar o uso do centroide, porém, quando os valores são comparados considerando também o desvio padrão, esses resultados ficam bem próximos. Para classes com representação homogênea, o uso do centroide é interessante, porém se as classes estão mais dispersas no espaço de características, o *ensemble* pode ser uma opção mais interessante.

A matriz de confusão feita para o modelo considerando que o representante foi escolhido pelo centroide é mostrada na Figura 4.8. A classe 1 indica que o par analisado pertencia a mesma classe e a classe 0 indica que pertenciam a classes diferentes.

Figura 4.8 – Matriz de confusão para o centroide como melhor representante

		Classes preditas	
		1	**0**
Classes reais	**1**	1026	114
	0	48170	448870

Fonte: Do Autor (2022)

Foram analisados quase 500 mil pares de amostras e, destes, somente 1140 pares pertenciam a mesma classe. O modelo conseguiu classificar 1026 destes pares na classe correta e, quando as amostras eram de classes diferentes, o modelo classificou 448870 amostras corretamente, o que gera um valor de *recall* de 90,30%. Apesar do modelo classificar alguns pares

como pertencentes a mesma classe e, na realidade esses pares serem de classes diferentes, a quantidade de pares classificados erradamente neste caso e no caso em que as amostras pertencem a mesma classe e não o são, é proporcional a quantidade de pares de cada caso enviado para o modelo. Os experimentos aqui relatados foram feitos usando o valor de 0,5 para o limiar que determina se uma amostra pertence a mesma classe ou não. Este valor pode ser ajustado buscando um melhor resultado do modelo, sendo que o valor ótimo vai depender da confiabilidade desejada na classificação.

A acurácia neste caso foi de 90,31% e a precisão geral do modelo considerando o número de pares de cada caso foi de 99,75% e o índice *F1-Score* foi de 94,68%. Os resultados obtidos mostram que a aplicação da rede siamesa tem um bom desempenho para diferenciação de classes das diferentes amostras. A escolha de qual método para escolha de representantes será o melhor deve ser avaliada de acordo com as características das amostras utilizadas, o que pode ser abordado em trabalhos futuros, em que pretende-se explorar o pré-processamento empregado para extrair características antes de aplicar a metodologia desenvolvida.

5 CONSIDERAÇÕES FINAIS

O uso de inteligência artificial em aplicações de comércio eletrônico melhora os processos existentes, trazendo rapidez e segurança às compras *online*. Um dos meios de obter dados relevantes para o *E-commerce* é pelo uso do processamento de linguagem natural, um método computacional de obtenção e análise de informações em formato textual. Muitos sistemas utilizam algoritmos de aprendizado profundo, que precisam de grandes bases de dados para seu treinamento.

Entretanto, é muito comum a existência de bases de dados menores, mas que também precisam ser consideradas. Os algoritmos de aprendizado *one/few shot learning* são usados em bases de dados com classes desbalanceadas ou em bases com classes que apresentam poucos dados. Neste trabalho é apresentada a aplicação de algoritmos de aprendizado *few-shot learning* em dados obtidos por PLN, um diferencial em relação aos trabalhos encontrados na literatura que comumente aplicam tais algoritmos em problemas envolvendo a análise de imagens.

No processo de extração de características e classificação de produtos nas diferentes categorias utilizado atualmente pela *Omnilogic* só são aproveitadas as classes que possuem mais de 1000 amostras por classe. As amostras que não pertencem a este grupo precisam ser processadas por um funcionário. Esse processo é demorado e ainda pode ter valores errados devido a diversos fatores que podem levar a erro humano como falta de atenção ou o não conhecimento do produto a ser classificado. Em trabalhos futuros, é possível utilizar aplicações com inteligência artificial para lidar com o problema de rotulação errada, tais como o algoritmo *confident learning* (NORTHCUTT; JIANG; CHUANG, 2021), que lida com a qualidade destes rótulos, identificando erros e atuando com princípios de poda de dados ruidosos.

Os resultados aqui obtidos mostram que a aplicação do classificador com a rede siamesa na base de dados de plataformas de comércio eletrônico discutida neste trabalho é uma opção interessante para lidar com o problema de classes com poucos dados. A utilização do centroide como melhor representante obteve resultados em torno de 98% de acurácia, contra 93% obtido na utilização do representante aleatório. A escolha de qual método para escolha de representantes será o melhor deve ser avaliada de acordo com as características das amostras utilizadas. Porém, quando for atualizado o extrator de características será necessária a avaliação do melhor método de escolha de representante, uma vez que as classes podem ter diferentes distribuições pelo espaço. A utilização do centroide como representante em uma base com características

vindas de um extrator diferente resultou em 90,31% de acerto, contra 83,62% do representante aleatório e 81,81% do KNN.

Também foram testadas diferentes topologias para a rede interna da rede siamesa, todas elas tendo como base um MLP. Foram testadas redes com diferentes números de camadas e também a inclusão de uma camada de *DropOut* em diferentes camadas da rede. A estrutura com três camadas utilizando *DropOut* obteve melhor resultado dentre as opções testadas, aumentando o desempenho do modelo independente do método de escolha de representante adotado. Entretanto, o treinamento de uma rede mais complexa, com mais camadas, é uma alternativa também, apesar de demandar mais tempo, pois a rede precisará calcular e atualizar mais pesos no processo.

Considerando que as bases de dados utilizadas neste trabalho representam apenas uma fração dos dados que a empresa *Omnilogic* precisa processar, é evidente que um tempo menor de treinamento do modelo é um benefício, já que gera uma redução no gasto com o uso de servidores para treinamento do modelo. Além disso, a diminuição no número de características analisadas também impacta positivamente por diminuir a complexidade do modelo construído. Também podem ser estudadas estratégias para melhorar o desempenho do modelo, tais como a formação de pares de treino de forma a maximizar as diferenças entre as classes, ao invés da combinação aleatória das amostras. Outra alternativa seria o treino do modelo com parâmetros definidos por algoritmos de otimização. Além disso, pode ser utilizada uma estrutura de servidores que permita o treinamento por mais épocas ou usando mais dados.

Como trabalhos futuros, visando o melhor desempenho do modelo, uma proposta de escolha de representantes é o cálculo do centroide após o processamento das amostras pela rede interna da rede siamesa pois a rede siamesa leva as amostras para um espaço de características otimizado, onde é minimizada a distância entre as amostras de mesma classe e maximizada a distância entre amostras de classes diferentes. Assim, a rede interna seria usada como um extrator de características da base de dados recebida, economizando n transformações das amostras, sendo n o número de classes usadas como representantes. Após o processamento dos representantes, seria necessário apenas o processamento da nova amostras recebida e os cálculos da distância entre tais amostras para definição da similaridade entre elas.

Também podem ser desenvolvidos diferentes extratores de características para os dados das plataformas de vendas *online*. A redução de características por PCA e FDR combinado com correlação feita neste trabalho mostrou que ainda há muita redundância nos dados utilizados.

Um extrator que gere características com menor dimensão contribui para a redução da complexidade do classificador aqui proposto, o que pode gerar uma economia no uso de servidores por diminuir o tempo gasto com treinamento do modelo.

REFERÊNCIAS

ABDALHALEEM, A.; BARAKAT, B.; EL-SANA, J. Case study: Fine writing style classification using siamese neural network. In: **Conference: 2018 IEEE 2nd International Workshop on Arabic and Derived Script Analysis and Recognition (ASAR)**. London, UK: IEEE, 2018. p. 62–66.

AL-AMIN, M.; ISLAM, M. S.; UZZAL, S. D. Sentiment analysis of bengali comments with word2vec and sentiment information of words. In: **2017 International Conference on Electrical, Computer and Communication Engineering (ECCE)**. Cox's Bazar, Bangladesh: IEEE, 2017. p. 186–190.

ANDERSON, C. The long tail. **Wired**, v. 1210, 2004. Disponível em: <https://www.wired.com/2004/10/tail/>. Acesso em: 24 mar. 2021.

AYUSO, A.; SOLER, J. **Speech Recognition and Coding: New Advances and Trends**. Heidelberg, Alemanha: Springer Berlin Heidelberg, 2012. (Nato ASI Subseries F:). ISBN 9783642577451.

BAR-YOSSEF, Z.; KRAUS, N. Context-sensitive query auto-completion. In: **Proceedings of the 20th International Conference on World Wide Web**. New York, NY, USA: ACM, 2011. p. 107–116. ISBN 978-1-4503-0632-4.

BENAJIBA, Y. et al. Siamese networks for semantic pattern similarity. **2019 IEEE 13th International Conference on Semantic Computing (ICSC)**, p. 191–194, 2019.

BILGIN, M.; SENTURK, I. F. Sentiment analysis on twitter data with semi-supervised doc2vec. In: **2017 International Conference on Computer Science and Engineering (UBMK)**. Antalya, Turkey: IEEE, 2017. p. 661–666.

BISHOP, C. **Pattern Recognition and Machine Learning**. Manhattan, New York City, USA: Springer, 2006. (Information Science and Statistics). ISBN 9780387310732.

BROMLEY, J. et al. Signature verification using a "siamese" time delay neural network. **In Advances in neural information processing systems**, p. 737–744, 1994.

CHAVES, A. G. S. et al. Extração de entidades de produtos utilizando técnicas de few-shot learning. In: **Anais do XV Simpósio Brasileiro de Automação Inteligente**. Rio Grande, RS, Brasil: Sociedade Brasileira de Automática, 2021.

CHEN, L.; WANG, F. Preference-based clustering reviews for augmenting e-commerce recommendation. **Knowledge-Based Systems**, v. 50, p. 44–59, 09 2013.

CHOPRA, S.; HADSELL, R.; LECUN, Y. Learning a similarity metric discriminatively, with application to face verification. In: **005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)**. San Diego, CA, USA: IEEE, 2005. v. 1, p. 539–546 vol. 1. ISBN 0-7695-2372-2.

COELHO, L.; RICHERT, W. **Building Machine Learning Systems with Python - Second Edition**. Birmingham, Reino Unido: Packt Publishing, 2015. (Community experience distilled). ISBN 9781784392888.

CONSULT, M. **National Tracking Poll #200394, March, 2020**. USA, 2020. Disponível em: <https://morningconsult.com/wp-content/uploads/2020/03/200394_crosstabs_CORONAVIRUS_CONTENT_RVs_v8_JB.pdf>. Acesso em: 13 dez. 2020.

DESURVIRE, E. **Classical and Quantum Information Theory: An Introduction for the Telecom Scientist**. Cambridge, Reino Unido: Cambridge University Press, 2009. ISBN 9781139476652.

DUDA, R.; HART, P.; STORK, D. **Pattern Classification**. Hoboken, New Jersey, USA: Wiley, 2012. ISBN 9781118586006.

EDOSIO, U. Z. Big data analytics and its application in e-commerce. In: **Conference: E-Commerce Technologies**. Bradford, West Yorkshire, England: University of Bradford, 2014.

FUKUNAGE, K.; NARENDRA, P. M. A branch and bound algorithm for computing k-nearest neighbors. **IEEE Trans. Comput.**, IEEE Computer Society, USA, v. 24, n. 7, p. 750–753, jul. 1975. ISSN 0018-9340.

GENTLE, J.; HARDLE, W.; MORI, Y. **Handbook of Computational Statistics: Concepts and Methods**. Manhattan, New York City, USA: Springer, 2012. (Springer Handbooks of Computational Statistics). ISBN 9783642215513.

GHANI, R.; FANO, A. E. Building recommender systems using a knowledge base of product semantics. **Accenture Technology Labs**, Chicago, IL, USA, 2002.

GHANI, R. et al. Text mining for product attribute extraction. **Special Interest Group on Knowledge Discovery and Data Mining Explorations**, v. 1, p. 41–48, 2006.

GONZáLEZ, S. et al. A practical tutorial on bagging and boosting based ensembles for machine learning: Algorithms, software tools, performance study, practical perspectives and opportunities. **Information Fusion**, v. 64, p. 205–237, 2020. ISSN 1566-2535.

GRAVES, A.; JAITLY, N. Towards end-to-end speech recognition with recurrent neural networks. In: **Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32**. Beijing, China: JMLR.org, 2014. (ICML'14), p. II–1764–II–1772.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. **Neural computation**, MIT Press, Cambridge, Massachusetts, EUA, v. 9, p. 1735–80, 12 1997.

JADON, S.; GARG, A. **Hands-On One-shot Learning with Python Learn**. Birmingham, Mumbai: Packt Publishing, 2020. v. 1. 1–136 p. ISSN 1098-6596. ISBN 9788578110796.

KANNAN, A. et al. Matching unstructured product offers to structured product specifications. **Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**, p. 404–412, 08 2011.

KARVELIS, P. et al. Topic recommendation using doc2vec. In: **2018 International Joint Conference on Neural Networks (IJCNN)**. Rio de janeiro, Brazil: International Neural Network Society - INNS, 2018. p. 8–13. ISSN null.

KIM, R. Y. The impact of covid-19 on consumers: Preparing for digital sales. **IEEE Engineering Management Review**, v. 48, n. 3, p. 212–218, 2020.

KIROS, R. et al. Skip-thought vectors. **Proceedings of the Twenty-ninth Conference on Neural Information Processing Systems**, NIPS Conference, Montréal Canada, abs/1506.06726, 2015.

KOCH, G.; ZEMEL, R.; SALAKHUTDINOV, R. Siamese neural networks for one-shot image recognition. **Proceedings of the 32nd International Conference on Machine Learning**, v. 37, 2015.

KOZAREVA, Z. Everyone likes shopping! multi-class product categorization for e-commerce. 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. 1329–1333.

KULLBACK, S.; LEIBLER, R. A. On Information and Sufficiency. **The Annals of Mathematical Statistics**, Institute of Mathematical Statistics, United States, v. 22, n. 1, p. 79 – 86, 1951.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. **Nature**, London, England, v. 521, p. 436–444, may 2015.

LEVENSHTEIN, V. I. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. **Soviet Physics Doklady**, v. 10, p. 707, fev. 1966.

LI, M. Y.; KOK, S.; TAN, L. Don't classify, translate: Multi-level e-commerce product categorization via machine translation. **Workshop on Information Technologies and Systems (WITS2018)**, Santa Clara, Califórnia, EUA, 2018.

LING, W. et al. Character-based neural machine translation. **Proceeding of the International Conference on Learning Representations (ICLR) 2016**, San Juan, Puerto Rico, 2015.

MAATEN, L. v. d.; HINTON, G. Visualizing data using t-sne. **Journal of machine learning research**, v. 9, n. Nov, p. 2579–2605, 2008.

MACQUEEN, J. et al. Some methods for classification and analysis of multivariate observations. In: OAKLAND, CA, USA. **Proceedings of the fifth Berkeley symposium on mathematical statistics and probability**. Oakland, CA, USA: Cambridge University Press, 1967. v. 1, n. 14, p. 281–297.

MARTIN, K. et al. A convolutional siamese network for developing similarity knowledge in the selfback dataset. In: **In Proceedings of the ICCBR 2017 Workshops**. Trondheim, Norway: ICCBR, 2017. p. 85–94.

MATSUMOTO, T. et al. Data analysis support by combining data mining and text mining. In: **6th IIAI International Congress on Advanced Applied Informatics (IIAI-AAI)**. Yonago, Tottori, Japan: CPS, 2017. p. 313–318.

MIKOLOV, T. et al. Efficient estimation of word representations in vector space. In: **International Conference on Learning Representations 2013**. Scottsdale, Arizona, USA: ICLR Workshop, 2013.

MIKOLOV, T. et al. Distributed representations of words and phrases and their compositionality. In: **Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2**. Red Hook, NY, USA: Curran Associates Inc., 2013. (NIPS'13, v. 2), p. 3111–3119.

MOHRI, M.; ROSTAMIZADEH, A.; TALWALKAR, A. **Foundations of Machine Learning**. Cambridge, Massachusetts, EUA: MIT Press, 2012. (Adaptive Computation and Machine Learning series). ISBN 9780262018258.

MUELLER, J.; THYAGARAJAN, A. Siamese recurrent architectures for learning sentence similarity. In: **Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence**. Phoenix, Arizona USA: Association for the Advancement of Artificial Intelligence, 2016.

NORTHCUTT, C. G.; JIANG, L.; CHUANG, I. L. Confident learning: Estimating uncertainty in dataset labels. **Journal of Artificial Intelligence Research**, AI Access Foundation, El Segundo, CA, United States, p. 1373–1411, 2021.

PENNINGTON, J.; SOCHER, R.; MANNING, C. Glove: Global vectors for word representation. In: **Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)**. Doha, Qatar: Association for Computational Linguistics, 2014. p. 1532–1543.

RISTOSKI, P. et al. A machine learning approach for product matching and categorization. **Semantic Web**, IOS Press, Amsterdam, Netherlands, v. 9, p. 707–728, 2018.

ROSENBLATT, F. The perceptron: A probabilistic model for information storage and organization in the brain. **Psychological Review**, v. 65(6), p. 386–408, 1958.

ROY, S. et al. Siamese networks: The tale of two manifolds. In: **2019 IEEE/CVF International Conference on Computer Vision (ICCV)**. Seoul, Republic of Korea: IEEE Computer Society, 2019. p. 3046–3055.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. **Nature**, London, England, p. 533–536, out. 1986.

SCHAFER, J.; KONSTAN, J.; RIEDL, J. E-commerce recommendation applications. **Data Mining and Knowledge Discovery**, v. 5, p. 115–153, 01 2001.

SILVA, F. C. e.; FERREIRA, D. D.; BARBOSA, B. H. G. Classificação few shot learning aplicada ao e-commerce: uma abordagem com redes siamesas e ensembles. In: **Anais do XXX Congresso da Pós-Graduação da Universidade Federal de Lavras**. Evento virtual, Brasil: Universidade Federal de Lavras, 2020.

SILVA, F. C. e. et al. Classificador fuzzy-genético aplicado ao processamento de linguagem natural. In: **Anais do XXIII Congresso Brasileiro de Automática**. Evento virtual, Brasil: Sociedade Brasileira de Automática, 2020.

SOUZA, A. de et al. Aplicação de redes neurais siamesas na autenticação de condutores. In: **Anais do 14º Simpósio Brasileiro de Automação Inteligente (SBAI)**. Ouro Preto, MG, Brasil: SBAI, 2019.

SREEPADA, R. S.; PATRA, B. K. Mitigating long tail effect in recommendations using few shot learning technique. **Expert Systems with Applications**, Elsevier, Amsterdam, Netherlands, v. 140, p. 112887, 2020. ISSN 0957-4174.

SUN, S.; LUO, C.; CHEN, J. A review of natural language processing techniques for opinion mining systems. **Proceedings of the 19th International Conference on Information Fusion**, v. 36, 11 2016.

SUTSKEVER, I.; VINYALS, O.; LE, Q. Sequence to sequence learning with neural networks. **Advances in Neural Information Processing Systems**, Red Hook, NY Curran, v. 4, 09 2014.

TAI, K. S.; SOCHER, R.; MANNING, C. D. Improved semantic representations from tree-structured long short-term memory networks. **Association for Computational Linguistics (ACL)**, 2015.

TIAN, W.; LI, J.; LI, H. A method of feature selection based on word2vec in text categorization. In: **Proceedings of the 37th Chinese Control Conference (CCC)**. Wuhan, China: IEEE, 2018. p. 9452–9455. ISSN 1934-1768.

TYLER, S. K.; TEEVAN, J. Large scale query log analysis of re-finding. In: **Proceedings of the Third ACM International Conference on Web Search and Data Mining**. New York, NY, USA: ACM, 2010. (WSDM '10), p. 191–200. ISBN 978-1-60558-889-6.

VELEZ, I.; RASCON, C.; FUENTES-PINEDA, G. One-shot speaker identification for a service robot using a cnn-based generic verifier. **Proceedings of the International Conference on Robotics and Automation (ICRA 2018)**, IEEE Robotics and Automation Letters, 09 2018.

WEINBERGER, K. Q.; SAUL, L. K. Distance metric learning for large margin nearest neighbor classification. **Journal of Machine Learning Research**, v. 10, p. 207–244, 2009.

WU, Y. et al. Google's neural machine translation system: Bridging the gap between human and machine translation. **Google Technical Report**, Google Research, 09 2016.

ZHU, N. et al. A novel coronavirus from patients with pneumonia in china, 2019. **New England Journal of Medicine**, Massachusetts Medical Society, v. 382, n. 8, p. 727–733, 2020.

ZONG, Z.; HONG, C. On application of natural language processing in machine translation. In: **3rd International Conference on Mechanical, Control and Computer Engineering (ICMCCE)**. Huhhot, China: IEEE Computer Society, 2018. p. 506–510.

APÊNDICE A – Produção científica

Ao longo do desenvolvimento deste trabalho foram produzidos artigos científicos que foram submetidos a diferentes congressos. Estes são listados a seguir:

a) Artigo "Classificador *Fuzzy*-genético aplicado ao processamento de linguagem natural" (SILVA et al., 2020), publicado no XXIII Congresso Brasileiro de Automática em 2020.

b) Artigo intitulado "Extração de Entidades de Produtos Utilizando Técnicas *Few-Shot Learning*" (CHAVES et al., 2021), publicado no XV Simpósio Brasileiro de Automação Inteligente em 2021.

c) Artigo "Classificação *few shot learning* aplicada ao *E-commerce*: uma abordagem com redes siamesas e *ensembles*" (SILVA; FERREIRA; BARBOSA, 2020) publicado no XXX Congresso da Pós-Graduação da Universidade Federal de Lavras.

d) Artigo submetido para aprovação no XXIV Congresso Brasileiro de Automática em 2022.

Classificador Fuzzy-genético aplicado ao processamento de linguagem natural

Fernanda C. e Silva * **Rafael H. de Sousa** *
Aleson G. S. Chaves * **Bruno H. G. Barbosa** *
Danton D. Ferreira *

* *Departamento de Automática,*
Universidade Federal de Lavras,
Lavras, MG, Brasil

Emails: fernanda.silva10@estudante.ufla.br,
rafael.sousa2@estudante.ufla.br, alesongsc@gmail.com,
brunohb@ufla.br, danton@ufla.br

Abstract: Opinion mining analyzes opinions and feelings about an entity, which can be a product, a service, a person, etc. With the increasing use of the Internet, sentiment analysis has become an essential approach to analyzing the large amount of data generated. This analysis makes it possible to draw a profile of consumers, being a tool to assist companies in creating campaigns and improving their products. Several methods have been developed for the automatic classification of data in text format. The objective of this work is to design and evaluate a fuzzy classifier for mining opinion and classifying the general feeling of texts. For this, a database containing product reviews from the Epinions.com website was used. Before performing the classification, it was necessary to pre-process the data and extract characteristics, using two different methods. This work also proposes the use of the genetic algorithm to determine the characteristics that will be used by the fuzzy algorithm for classification, in order to maximize the accuracy value. The results obtained show a better accuracy for the classifier using characteristics extracted via Word2Vec when compared to the polarity method. In addition, the results obtained with the proposed method using 5 characteristics extracted via Word2Vec are superior to those obtained in other methods using 200 characteristics.

Resumo: A mineração de opinião analisa as opiniões e sentimentos sobre alguma entidade, podendo ser um produto, um serviço, uma pessoa, etc. Com o crescente uso da Internet, a análise de sentimentos tornou-se uma abordagem essencial para analisar a grande quantidade de dados gerados. Essa análise permite traçar um perfil dos consumidores, sendo uma ferramenta para auxílio das empresas na criação de campanhas e melhorias de seus produtos. Vários métodos foram desenvolvidos para a classificação automática de dados em formato de texto. O objetivo desse trabalho é projetar e avaliar um classificador *fuzzy* para mineração de opinião e classificação do sentimento geral de textos. Para isso, foi utilizada uma base de dados contendo revisões de produtos do site *Epinions.com*. Antes de realizar a classificação, foi necessário fazer o pré-processamento dos dados e a extração de características, com dois métodos diferentes. Esse trabalho também propõe a utilização do algoritmo genético para determinar as características que serão usados pelo algoritmo *fuzzy* para classificação, de forma a maximizar o valor da acurácia. Os resultados obtidos mostram uma melhor acurácia para o classificador usando características extraídas via *Word2Vec* quando comparado ao método por polaridades. Além disso, os resultados obtidos com o método proposto utilizando 5 características extraídas via *Word2Vec* é superior aos obtidos em outros métodos utilizando 200 características.

Keywords: Natural Language Processing; Fuzzy Logic; Genetic Algorithms; Feature Extraction

Palavras-chaves: Processamento de Linguagem Natural; Lógica Fuzzy; Algoritmos Genéticos; Extração de Características.

1. INTRODUÇÃO

A área de Inteligência Artificial conhecida como Processamento de Linguagem Natural (PLN) estuda a capacidade e as limitações de uma máquina em entender a linguagem dos seres humanos (Zong and Hong (2018)). Com o crescente uso das mídias sociais na Internet, a mineração de opiniões se tornou uma abordagem essencial para analisar os dados, sendo utilizada em aplicações como precificação de produtos, previsão de mercado, previsão de eleições, inteligência competitiva, entre outras (Sun et al. (2016)).

Com a tendência das empresas disponibilizarem seus produtos para compra online, o aprendizado de máquina começou a ser aplicado para realizar o entendimento semântico de textos com o intuito de construir perfis de usuário e descobrir suas preferências, na expectativa de melhorar a recomendação de produtos e gerar uma melhor experiência de compra (Chen and Wang (2013)).

Muitos métodos de classificação foram desenvolvidos para a classificação automática de dados em formato de texto. As metodologias tradicionais de classificação como *Naive Bayes* (NB), *k-nearest neighbor* (KNN) e Máquina de vetores de suporte (SVM, do inglês Support Vector Machine) são utilizadas para realizar classificação de sentimentos em documentos, usando PLN (Li et al. (2018) e Vanaja and Belwal (2018)).

No artigo de Solangi et al. (2018) são revisadas as técnicas de PLN para mineração de opinião e análise de sentimentos. Já em seu trabalho, Tang et al. (2014) introduziram uma técnica de coleta de informações contextuais e de sentimentos das palavras, aprendendo a incorporação de palavras específicas de sentimentos. Eles aplicaram seu modelo para extrair sentimento de publicações realizadas no Twitter. Já Cui et al. (2006) trabalharam em análises de produtos online, classificando cerca de 100 mil revisões de produtos em duas classes principais: positiva e negativa.

No trabalho de Choudhary and Choudhary (2018) é feita a análise de opiniões sobre as mais recentes marcas de celulares. As opiniões foram coletadas diretamente do *Twitter*. Os resultados foram apresentados em gráficos e podem ser utilizados pelas marcas para melhorarem suas vendas. Já G. Chen and Xu (2016) observaram que um grande volume de vendas não gera necessariamente sentimentos positivos e vice-versa. A análise foi feita construindo-se perfis de consumidores de serviços de vendas *online*, de diferentes regiões. Esses trabalhos mostram como aplicações com PLN podem auxiliar empresas de comércio eletrônico.

No artigo de P. Pankaj and Soni (2019) é proposto um método geral para encontrar opinião contida em análises de produtos *online*, explorando a diferença nas estatísticas de duas compilações, um corpus específico de domínio e um corpus independente de domínio. Também discute técnicas e abordagens existentes para extração de características em análise de sentimentos e mineração de opinião.

No trabalho de Vanaja and Belwal (2018) foi feita a comparação entre a classificação realizada pelos algoritmos NB e SVM. Já Y. Liu and Shahbazzade (2018) compararam o algoritmo FL-SVM com os algoritmos NB, KNN e SVM em diferentes conjuntos de dados e mostrou que o FL-SVM pode alcançar a melhor precisão de classificação de sentimentos, aumentando de 1% a 3% quando comparada aos outros algoritmos.

Um sistema de inferência neuro-fuzzy adaptável (ANFIS) é um tipo de rede neural artificial baseada no sistema de inferência *fuzzy* Takagi–Sugeno. A lógica *fuzzy*, quando comparada com a lógica booleana, permite mais representações do conhecimento em um ambiente de incerteza e imprecisão, modelando as interações e relacionamentos entre as variáveis do sistema. (Cordón et al. (2001)). A lógica booleana somente permite que as variáveis tenham 0 ou 1 como valores lógicos. Já a lógica *fuzzy* permite qualquer número real entre 0 e 1.

Os algoritmos genéticos (AG) são métodos de otimização global que funcionam com base na sobrevivência do mais apto e nos mecanismos da seleção natural (Padmaja and Hegde (2019)). Diversos trabalhos utilizam algoritmos genéticos para otimização dos resultados em diferentes tipos de problemas de otimização (Chen and Dai (2020)), classificação (Mortezanezhad and Daneshifar (2019) e Shi and Xu (2018)), entre outros.

Um Sistema Fuzzy Genético (SFG) é um sistema híbrido no qual um sistema *fuzzy* é otimizado por um processo de aprendizado genético. Uma otimização possível é aquela em que um algoritmo genético é usado para ajustar os diferentes componentes de um sistema baseado em regras *fuzzy* (Cordón et al. (2001)).

No trabalho de Padmaja and Hegde (2019) foi desenvolvido um SFG em que as regras *fuzzy* são geradas usando o método de classificação ANFIS. A saída do classificador ANFIS é fornecida como entrada para o AG encontrar os valores ideais para regras *fuzzy*.

De maneira semelhante, o trabalho apresentado neste artigo propõe um SFG, porém com uma abordagem diferente, utilizando o AG para determinar os valores que serão usados pelo ANFIS para classificação, de forma a maximizar o valor da acurácia. O trabalho está organizado da seguinte forma: a Seção II apresenta a pesquisa de vários artigos envolvendo análise de sentimentos em áreas relacionadas ao comércio eletrônico. A seção III apresenta as etapas para construção do SFG proposto. Já a análise quantitativa e comparativa do sistema proposto está detalhada na seção IV. Por fim, a conclusão do trabalho de pesquisa com sugestões para futuros trabalhos estão descritas na seção V.

2. MATERIAIS E MÉTODOS

2.1 Base de dados

A base de dados utilizada contém revisões de produtos do site *Epinions.com*, que foi um serviço online criado em 1999, e oferecia avaliações de produtos feitas pelos próprios consumidores. O site foi desativado em 2014 e substituído pelo site *Shopping.com*. Apesar de as avaliações serem antigas, a estrutura textual é semelhante aos comentários encontrados hoje em diversos sites de compras *online*.

A base passou por um processo de balanceamento, para que contivesse o mesmo número de avaliações positivas e negativas. Foram selecionadas 691 postagens que recomendam automóveis Ford e 691 postagens que não recomendam automóveis Ford. No total, foram analisadas 1382 amostras. As avaliações analisadas estão em formato textual, em inglês. Os dados podem ser divididos em duas classes: Pos (críticas que expressam sentimentos positivos ou favoráveis) e Neg (críticas que expressam sentimentos negativos ou desfavoráveis).

2.2 Pré-processamento

Os dados foram extraídos de um arquivo, em que cada comentário estava em uma linha do arquivo. Considerando que textos são dados não estruturados e podem conter caracteres especiais e sinais de pontuação, é importante realizar o pré-processamento dos dados antes do início do processo de classificação. Os caracteres indesejados, tais como #, @ e / foram removidos.

2.3 Tokenização e transformação dos dados via extração de polaridades

A etapa de tokenização é utilizada para dividir cada amostra em *tokens* para o processo de análise e mineração de texto. Inicialmente, as amostras são segmentadas e são localizados os limites para formar os *tokens*. O limite foi determinado pelo início e fim de uma palavra. Para realizar esse processo, utilizou-se a biblioteca *Textblob*, que é escrita em *Python* e faz processamento de dados em formato textual.

Como nem toda palavra da frase expressa uma opinião, a biblioteca também foi usada para realizar a marcação gramatical de cada *token*, associando a classe gramatical como pronome, verbo, adjetivo, advérbio. Essa classificação é conhecida como *part-of-speech tag*.

Após essa etapa foram selecionados os adjetivos e advérbios, por serem as palavras relevantes para a análise. Para determinar a polaridade de cada comentário, foi analisada cada palavra restante, usando o dicionário SentiWordNet 3.0.

Para análise, cada palavra foi reduzida ao seu radical, removendo prefixos, sufixos e outros modificadores da palavra para aumentar a chance dessa palavra ser encontrada no dicionário. Para cada palavra são buscados seus *synsets*, que são os sinônimos da palavra analisada. O *synset* possui três valores, o de positividade, o de negatividade e o de objetividade.

Para cada palavra, é realizada a soma da parte positiva menos a soma da parte negativa de cada *synset*. Depois esse valor é dividido pela quantidade de *synsets* da palavra. O resultado dessas etapas é um vetor contendo os adjetivos e advérbios do comentário original e, para cada um, a classificação gramatical e o valor de polaridade calculado.

Os valores obtidos foram então combinados para gerar as 20 características listadas a seguir:

(1) Soma dos adjetivos positivos;
(2) Soma dos adjetivos positivos dividido pelo número de palavras extraídas do comentário original para classificação;
(3) Quantidade de adjetivos positivos extraídos do comentário;
(4) Soma dos adjetivos negativos;
(5) Soma dos adjetivos negativos dividido pelo número de palavras extraídas do comentário original para classificação;
(6) Quantidade de adjetivos negativos extraídos do comentário;
(7) Soma dos advérbios positivos;
(8) Soma dos advérbios positivos dividido pelo número de palavras extraídas do comentário original para classificação;
(9) Quantidade de advérbios positivos extraídos do comentário;
(10) Soma dos advérbios negativos;
(11) Soma dos advérbios negativos dividido pelo número de palavras extraídas do comentário original para classificação;
(12) Quantidade de advérbios negativos extraídos do comentário;
(13) Polaridade resultante para adjetivos;
(14) Polaridade resultante para advérbios;
(15) Quantidade de adjetivos positivos menos a quantidade de adjetivos negativos;
(16) Quantidade de advérbios positivos menos a quantidade de advérbios negativos;
(17) Valor do adjetivo mais positivo ou do mais negativo;
(18) Valor do advérbio mais positivo ou do mais negativo;
(19) Quantidade de palavras extraídas do comentário original;
(20) Quantidade de palavras do comentário original.

O processo é repetido para todos os comentários e a matriz de características obtida é salva em um documento *txt*, que é modificado posteriormente, para inserção da 21ª coluna, com o valor da polaridade de cada comentário.

2.4 Tokenização e transformação dos dados via Word2Vec

O *Word2Vec* é empregado, em geral, para *word embedding*. Ele é capaz de extrair conhecimento semântico dos textos, além de realizar a análise sintática e morfológica. O *Word2Vec* é eficiente em termos computacionais devido ao seu modelo ser redes neurais com poucas camadas, sendo eficientes para treinamentos em grandes conjuntos de dados textuais. Apresenta, a capacidade de ser treinado rapidamente e produz resultados com uma boa acurácia.

O *Word2Vec* tem a característica de mapear palavras que apresentam similaridade em posições próximas dos vetores, ou seja, possuem valores próximos de caracterização (Mikolov et al. (2016)).

Para a implementação do *Word2Vec* foi utilizada a linguagem *Python* e suas bibliotecas *open source* para aprendizagem de máquina como *pandas, numpy, matplotlib, seaborn, string e nltk*. Essas bibliotecas são comumente utilizadas para o desenvolvimento de projetos na área de inteligência artificial, tal como em aplicações de processamento de linguagem natural.

Após a etapa de extração de caracteres via *Word2Vec* seguiram-se as etapas de pré processamento, tokenização e *stemming*, de forma similar à extração de características por polaridade. Porém o *Word2Vec* utilizou as bibliote-

cas *NLTK e numpy* para realização destas operações. Já para a extração de características foi utilizada a biblioteca *gensim*. Esta implementação do *Word2Vec* permite a configuração do modelo em diversos parâmetros.

A arquitetura do *Word2Vec* utilizada foi do tipo *skip-gram*, que faz a previsão das palavras de contexto dada uma palavra de origem. Neste caso, a rede neural vai ter como entrada o vetor com uma palavra, e a saída será os vetores com as palavras de contexto. Após ser treinada (no caso deste trabalho por 20 gerações), apresenta como resultado a transformação das palavras tokenizadas em valores numéricos, formando um vetor com as características das palavras normalizadas entre -1 e 1.

O *Word2Vec* gera um vetor com n características para cada comentário analisado, sendo esse valor escolhido pelo usuário. Esta é a representação vetorial das palavras das sentenças, sendo os valores numéricos (características) gerados seguindo as análises sintáticas e morfológicas de acordo com aspectos construtivos do modelo. Este apresenta a capacidade de realizar a composicionalidade. Ao adicionar-se dois vetores de palavras resulta em um vetor que é uma composição semântica de palavras individuais, por exemplo, "homem"+ "real"= "rei"(Mikolov et al. (2013) e Mikolov et al. (2016)).

A análise de sentimentos é realizada pelos classificadores que fazem a leitura dos valores gerados pelo *Word2Vec*.

2.5 Algoritmo Genético

Algoritmos genéticos são técnicas de otimização desenvolvidas baseadas na teoria da evolução de Darwing, mais especificamente, no processo de seleção natural, onde os indivíduos com maior aptidão ao ambiente sobrevivem e se sobressaem sobre os menos aptos que são eliminados (Silveira and Barone (1998)).

Estes algoritmos de acordo com Choi et al. (2016) seguem uma sequência de processos de evolução dos indivíduos contidos na população durante iterações denominadas gerações. Durante cada geração os indivíduos passam por processos de *crossover* e mutação, recebem valores de aptidão de acordo com uma função de *fitness* e são selecionados com base nestes valores para compor a população na próxima geração.

O algoritmo genético construído para esse trabalho recebe como entradas a quantidade de características, a função *fitness*, o tamanho da população, a quantidade de indivíduos que serão candidatos a pais, a taxa de mutação, a quantidade de gerações e o valor de erro mínimo.

Antes de iniciar a execução do processo genético, o algoritmo importa os dados, os valores são normalizados e as amostras são embaralhadas, para que haja amostras positivas e negativas nos dados de treino, validação e teste. Os dados são divididos segundo a proporção: 60% treino, 20% validação e 20% teste. Então a população inicial é gerada selecionando para cada indivíduo as características que vão formar seu genótipo de forma aleatória, sendo cada um desses indivíduos composto de 5 valores inteiros.

Para cada geração é calculado o valor do *fitness* de acordo com a função passada e as amostras. Nesse caso, a função *fitness* é o próprio ANFIS. Depois, a população é ordenada de acordo com o valor calculado do *fitness*, em ordem decrescente, já que o objetivo é maximizar a acurácia.

O algoritmo genético calcula a probabilidade de sorteio de cada pai, baseado no valor do ranking dos indivíduos, utilizando-se o método da roleta viciada. Para determinar os indivíduos da próxima geração, é utilizado elitismo, ou seja, o melhor indivíduo sempre permanece na população.

Todos os indivíduos candidatos a pais também permanecem na população e os filhos substituem os indivíduos que não foram selecionados como candidatos à pais. Cada par (pai, mãe) selecionado gera dois filhos através do *crossover* de um ponto, sendo que o ponto de corte é escolhido de forma aleatória. O *crossover* se repete até que seja obtido o número de filhos predeterminado. A mutação é feita gene a gene, substituindo-se o valor do gene a ser mutado por um número aleatório dentro do intervalo válido correspondente as características (inteiros de 1 a 20). Não ocorre mutação no melhor indivíduo, o que garante que a próxima geração terá um desempenho no mínimo, igual a população anterior. Os parâmetros utilizados são mostrados na Tabela 1.

Tabela 1. Tabela de Parâmetros aplicados ao Algoritmo Genético.

Parâmetros	Valores
Geração da População Inicial	Aleatória
Número de Características	5
Tipo de Característica	Números Inteiros
Número de Indivíduos	12
Número de Filhos Gerados	6
Taxa de Mutação	40%
Número de Gerações	12
Método de Seleção	Roleta Viciada
Método de Crossover	Corte em um ponto aleatório
Método de Mutação	Gene a gene
Número de Indivíduos Elite	1

2.6 Classificador

O modelo utilizado nesse trabalho é aquele em que, a partir de amostras de textos, o classificador realiza a classificação das amostras em duas categorias: negativo ou positivo. O classificador proposto é *fuzzy* genético.

O classificador, de acordo com o algoritmo, transforma os valores recebidos dos vetores de características *word embedding* nos valores 0 ou 1. Nesta implementação, o valor 0 é atribuído para os sentimentos positivos e o valor 1 é atribuído para os sentimentos negativos. O classificador faz então a comparação com a classificação original da base de dados e analisa estatisticamente os acertos.

Para o SFG foram utilizadas 20 características e, após a seleção do AG, 5 características foram enviadas para classificação pelo ANFIS, cuja configuração pode ser vista na tabela 2. Além do SFG, para fins de comparação e análise de desempenho, foram utilizados para a classificação, os algoritmos de Regressão Logística, *Support Vector Machine* e *Random Forest*, que foram implementados utilizando a biblioteca *python sklearn*. Estes classificadores utilizaram 200 características geradas pelo *Word2Vec* e não foi feita a seleção das características utilizadas.

O algoritmo *Random Forest*, cria uma floresta de árvores de decisão aleatoriamente. Tal combinação de árvores de

decisão é um dos mais simples algoritmos de aprendizagem. A classificação é feita ordenando os dados, criando uma árvore de decisão da raiz para as folhas. O algoritmo *Random Forest* busca a melhor característica em um subconjunto aleatório das características (Lan and Pan (2019)).

O algoritmo de Regressão Logística fornece um resultado binomial e a probabilidade do evento ocorrer ou não, ou seja, retorna como resultado 0 ou 1. A regressão logística apresenta como vantagens, a simplicidade de implementação, eficiência computacional, a rapidez no treinamento e a facilidade de regularização. Ele é capaz de resolver problemas de escala industrial (Ray (2019)).

O algoritmo *Support Vector Machine* pode ser utilizado tanto para regressão quanto para classificação. Este é um algoritmo de aprendizado supervisionado que faz a separação de um conjunto de objetos de diferentes classes através de um plano de decisão. Quando os conjuntos de objetos não são linearmente separáveis, utiliza-se funções matemáticas complexas chamadas *kernels* para separá-los (Silva et al. (2015) e Ray (2019)).

2.7 ANFIS

O ANFIS é utilizado como função *fitness* do algoritmo genético (AG). As entradas que o ANFIS recebe do AG são as características definidas pelo indivíduo em avaliação. De acordo com o tamanho da população, é gerado um ANFIS para cada indivíduo com número de variáveis correspondente ao número de características não repetidas contidas no indivíduo, e então, é feita a avaliação de todos os indivíduos da população.

Os dados são divididos segundo a proporção: 60% treino, 20% validação e 20% teste. O sistema de inferência fuzzy (*fis*, do inglês fuzzy inference system) inicial é gerado via *grid partition*, com função de pertinência *gbell*. Para cada entrada foram consideradas duas funções de pertinência. Todos os treinamentos foram feitos com 200 épocas e com objetivo de minimizar a raíz do erro quadrado médio (*RMSE*), considerando como critério de parada antecipada *RMSE* igual a zero. Foi usado o método de otimização *hybrid*, que utiliza mínimos quadrados e gradiente descendente para atualização de parâmetros consequentes e antecedentes.

Depois de terminar o treino do ANFIS, todo o banco de dados é avaliado considerando os conjuntos de treino, validação e teste. Para cada amostra é feita a comparação entre a previsão de polaridade do ANFIS com o valor de polaridade real. O valor da acurácia é calculado de acordo com a equação 1, e corresponde ao valor de *fitness* de cada indivíduo. Os parâmetros utilizados no ANFIS são mostrados na Tabela 2.

3. ANÁLISE E DISCUSSÃO DOS RESULTADOS

O algoritmo genético-fuzzy foi usado para determinar as características mais apropriadas e realizar a classificação de sentimentos dos comentários do banco de dados *Epinions*, utilizando os métodos de extração de características *Word2vec* e Extração por polaridades.

Tabela 2. Tabela de Parâmetros aplicados ao Anfis.

Parâmetros	Valores
Número Máximo de Entradas	5
Percentual de Dados para Treino	60%
Percentual de Dados para Verificação	20%
Percentual de Dados para Validação	20%
Tipo de Fis utilizado	Grid Partition
Número de Funções de Pertinência por Entrada	2
Tipo de Função de Pertinência	Gbellmf
Número de Épocas de Treinamento	200
Método de Otimização	Hybrid

Para análise dos resultados foram utilizados os valores da acurácia (Equação 1), precisão (Equação 2), recall (Equação 3) e F1 (Equação 4), onde TP, FP TN, FN representam as instâncias Verdadeiras Positivas, Falso Positivas, Verdadeiras Negativas e Falso Negativas, respectivamente. O valor F1 é obtido usando os valores de precisão e recall, gerando um número único que indica a qualidade geral do modelo. As fórmulas de cada equação anteriormente descritas são:

$$Acurácia = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

$$Precisão = \frac{TP}{TP + FP} \tag{2}$$

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

$$F1 = \frac{2 * Precisão * Recall}{Precisão + Recall} \tag{4}$$

Os resultados obtidos são mostrados nas subseções a seguir.

3.1 Resultados utilizando a análise de polaridade das palavras como método de extração de características

O gráfico mostrado na Figura 1 representa o desempenho do melhor indivíduo de cada geração do algoritmo de acordo com a acurácia (equação 1). O gráfico da Figura 2 mostra quais características foram usadas no classificador em cada geração pelo melhor indivíduo, onde cada linha do gráfico corresponde a um gene.

Observando os gráficos é possível perceber que as alterações nas características empregadas geram alteração na acurácia do algoritmo. O resultado na primeira geração já é otimizado, uma vez que todos os indivíduos da população foram testados e o resultado mostrado corresponde ao indivíduo com maior acurácia dentre eles, e consequentemente, ao longo das outras gerações foram obtidos resultados ainda melhores, gerando um incremento adicional na acurácia de 2%. É importante citar que usar o ANFIS para as 20 características é inviável devido ao tempo de processamento e custo computacional.

O conjunto de características selecionadas, os valores da acurácia e F1 obtidos são mostrados na Tabela 3.

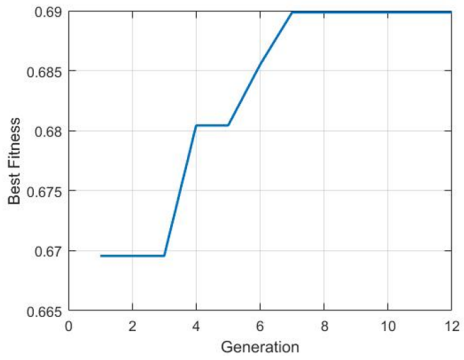


Figura 1. Evolução da acurácia da melhor solução - Polaridade de palavras.

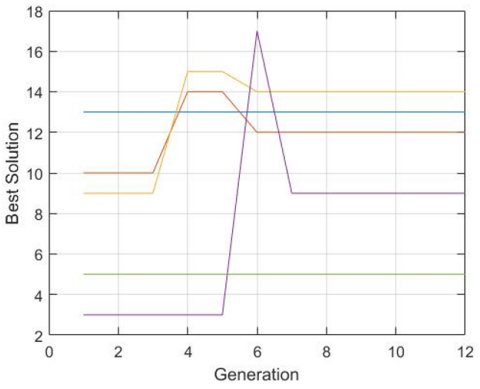


Figura 2. Evolução das características selecionadas - Polaridade de palavras.

3.2 Resultados utilizando o Word2Vec como método para extração de características

O gráfico mostrado na Figura 3 representa o desempenho do melhor indivíduo de cada geração do algoritmo. O gráfico da Figura 4 mostra quais características foram usadas no classificador em cada geração, onde cada linha representa um gene do indivíduo.

De maneira semelhante ao resultado exposto anteriormente para o método de extração de polaridades, é possível perceber que as alterações nas características empregadas geram alteração na acurácia do algoritmo. Entre a primeira geração e a última houve um incremento na acurácia de 3% (Além da otimização inicial da primeira geração). O conjunto de características selecionadas, os valores da acurácia e *F1* obtidos são mostrados na Tabela 4.

Tabela 3. Valores obtidos com Polaridade de palavras.

Geração	Características Selecionadas	Acurácia	F1
Primeira	3, 5, 9, 10, 13	0,6696	0,6864
Última	5, 9, 12, 13, 14	0,6899	0,7040

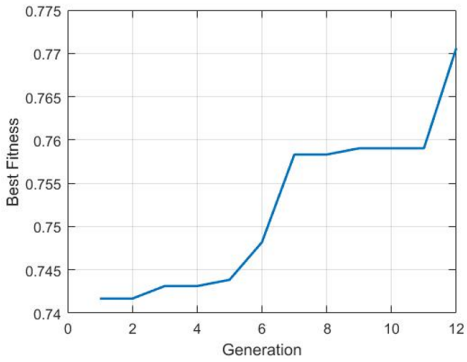


Figura 3. Evolução da acurácia da melhor solução - Word2vec.

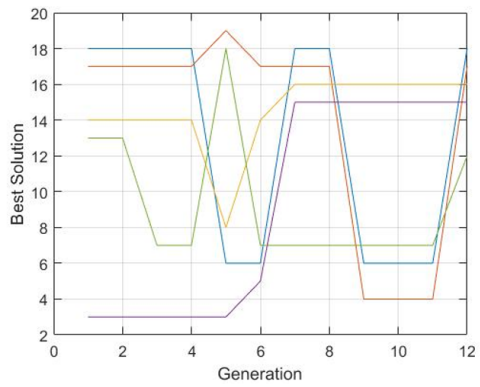


Figura 4. Evolução das características selecionadas - Word2vec.

É notável a ocorrência de maiores variações do conjunto de características que melhoram a acurácia do classificador para o método de extração usando o *Word2Vec* do que para o método de extração de polaridades (Figuras 2 e 4). Isto ocorre, principalmente, devido ao fato de as características geradas pelo método de extração de polaridades possuírem uma maior correlação entre si do que as características geradas pelo *Word2Vec*.

Nota-se ainda uma melhor acurácia, em geral, para o classificador usando características extraídas via *Word2Vec* do que para o modelo usando características extraídas via o método de polaridades, o que pode ser observado comparando as Figuras 1 e 3.

A partir dos valores de acurácia e *F1* obtidos para o algoritmo proposto com AGs e ANFIS, é possível fazer a comparação com diferentes algoritmos aplicados na análise

Tabela 4. Valores obtidos com Word2vec.

Geração	Características Selecionadas	Acurácia	F1
Primeira	3, 13, 14, 17,18	0,7417	0,7411
Última	12, 15, 16, 17, 18	0,7706	0,7766

de sentimentos. A Tabela 5 mostra os dados de *F1* obtidos utilizando o método da extração de polaridades (20 características geradas) e o método de extração via *Word2Vec* (20 características geradas) para o SFG proposto, além de outros métodos de análise de sentimentos automáticos como a SVM, *Random Forest* e Regressão Logística, todos aplicados a base *Epinions 3* e com 200 características extraídas via *Word2Vec*.

Utilizou-se o *Word2Vec* para extrair 200 características a serem utilizadas pelos métodos de análise automáticos, uma vez que foi notada uma grande queda de desempenho destes algoritmos quando usando apenas 20 características como em nosso SFG.

Tabela 5. Comparação com outros métodos.

Algoritmos - Métodos de Extração de Características	F1
SFG – Extração de Polaridades	0,7040
SFG – Word2Vec	0,7766
Regressão Logística – Word2Vec	0,7491
Support Vector Machine – Word2Vec	0,7606
Random Forest – Word2Vec	0,7563

Pode-se verificar que o SFG proposto neste trabalho, que utilizou como banco de dados características extraídas via método de extração de polaridades, possui o pior desempenho, com o valor *F1* de aproximadamente 0,70. É possível que isso tenha ocorrido devido ao fato de as características obtidas possuírem, em geral, uma alta correlação, já que muitas delas são derivadas de combinações entre si. É possível ainda que um melhor conjunto de características possa ser selecionado, uma vez que foram utilizadas poucas gerações na busca da melhor solução pelo SFG, devido ao tempo que o algoritmo demanda.

Para o banco de dados com características extraídas via *Word2Vec*, o SFG conseguiu determinar um conjunto de características que apresenta melhor desempenho com relação a todos os algoritmos automáticos de análise de sentimentos presentes na Tabela 5, o que revela o potencial aplicável do SFG proposto. Além disto, o SFG nesta situação também utilizou um número pequeno de gerações, o que permite dizer que, possivelmente, poderiam ocorrer resultados ainda melhores na acurácia.

Por fim, o SFG permite a redução da complexidade do problema, uma vez que atingiu tais resultados utilizando apenas 5 das características extraídas, enquanto os métodos de análise de sentimentos automáticos utilizados na comparação utilizaram 200 características para atingir os resultados mostrados na Tabela 5.

4. CONCLUSÃO

Atualmente a análise de sentimentos é uma área que tem despertado bastante interesse por suas diversas aplicações. Neste trabalho é proposto um classificador *fuzzy* para mineração de opinião e classificação do sentimento geral de textos. Os resultados obtidos mostram uma melhor acurácia para o classificador usando características extraídas via *Word2Vec* quando comparado ao método por polaridades. O *F1 Score* obtido com o SFG utilizando 5 características extraídas via *Word2Vec* é superior aqueles obtidos pelos métodos NB, KNN e SVM utilizando 200 características. Para trabalhos futuros, é possível estudar maneiras de selecionar um melhor conjunto de características, aumentar o número de gerações na busca da melhor solução pelo SFG ou reduzir o tempo de processamento, pois foram fatores limitantes para o desenvolvimento deste trabalho. Além disso, o classificador pode ser adaptado para atribuir diferentes classes às avaliações, considerando os graus de subjetividade existentes, por exemplo, se um produto é bom ou muito bom, criando uma espécie de *ranking* das opiniões.

AGRADECIMENTOS

Os autores agradecem à CAPES e FAPEMIG por apoiarem este trabalho.

REFERÊNCIAS

Chen, L. and Wang, F. (2013). Preference-based clustering reviews for augmenting e-commerce recommendation. *Knowledge-Based Systems*, 50, 44–59.

Chen, X. and Dai, Y. (2020). Research on an improved ant colony algorithm fusion with genetic algorithm for route planning. *2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, 1, 1273–1278.

Choi, K., Jang, D., Kang, S., Lee, J., Chung, T., and Kim, H. (2016). Hybrid algorithm combing genetic algorithm with evolution strategy for antenna design. *IEEE Transactions on Magnetics*, 52(3), 1–4.

Choudhary, M. and Choudhary, P.K. (2018). Sentiment analysis of text re-viewing algorithm using data mining. *2018 International Conferenceon Smart Systems and Inventive Technology (ICSSIT)*, 532–538.

Cordón, O., Herrera, F., Hofmann, F., and Magdalena, L. (2001). Genetic fuzzysystems. evolutionary tuning and learning of fuzzy knowledge bases. *World Scientic*.

Cui, H., Mittal, V., and Datar, M. (2006). Comparative experiments on sentiment classification for online product reviews. In *Proceedings of the 21st National Conference on Artificial Intelligence - Volume 2*, AAAI'06, 1265–1270. AAAI Press.

G. Chen, Y.W. and Xu, X. (2016). An analysis of the sales and consumerpreferences of e-cigarettes based on text mining of online reviews. *2016 3rd International Conference on Systems and Informatics (ICSAI)*, 1045–1049.

Lan, H. and Pan, Y. (2019). A crowdsourcing quality prediction model based on random forests. *Proceedings - 18th IEEE/ACIS International Conference on Computer and Information Science, ICIS 2019*, 315–319.

Li, M.Y., Kok, S., and Tan, L. (2018). Don't classify, translate: Multi-level e-commerce product categorization via machine translation. *CoRR*, abs/1812.05774.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *1st International Conference on Learning Representations, ICLR 2013 - Workshop Track Proceedings*, 1–12.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J. (2016). Distributed representations of words and phrases and their compositionality. *EMNLP 2016 - Conference on Empirical Methods in Natural Language Processing, Proceedings*, 1389–1399.

Mortezanezhad, A. and Daneshifar, E. (2019). Big-data clustering with genetic algorithm. *2019 5th Conference on Knowledge Based Engineering and Innovation (KBEI)*, 702–706.

P. Pankaj, P. Pandey, M. and Soni, N. (2019). Sentiment analysis on customer feedback data: Amazon product reviews. *2019 InternationalConference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)*, 320–322.

Padmaja, K. and Hegde, N.P. (2019). Twitter sentiment analysis using adaptiveneuro-fuzzy inference system with genetic algorithm. *2019 3rd Inter-national Conference on Computing Methodologies and Communication(ICCMC)*, 498–503.

Ray, S. (2019). A quick review of machine learning algorithms- proceedings of the international conference on machine learning. *Proceedings of the International Conference on Machine Learning, Big Data, Cloud and Parallel Computing: Trends, Prespectives and Prospects, COM-IT-Con 2019*, 35–39.

Shi, H. and Xu, M. (2018). A data classification method using genetic algorithm and k-means algorithm with optimizing initial cluster center. *2018 IEEE International Conference on Computer and Communication Engineering Technology (CCET)*, 224–228.

Silva, R.D., Thé, G.A., and De Medeiros, F.N. (2015). Geometrical and statistical feature extraction of images for rotation invariant classification systems based on industrial devices. *21st International Conference on Automation and Computing: Automation, Computing and Manufacturing for New Economic Growth, ICAC 2015*, 1–6.

Silveira, S.R. and Barone, D.A.C. (1998). Jogos educativos computadorizados utilizando a abordagem de algoritmos genéticos. *Universidade Federal do Rio Grande do Sul. ...*.

Solangi, Y.A., Solangi, Z.A., Aarain, S., Abro, A., Mallah, G.A., and Shah, A. (2018). Review on natural language processing (nlp) and its toolkits for opinion mining and sentiment analysis. In *2018 IEEE 5th International Conference on Engineering Technologies and Applied Sciences (ICETAS)*, 1–4.

Sun, S., Luo, C., and Chen, J. (2016). A review of natural language processing techniques for opinion mining systems. *Information Fusion*, 36.

Tang, D., Wei, F., Yang, N., Zhou, M., Liu, T., and Qin, B. (2014). Learning sentiment-specific word embedding for twitter sentiment classification. *52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014 - Proceedings of the Conference*, 1, 1555–1565.

Vanaja, S. and Belwal, M. (2018). Aspect-level sentiment analysis on e-commerce data. *2018 International Conference on Inventive Researchin Computing Applications (ICIRCA)*, 1275–1279.

Y. Liu, J.L. and Shahbazzade, S. (2018). Sentiment classification of e-commerce product quality reviews by fl-svm approaches. *2018 IEEE 17th International Conference on Cognitive Informatics Cognitive Computing (ICCI*CC)*, 292–298.

Zong, Z. and Hong, C. (2018). On application of natural language processing in machine translation. In *2018 3rd International Conference on Mechanical, Control and Computer Engineering (ICMCCE)*, 506–510.

Sociedade Brasileira de Automática (SBA)
XV Simpósio Brasileiro de Automação Inteligente - SBAI 2021, 17 a 20 de outubro de 2021

Extração de Entidades de Produtos Utilizando Técnicas de *Few-Shot Learning*

Aleson G. S. Chaves [*] **Fernanda C. e Silva** [*] **Danton D. Ferreira** [**]
Bruno H. G. Barbosa [**] **Sinval T. Nascimento** [***]

[*] *Programa de Pós-Graduação em Engenharia de Sistemas e Automação, Universidade Federal de Lavras, MG*
[**] *Departamento de Automática, Universidade Federal de Lavras, MG, (e-mail: aleson.chaves@estudante.ufla.br, fernanda.silva10@estudante.ufla.br, danton@ufla.br, brunohb@ufla.br)*
[***] *Omnilogic Inteligência, (e-mail: sinval@omnilogic.com.br).*

Abstract: The interpretation of unstructured data in textual format is a research area that can be applied to electronic commerce, which has generated a large amount of natural language data. With the growth of marketplaces, new products are added daily by several different retailers, and there may be classes in the product databases that have only a few samples. Obtaining classifiers that consider new classes with few samples is a complex task, whether due to the computational cost of retraining existing models to include these classes, or due to their low number of samples. In this sense, Few Shot Learning (FSL) techniques are promising options. That said, this paper compared the FSL classifiers Matching Networks and Siamese Neural Network in a marketplace's product classification problem, with 34 classes and 394 samples in total. These classifiers were compared with the K-Nearest Neighbors (KNN) algorithm and Principal Component Analysis was applied for database size reduction. The FSL algorithms implemented performed better than the KNN method in the leave-one-out cross validation test, showing an accuracy of 96.85%, even considering classes with a low number of samples.

Resumo: A interpretação de dados não estruturados no formato textual é uma área de pesquisa que pode ser aplicada para o comércio eletrônico, que tem gerado uma grande quantidade de dados em linguagem natural. Com o crescimento dos *marketplaces*, novos produtos são adicionados diariamente por diferentes lojistas, e podem existir classes de produtos que tenham poucas amostras nas bases de dados. A obtenção de classificadores que considerem novas classes com poucas amostras é uma tarefa complexa, seja pelo custo computacional de retreinar os modelos existentes para contemplar essas classes, ou pela baixa quantidade de amostras delas. Nesse sentido, técnicas de *Few Shot Learning* (FSL) são opções promissoras. Isso posto, este trabalho comparou os classificadores FSL *Matching Networks* e Redes Neurais Siamesas no problema de classificação de produtos de um *marketplace*, com 34 classes e 394 amostras. Esses classificadores foram comparados com o algoritmo *K-Nearest Neighbors* (KNN) e foi aplicada a Análise de Componentes Principais para redução de dimensão do banco de dados. Os algoritmos FSL obtiveram desempenho superior ao KNN no teste de validação cruzada *leave-one-out*, apresentando acurácia de 96,85%, mesmo considerando classes com baixo número de amostras.

Keywords: Natural Language Processing; E-commerce; Machine Learning; Few-Shot Learning; Siamese Neural Network; Matching Networks; Artificial Intelligence

Palavras-chaves: Processamento de Linguagem Natural; Comércio Eletrônico; Aprendizado de Máquina; Few-Shot Learning; Redes Neurais Siamesas; Redes Matching; Inteligência Artificial

1. INTRODUÇÃO

Com o desenvolvimento dos recursos computacionais e do acesso à internet por grande parte da população, houve um aumento significativo do comércio eletrônico (Ristoski et al., 2018). O número de consumidores e vendedores adeptos ao modelo de mercado *on-line* tem crescido, o que gera um aumento no número de ofertas nas plataformas, com uma grande variedade de produtos. Novas ofertas são adicionados diariamente em grandes plataformas de mercado como *Amazon* e Magazine Luiza (Krishnan and Amarthaluri, 2019), criando uma grande quantidade de dados em formato textual (Zheng et al., 2019). A área de pesquisa que estuda os problemas da geração e interpretação da linguagem humana pelas máquinas é o Processamento de Linguagem Natural (NLP, do inglês *Natural Language Processing*), uma subárea da ciência da computação e inteligência artificial (e Silva et al., 2020).

Um problema comum nas bases de dados dos *marketplaces* é a ocorrência de classes que contém poucas amostras, o que dificulta o processo de treinamento de classificadores tradicionais, por exemplo, aqueles baseados em Redes Neu-

Sociedade Brasileira de Automática (SBA)
XV Simpósio Brasileiro de Automação Inteligente - SBAI 2021, 17 a 20 de outubro de 2021

rais Artificiais. A obtenção de classificadores de produtos que levem em consideração novas classes com poucas amostras é uma tarefa complexa, seja pelo custo computacional de retreinar os modelos existentes de forma a contemplar as novas classes, ou seja pela baixa quantidade de amostras dessas novas classes. Como existem classes novas que não passaram pelo treinamento do classificador em operação, é necessário o re-treino do classificador com toda a base de dados, incluindo essas novas classes.

Os erros de classificação provocam, dentre outros problemas, a categorização incorreta de produtos nos *marketplaces*, podendo causar experiências desagradáveis no processo de compra. A falta de padrão pode fazer com que novas ofertas não apareçam ou apareçam na página do produto errado, devido a não correspondência dos mesmos produtos de vendedores distintos (Ristoski et al., 2018).

Nesse sentido, técnicas de *Few Shot Learning* (FSL) são opções promissoras por serem projetadas especificamente para lidar com treinamento de modelos com pequenas quantidades de amostras por classe (Jadon and Garg, 2020). Existem vários algoritmos de aprendizado FSL, como: *Matching Networks* (MN) (Vinyals et al., 2016), *Model Agnostic Meta-Learning* (MAML) (Finn et al., 2017), Redes Siamesas (Koch et al., 2015) e *Graph Neural Networks* (GNN) (Garcia and Bruna, 2018).

As Redes Siamesas são boas alternativas para lidar com problemas de reconhecimento de padrões a partir de uma ou poucas amostras (Koch et al., 2015). Essa rede pode ser usada para medir a similaridade entre duas entradas e determinar se o par de amostras pertence a uma mesma classe ou não (Bromley et al., 1994). As Redes *Matching* aprendem a mapear os dados de entrada e utilizam tais dados para fazer predição por meio de um mecanismo de atenção aplicado ao cálculo de similaridade do cosseno entre os representantes e o alvo (Vinyals et al., 2016).

Isso posto, este trabalho tem por objetivo propor e comparar o desempenho das Redes Siamesas e Redes *Matching* para extração de entidades de produtos de um *marketplace*. Esses classificadores foram comparados também com o algoritmo *K-Nearest Neighbors* (KNN), que foi utilizado como *baseline* do projeto. Ademais, foi analisado o desempenho dos algoritmos após a redução de dimensão das amostras do banco de dados, utilizando o método de transformação das características *Principal Component Analysis* (PCA) (Krishnan and Dutta, 2018).

As abordagens propostas têm como benefício manter uma boa acurácia dos classificadores em operação nos *marketplaces*, mesmo com a chegada de novas classes, o que mantém a plataforma de *marketplace* bem categorizada, com as especificações de produto organizadas e anúncios padronizados em uma estrutura de texto que atenda as necessidades ao consumidor e facilita o processo de busca e compra de produtos. Também podem reduzir o uso do servidor em nuvem, disponibilizando o mesmo para uso em outras aplicações ou reduzindo o valor do contrato.

2. MATERIAIS E MÉTODOS

2.1 Base de Dados

Os códigos computacionais foram implementados em linguagem *Python*. A base de dados utilizada foi cedida pela empresa parceira deste trabalho e é composta por amostras com textos não estruturados, provenientes de uma plataforma de comércio eletrônico. A base de dados contém 394 ofertas divididas em 34 classes, como mostrado na Tabela 1. Conforme pode ser observado, trata-se de uma base bastante desbalanceada e que possui poucas amostras.

Tabela 1. Base de Dados de Classes Novas

Descrição da Base	Valores
Número de amostras	394
Número de classes	34
Classes com 1 Amostra	13
Classes com 2 amostras	5
Classes com 3 a 8 amostras	10
Classes com 20 a 60 amostras	5
Classes com 113 amostras	1

Cada oferta em linguagem natural é transformada em um vetor de dados numéricos com 1200 valores, que são as características de cada oferta que serão usadas nos classificadores *few-shot*. Esses valores são extraídos por um modelo que encontra-se em operação e que foi previamente treinado com amostras de classes diferentes daquelas utilizadas neste trabalho. Ou seja, a partir de um classificador (Rede Neural Artificial) treinado previamente com um número grande de classes e amostras, são extraídas características (saídas dos neurônios da última camada intermediária da rede neural), em um processo que pode ser interpretado como *transfer learning*.

2.2 Few-Shot Learning

O processo de aprendizagem com a utilização de poucos dados é uma tarefa complexa para as redes neurais tradicionais (Jadon and Garg, 2020). Para o caso de uma pequena base de dados, um algoritmo simples como o KNN pode ter desempenho melhor do que uma rede neural *Multi-Layer Preceptron* (MLP), caso se tenha um bom pré-processamento que leve a uma boa separação entre as classes e a escolha de uma boa métrica para o cálculo da distância (Jadon and Garg, 2020).

As redes de aprendizado profundo conseguem um bom desempenho em uma variedade de tarefas, porém necessitam de uma grande quantidade de dados para aprender (Zheng et al., 2019). No entanto, existem bases de dados com apenas algumas poucas amostras por classes e isso dificulta o processo de treinamento das redes neurais. Neste caso, é necessário implementar métodos de aprendizagem de máquinas específicos para redes neurais que sejam capazes de realizar o treinamento quando se tem poucas amostras por classe. Estes métodos são conhecidos como *Few-Shot Learning* (Wang et al., 2020).

2.3 K-Vizinhos Próximos

O k-vizinhos mais próximos (KNN) é um dos mais simples e tradicionais classificadores utilizados em reconhecimento

Sociedade Brasileira de Automática (SBA)
XV Simpósio Brasileiro de Automação Inteligente - SBAI 2021, 17 a 20 de outubro de 2021

de padrões, cujo desempenho é competitivo em relação aos mais complexos classificadores da literatura. A grande vantagem do KNN é que ele não depende de treinamento pois a sua tarefa se resume em identificar, em um conjunto rotulado, os k objetos mais semelhantes (com menores distâncias) da amostra não rotulada (Trstenjak et al., 2014). Portanto, o KNN é uma opção interessante para um conjunto de dados com pequena quantidade de amostras. De fato, o KNN pode ser considerado uma técnica de *few-shot learning* uma vez que, a partir do momento que uma ou mais amostras de uma classe nova estiver disponível, tal amostra pode ser prontamente adicionada no conjunto de dados e poderá ser usada na classificação de outra amostra de sua mesma classe.

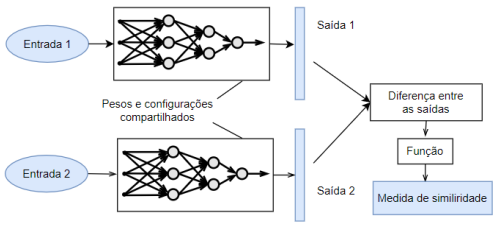
O algoritmo KNN foi implementado como classificador *few-shot* para o banco de dados da Seção 2.1. As características da nova oferta a ser classificada são comparadas (medida de similaridade) com as características das amostras já existentes no banco de dados, e o algoritmo KNN é então implementado.

Para os testes, as classes que possuíam apenas uma amostra foram colocadas na base de treino apenas para tornar o problema mais complexo. As que possuíam duas amostras, uma foi usada para teste (*Leave-One-Out*) e a outra como representante. Nas classes com três ou mais amostras, uma foi utilizada para teste (*Leave-One-Out*) e as demais utilizadas como representantes ou para o cálculo do centroide que foi utilizado como representante da respectiva classe. As distâncias utilizadas foram: *Euclidiana (ED), cosseno (CD), Manhattan (MD), Hassanat (Has-D), Lorentzian (LD)* e *Clark (Cla-D)*. Também foram testados diferentes valores do parâmetro k.

2.4 Redes Siamesas

A Rede Neural Siamesa foi apresentada no trabalho de Bromley et al. (1994), fazendo a comparação de assinaturas escritas em um *tablet*. Esse tipo de rede neural de treinamento supervisionado pode ser usada para medir a similaridade entre duas entradas e, portanto, pode determinar se um par de amostras pertence a uma mesma classe ou não. A Figura 1 apresenta a arquitetura de uma rede siamesa. Ela é formada por duas sub-redes idênticas, que possuem a mesma estrutura e são configuradas com os mesmos pesos e parâmetros.

Figura 1. Arquitetura simplificada de uma rede siamesa



Fonte: Do Autor (2021)

Para o treinamento da rede são necessários pares de amostras da mesma classe, que terão valor de alvo 1, e pares de amostras de classes diferentes que terão valor de alvo 0. Cada sub-rede recebe as entradas e elas produzem um vetor na saída, em um espaço multidimensional de similaridade semântica, criado pela rede siamesa. Esse espaço tem a mesma dimensão do número de neurônios de saída das sub-redes.

O neurônio que faz a junção dos vetores de saída das sub-redes também é responsável por medir a distância entre eles. A distância é usada para determinação do valor de similaridade, que pode ser usado para definir se as entradas da rede são de uma mesma classe ou não, de acordo com os limites estabelecidos para que as amostras sejam consideradas similares. Esse valor de similaridade é utilizado pela rede para calcular a perda, que será empregada na atualização dos pesos das sub-redes via *backpropagation*. A perda é calculada pela diferença entre o valor de similaridade calculado e o valor do alvo de cada par de amostras.

Após o treino da rede siamesa, o conceito *few-shot learning* pode ser aplicado diretamente, pois a princípio a rede aprendeu a distinguir amostras de classes diferentes ou a identificar amostras pertencentes à mesma classe. No problema analisado neste trabalho, a aplicação das redes siamesas em *few-shot learning* será feita para classificação de novas amostras de produtos que tenham poucas amostras e não são conhecidas pelo classificador em operação, assim como realizado para o classificador KNN.

As sub-redes utilizadas na construção da rede siamesa foram propostas com uma única camada intermediária, com 300 neurônios. Após a camada intermediária há uma camada que calcula a distância ponto a ponto entre as saídas das duas sub-redes. A saída dessa camada alimenta uma camada de saída com um único neurônio que calcula o valor de similaridade entre as entradas.

2.5 Redes Matching

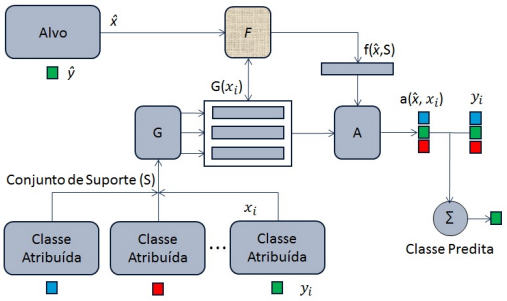
As redes *matching* aprendem a mapear uma pequena base de dados de treino e teste em um mesmo espaço de *embeddings* (tarefa *few-shot*). Elas são treinadas para aprender uma representação com os *embeddings* da pequena base de dados de treino de forma adequada, em busca da minimização dos erros do cálculo de similaridade entre o alvo e os representantes de classe. As redes *matching* funcionam com a ideia de um KNN diferenciável com medida de similaridade do cosseno. Além dos *encoders* que formam os *embeddings* estas redes utilizam um mecanismo de atenção pela aplicação da função *softmax* na distância do cosseno.

As redes *matching* desempenham bem a tarefa *few-shot* para classes não vistas no treinamento (Vinyals et al., 2016), sendo implementadas em cinco principais etapas: (i) Extrator de *embeddings* G, que forma o *encoder* com os dados das amostras; (ii) *Embeddings* de contexto completo F, cujo objetivo é obter o contexto entre as amostras, quando houver, e é formado por uma rede LSTM bidirecional (o uso é opcional); (iii) cálculo de similaridade com a função de distância cosseno, c; (iv) mecanismo de atenção que é dado pela aplicação da função *softmax* na saída da função de cálculo de similaridade c; e (v) cálculo da função de perda entropia cruzada (Jadon and Garg, 2020).

Sociedade Brasileira de Automática (SBA)
XV Simpósio Brasileiro de Automação Inteligente - SBAI 2021, 17 a 20 de outubro de 2021

A arquitetura com as etapas das redes *matching* pode ser vista na Figura 2. O conjunto de suporte (S) é o conjunto de dados de entrada, como se fossem os representantes (vizinhos) do KNN sendo que x_i é o vetor de dados, y_i são os rótulos. O alvo $\hat{x}$ é a amostra a ser predita.

Figura 2. Arquitetura das redes *Matching*



Fonte: Do Autor (2021)

Os dados do conjunto de suporte formados são encaminhados para o *encoder* G, que aprende uma nova representação para as amostras. Depois os dados passam pelo *embedding* de contexto completo F, ou seja, a saída de G e o alvo $\hat{x}$ passam pela rede BILSTM (Vinyals et al., 2016). Conforme a Figura 2, na sequência de F pode ser observada a equação $f(\hat{x}, S)$ que é a nova representação dos dados do conjunto de suporte (S) e do alvo $\hat{x}$ obtida por F.

O próximo passo antes da aplicação da atenção é o cálculo de similaridade utilizando a função do cosseno. A similaridade é calculada entre a nova representação do conjunto de suporte (x_i) e o alvo $\hat{x}$. O mecanismo de atenção A é obtido pela aplicação da função *softmax* na saída da função de similaridade do cosseno conforme a Equação 1, em que c é a função que executa o cálculo de similaridade com a distância do cosseno:

$$a(\hat{x}, x_i) = \frac{e^{c(f(\hat{x}),g(x_i))}}{\sum_{j=1}^{k} e^{c(f(\hat{x}),g(x_j))}}. \quad (1)$$

A predição é dada pela Equação 2 que implementa a combinação linear da *softmax* da camada de atenção com o vetor *one hot encoded* dos rótulos y_i:

$$P(\hat{y}|\hat{x}, S) = \sum_{i=1}^{k} a(\hat{x}, x_i) y_i. \quad (2)$$

Esta equação é uma combinação linear de probabilidades que determina a qual classe o alvo pertence. A função de perda normalmente utilizada é a entropia cruzada.

O *encoder* G foi implementado como uma rede MLP com apenas uma camada intermediária com 100 neurônios e função *tanh*, configuração esta definida após testes preliminares na base de dados. Na parametrização das redes *matching*, além dos parâmetros tradicionais tal como os existentes na MLP, existem dois parâmetros a serem ajustados, esses são chamados "amostras por classe" e lote (*batch*). Em cada posição do lote é realizado um teste de similaridade entre o conjunto de suporte e o alvo $\hat{x}$ com a distância do cosseno. O parâmetro "amostras por classe" é bem intuitivo, pois são os números de amostras de cada uma das classes que vão compor o lote, é algo similar ao número de representantes do KNN. A atualização dos pesos acontece após o término do cálculo de cada um dos lotes. A acurácia é dada pela média dos lotes.

As redes *Matching* foram implementadas empregando a função de perda de entropia cruzada e o otimizador *Adam* com taxa de aprendizagem de 0,001 e 0,0001 para 307 e 1200 características respectivamente. Foram utilizadas 34 "classes por conjunto de suporte" e 2 "amostras por classe". Número de épocas de treinamento foram 50 e 80 para 307 e 1200 características respectivamente e 10 lotes. O *embedding* de contexto completo F não foi utilizado. (Os parâmetros foram ajustados experimentalmente usando o banco de dados de treino, com a seguinte faixa de variação: "amostras por classe" (2 a 111), lotes (10 a 20), taxa de aprendizagem (0,01 a 0,0001). No caso da MLP as faixas de variações dos parâmetros da camada intermediária foram as seguintes: camadas (1 a 5), neurônios (10 a 307), função não linear (*tanh* e *relu*)).

Para a avaliação de desempenho dos algoritmos aqui implementados foi utilizada a medida de acurácia, obtida no método de validação cruzada *Leave-One-Out* (SCHREIBER et al., 2017). Esse método é aconselhável para este trabalho, pois a base de dados possui poucas amostras.

3. ANÁLISE E DISCUSSÃO DOS RESULTADOS

3.1 O Algoritmo KNN

Os experimentos com o KNN foram iniciados com um vizinho, $k = 1$, ou um representante (centroide) por classe para as 34 classes e 394 amostras da base de dados. O uso de vários representantes por classe gerou melhor resultado do que apenas o uso do centroide, como mostrado na Tabela 2, juntamente com os resultados obtidos para diferentes medidas de similaridade. As distâncias *Manhattan (MD), Hassanat (Has-D)* e *Lorentzian (LD)* apresentaram os melhores resultados, com 96,33% de acerto, seguidas pela distância cosseno, com 95,80% de acerto.

Tabela 2. Valores da acurácia (k=1) para várias distâncias, com uso de centroide ou não

Distâncias	MD	Has-D	LD	CD	ED	Cla-D
KNN	96,33	96,33	96,33	95,80	95,01	94,75
KNN (cent.)	81,36	77,16	80,05	82,41	84,78	76,11

A fim de encontrar o melhor número de k vizinhos, foram escolhidas as distâncias Cosseno, Euclidiana e *Manhattan*, e foram mantidas 21 classes para teste. Para as classes com números de amostras menor do que k, foram adicionadas cópias de suas amostras até se obter $k + 1$ amostras em cada classe. Os resultados encontrados com o teste *Leave-One-Out* são apresentados na Tabela 3.

Pode ser inferido que aumentar o número de vizinhos não foi uma estratégia interessante, pois o valor de $k = 1$ obteve os melhores resultados, o que pode ser justificado pelo fato de alguns *clusters* não serem tão bem definidos na base. A distância cosseno é interessante por possibilitar de forma mais direta a elaboração de um limiar geral de

Sociedade Brasileira de Automática (SBA)
XV Simpósio Brasileiro de Automação Inteligente - SBAI 2021, 17 a 20 de outubro de 2021

decisão de classificação e será a distância utilizada para comparações no restante deste trabalho.

Tabela 3. Valores da acurácia do KNN ($k \geq 1$)

Valores (k)	k=1	k=3	k=4	k=5	k=6	k=7
Cosseno	95,80	94,49	93,96	93,44	93,18	90,81
Euclidiana	95,01	93,96	94,23	92,65	92,13	90,81
Manhatan	96,33	95,01	94,49	93,18	92,65	92,91

Com o objetivo de redução de dimensão do problema em estudo foi realizada a transformação das características com *Principal Component Analysis* (PCA).

O resultado do classificador KNN com $k = 1$ e distância cosseno, utilizando características reduzidas com PCA é apresentado na Tabela 4. A PCA apresentou um resultado interessante, mostrando que há bastante redundância de características, conseguindo uma redução de 74,41% (de 1.200 para 307 características) mantendo a mesma acurácia de 95,8% (99,9999% de variância explicada).

Os resultados obtidos com o algoritmo KNN foram usados para construção de um *baseline* para comparações com algoritmos de *Few-Shot Learning*. A utilização do algoritmo KNN se mostrou uma boa opção para a tarefa de classificação de produtos com poucas amostras, sem a necessidade de treinamento ou ajuste de parâmetros de modelos. O índice de acerto sempre superior a 90%, indica que a maioria das amostras estão bem localizadas em seus respectivos agrupamentos, validando o uso do extrator de características utilizado pelo método de *transfer learning*.

Tabela 4. Resultados do KNN ($k = 1$) com dimensão reduzida com PCA, para diferentes números de características (NC)

Variância(%)	100	98	95	90	85	80
NC	307	107	64	37	24	16
Resultado	95,80	95,54	95,28	94,23	93,70	92,39

3.2 Redes Siamesas

A estrutura utilizada na construção da rede siamesa foi proposta com uma única camada intermediária, em que o número de neurônios foi variado experimentalmente de 200 a 800, sendo que o melhor resultado foi com 300 neurônios nesta camada. O modelo utilizou o otimizador *Adam* com taxa de aprendizagem que foi variada de 0,0001 a 0,001, sendo 0,0006 o valor que com melhor acurácia.

Também foram realizados testes para encontrar o melhor representante das classes, uma vez que, na aplicação proposta, a rede siamesa irá comparar uma amostra de classe conhecida com uma nova amostra que não foi rotulada. Um teste *Leave-One-Out* adicional foi realizado, em que foi calculado um centroide para cada classe a partir da média dos dados de treino de cada classe. Esse centroide foi usado como representante da classe no teste, sendo fixado em uma das entradas da rede siamesa para cada classe.

Os resultados da acurácia e o tempo utilizado aproximado para realizar o treinamento e teste da rede siamesa (SN) utilizando as características da base com 1200 características e da base de 307 características obtidas com o PCA são mostrados na Tabela 5, juntamente com os valores para a rede siamesa com centroide (SNC), o KNN e redes *matching* (MN). Os valores das colunas Acc.(1200) e Tempo (1200) se referem aos resultados de acurácia e tempo de treinamento dos algoritmos para a base de 1200 características e as colunas Acc.(307) e Tempo (307) se referem aos resultados com a base reduzida pelo PCA.

Tabela 5. Desempenho dos classificadores

*Alg.	*Acc.(1200)	Tempo (1200)	Acc.(307)	Tempo (307)
SN	96,32%	3,2 horas	96,85%	2,3 horas
SNC	95,27%	2,1 horas	95,80%	1,0 horas
KNN	95,80%	NA*	95,80%	NA*
MN	96,32%	44,4 horas	96,85%	12,3 horas

*Abreviações: Acc.=acurácia em (%), Alg.=algoritmo, NA=não aplicável

É possível perceber que a rede siamesa superou os resultados obtidos pelo KNN para o teste *Leave-One-Out*. A abordagem com a base de dados reduzida apresenta melhor desempenho que a abordagem com as características originais. Em relação à escolha do melhor representante de cada classe com o centroide como candidato, o resultado obtido não superou os resultados obtidos com o KNN em nenhuma das duas bases de dados.

3.3 Redes Matching

As redes *matching* só conseguem realizar treinamento *Leave-One-Out* para classes que possuem 3 amostras ou mais. Uma amostra é separada para o teste e, no treinamento, o cálculo de similaridade é realizado, ou seja, são necessárias uma amostra para representante de classe e uma amostra para ser o alvo. Para aproveitar classes com somente 2 amostras para ajuste dos parâmetros da rede *encoder*, foi realizada uma cópia da amostra destas classes do conjunto de treino para ser o alvo.

Na tabela 5 (Algoritmo MN, Acc.(307)) pode ser observado o resultado das redes *matching* utilizando as 307 características obtidas por redução com PCA e, que levou a uma rede mais compacta (307 neurônios na entrada e na saída). Com esta configuração, o treinamento convergiu com maior acurácia e rapidez do que com as 1200 características originais (Algoritmo MN, Acc.(1200))(1200 neurônios na entrada e na saída), além de ter reduzido o tempo utilizado nos cálculos de similaridade, que interferem bastante no tempo de treinamento.

As redes *matching* conseguiram superar o KNN, e apresentaram a mesma acurácia da rede siamesa. Porém, as redes *matching* com duas "amostras por classe" apresentaram maior tempo de treinamento do que a rede siamesa. Ela também tem mais parâmetros e arquitetura mais complexa que a rede siamesa, logo é mais complicada de ser parametrizada e treinada. Desta forma, a rede siamesa é mais adequada para a aplicação.

Levando em consideração que as características extraídas das amostras utilizadas são de boa qualidade, e que a base de dados possuiu menos classes do que é utilizado em bases de dados da literatura para tarefa *few-shot* (por exemplo, o conjunto de dados *Omniglot* possui 1623 classes com 20 amostras cada (Lake et al., 2011) e o conjunto de dados *mini*ImageNet possui 100 classes com 600 amostras cada (Vinyals et al., 2016)), superar o KNN não é uma tarefa

Sociedade Brasileira de Automática (SBA)
XV Simpósio Brasileiro de Automação Inteligente - SBAI 2021, 17 a 20 de outubro de 2021

simples, pois tanto a rede siamesa quanto o *encoder* da rede *matching* precisam ser bem treinados para obtenção de características ainda melhores do que aquelas já obtidas pelo extrator de características utilizado.

4. CONCLUSÃO

Os algoritmos de aprendizado *one/few shot learning* são usados em situações em que se dispõe de poucos dados de alguma classe. A maioria das aplicações de comércio eletrônico utilizam algoritmos de aprendizado profundo que apresentam bom desempenho somente nas grandes bases de dados. Então conseguir algoritmos que funcionem bem para bases de dados com poucas classes é um ganho, pois as classes recém chegadas não possuem amostras suficientes para o treinamento das redes neurais tradicionais. Neste sentido os algoritmos *few-shot learning* empregados são capazes de suprir esta lacuna deixada pelas redes tradicionais. Além do mais, este trabalho tem como diferencial aplicar as ferramentas *few-shot learning* ao processamento de linguagem natural (base obtida por *transfer learning*), visto que, na literatura a maioria das aplicações são para base de dados de imagens, tal como, *mini*ImageNet.

Foram implementadas as técnicas de *few-shot learning* Redes Siamesas e Redes *Matching*, sendo seus resultados comparados com o algoritmo KNN. Observou-se que as mesmas obtiveram desempenho satisfatório, conseguindo superar os resultados obtidos com o KNN. Com o aumento de classes a serem treinadas, espera-se que os seus benefícios fiquem ainda mais evidentes. Em relação aos algoritmos FSL, as redes siamesas são mais adequadas para a aplicação pois apresentaram o mesmo resultado que as redes *matching*, porém possuem menos parâmetros e apresentaram menor tempo empregado nas etapas de treino e teste. Além disso, as redes siamesas são menos complexas e mais fáceis de serem parametrizadas e treinadas do que as redes *matching*.

Foi possível observar que há redundância nas características e a utilização de PCA reduziu bastante a dimensão das entradas, permitindo a obtenção de modelos mais parcimoniosos. É importante ressaltar que a aplicação de PCA não diminuiu o desempenho dos classificadores propostos, quando comparado ao desempenho em testes na base de dados sem redução de características.

Visando o melhor desempenho da rede siamesa, podem ser estudadas estratégias para escolha do melhor representante de cada classe, já que o uso do centroide não superou o desempenho da rede com os dados originais. Também é considerada a alteração na estrutura da rede siamesa, colocando mais camadas ou outro tipo de rede neural, além da mudança na escolha e formação dos pares de treinamento. Para as redes *matching* podem ser testadas outras configurações, como uso de *ensembles* e definição de representantes por classes. Também podem ser analisadas opções de FSL da literatura, como redes neurais de grafos.

AGRADECIMENTOS

Os autores agradecem à empresa *Omnilogic Inteligência* e à FAPEMIG por apoiarem este trabalho.

REFERÊNCIAS

Bromley, J., Guyon, I., LeCun, Y., Sackinger, E., and Shah, R. (1994). Signature verification using a "siamese" time delay neural network. *In Advances in neural information processing systems*, 737–744.

e Silva, F.C., de Sousa, R.H., Chaves, A.G.S., Barbosa, B.H.G., and Ferreira, D.D. (2020). Classificador fuzzy-genético aplicado ao processamento de linguagem natural. *Anais do XXIII Congresso Brasileiro de Automática*, 2.

Finn, C., Abbeel, P., and Levine, S. (2017). Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. In *34th International Conference on Machine Learning, ICML 2017*, volume 3, 1856–1868.

Garcia, V. and Bruna, J. (2018). Few-Shot Learning With Graph Neural Networks. In *arXiv*, 1–13.

Jadon, S. and Garg, A. (2020). *Hands-On One-shot Learning with Python Learn*, volume 1.

Koch, G., Zemel, R., and Salakhutdinov, R. (2015). Siamese neural networks for one-shot image recognition. *Proceedings of the 32nd International Conference on Machine Learning*, 37.

Krishnan, A. and Amarthaluri, A. (2019). Large Scale Product Categorization using Structured and Unstructured Attributes. In *Conference on Knowledge Discovery and Data Mining, August 04–08, 2019, Anchorage, Alaska. ACM, New York, NY, USA*, 9.

Krishnan, M. and Dutta, D. (2018). A Study of Effectiveness of Principal Component Analysis on Different Data Sets. In *2017 IEEE International Conference on Computational Intelligence and Computing Research, ICCIC 2017*. IEEE. doi:10.1109/ICCIC.2017.8524329.

Lake, B.M., Salakhutdinov, R., Gross, J., and Tenenbaum, J.B. (2011). One shot learning of simple visual concepts. In *Proceedings of the 33rd Annual Conference of the Cognitive Science Society*, 6.

Ristoski, P., Petrovski, P., Mika, P., and Paulheim, H. (2018). A machine learning approach for product matching and categorization. *Semantic Web*, 9(5), 707–728.

SCHREIBER, J.N.C., BESKOW, A.L., MÜLLER, J.C.T., NARA, E.O.B., SILVA, J.I.D., and REUTER, J.W. (2017). Técnicas de validação de dados para sistemas inteligentes: Uma abordagem do Software SDbayes. In *XVII Colóquio Internacional de Gestão Universitária*, 1–18. doi:10.1017/CBO9781107415324.004.

Trstenjak, B., Mikac, S., and Donko, D. (2014). Knn with tf-idf based framework for text categorization. *Procedia Engineering*, 69, 1356–1364. 24th DAAAM International Symposium on Intelligent Manufacturing and Automation, 2013.

Vinyals, O., Blundell, C., Lillicrap, T., Kavukcuoglu, K., and Wierstra, D. (2016). Matching networks for one shot learning. *Advances in Neural Information Processing Systems*, 3637–3645.

Wang, Y., Yao, Q., Kwok, J., and Ni, L.M. (2020). Generalizing from a Few Examples: A Survey on Few-Shot Learning. *ACM Comput. Surv.*, 1(1), 1–34.

Zheng, Y., Wang, R., Yang, J., Xue, L., and Hu, M. (2019). Principal characteristic networks for few-shot learning. *Journal of Visual Communication and Image Representation*, 59, 563–573.

Classificação *few shot learning* aplicada ao *E-commerce*: uma abordagem com redes siamesas e *ensembles*

Fernanda Costa e Silva[1], **Danton Diego Ferreira**[2], **Bruno Henrique Groenner Barbosa** [2]

[1]Programa de Pós-Graduação em Engenharia de Sistemas e Automação – Universidade Federal de Lavras (UFLA) Caixa Postal 3037 – 37200-000 – Lavras, MG – Brazil

[2]Departamento de Automática – Universidade Federal de Lavras (UFLA)

Caixa Postal 3037 – 37200-000 – Lavras, MG – Brazil

fernanda.silva10@estudante.ufla.br, {danton, brunohb}@ufla.br

Abstract. *E-commerce has generated a large amount of data in text format, which can be used by companies. The proposed application aims to develop a product classifier in different niches of an online sales platform. The application of few shot learning classifiers is proposed, with the use of siamese neural networks and siamese ensembles to determine the similarity between the samples. With the use of only one siamese network, an accuracy of 93.36% was observed. When using an ensemble structure of Siamese networks with 3 representatives of each class, the accuracy ranged between 92,50% and 96.25%. However, both cases require the choosing of the class representative.*

Keywords: *Few shot learning, E-commerce, Ensembles, Siamese networks*

Resumo. *O comércio eletrônico tem gerado grande quantidade de dados em formato textual, que pode ser utilizado pelas empresas. A aplicação proposta objetiva o desenvolvimento de um classificador de produtos nos diferentes nichos de uma plataforma de vendas online. É proposta a aplicação de classificadores few shot learning, com o uso das redes neurais siamesas e ensembles de siamesas para determinar a similaridade entre as amostras. Com o uso de apenas uma rede siamesa foi observado uma acurácia de 93,36%. Ao utilizar uma estrutura de ensemble de redes siamesas com 3 representantes de cada classe a acurácia variou entre 92,50% e 96,25%. Entretanto, ambos os casos necessitam da escolha do representante da classe.*

Palavras-chave: *Few shot learning, Comércio eletrônico, Ensembles, Redes siamesas.*

1. Introdução

Segundo o estudo de Kim (2020), a pandemia do novo coronavírus (COVID19) acelerou o crescimento do número de vendas em lojas de comércio eletrônico *(E-commerce*, em inglês). Com a facilidade de comprar sem ter que sair de casa, somente acessando um *site* ou aplicativo, e os benefícios de ter a sua compra entregue em casa, ter acesso a mais opções de produtos e vendedores, além das diversas formas de pagamento, os consumidores tem optado pela compra *online*.

Com maior utilização das plataformas de compra e venda *online* as empresas tem acesso a mais dados do comportamento de compras de seus consumidores. O estudo destes dados é interessante para as empresas pois elas podem estabelecer melhores relações com os consumidores, entender quais são seus desejos e proporcionar uma melhor experiência de compra e venda (Chen, L. and Wang, F. 2013). Para o entendimento das informações contidas nestes dados, são necessários métodos de processamento de linguagem natural (Silva, F. C. et al. 2020; Haixiang, G. et al. 2017).

As plataformas de *E-commerce*, também conhecidas como *marketplaces*, permitem que diferentes vendedores cadastrem seus produtos e realizem as vendas. O consumidor, por sua vez, pode comprar, de uma só vez, produtos de diferentes vendedores e realizar um único pagamento, que será repassado aos vendedores pela plataforma utilizada. Em um *marketplace*, normalmente, os produtos são divididos em diferentes categorias, para facilitar no processo de busca de um produto e também para exibição de produtos similares em um mesmo local.

Com o crescimento do número de vendedores e o aumento da quantidade de produtos cadastrados nestas plataformas, o processo de classificá-los em suas respectivas categorias se torna custoso para ser feito manualmente, sendo necessário um algoritmo que agrupe os anúncios nas respectivas categorias. Essa classificação deve ser feita considerando as características informadas no título do anúncio e demais campos descritivos, independente da categoria atribuída previamente pelo vendedor ao cadastrar um produto.

Os classificadores tradicionais baseados em redes neurais artificiais não têm bom desempenho ao lidar com classes desbalanceadas, que ocorre quando, no banco de dados, há muitos exemplos de uma ou mais classes e poucos exemplos de outras classes (He, H. and Garcia, E. A. 2009). Neste caso, os modelos podem criar vieses de acordo com os exemplos das classes que são mais representativas na base de dados (Jadon, S. and Garg, A. 2020). Diante disso, a aplicação de modelos com treinamento *one/few shot learning* é uma possibilidade interessante pois esses modelos conseguem aprender a relacionar informações até mesmo das classes pouco representadas, obtendo bom desempenho em bases de dados com classes desbalanceadas.

O uso de classificadores baseados em Redes Siamesas como classificador *few shot learning* vem sendo usado na literatura em aplicações em diversas áreas (Chaves, A. G. S. et al. 2021; Souza, A. de et al. 2019; Koch, G. and Zemel, R. and Salakhutdinov, R. 2015). Os experimentos descritos neste trabalho têm por objetivo propor e comparar o desempenho das Redes Siamesas como classificador *few shot learning* em duas abordagens, sendo a primeira a implementação simples da rede siamesa e a segunda utilizando uma estrutura de *ensembles* (Zhang, C. and Ma, Y. 2014) para testes, abordando o problema do melhor representante para cada classe. O uso da estrutura de *ensembles* permite que seja escolhido mais de um representante por classe, permitindo que seja feita uma representação melhor das características de cada classe quando comparada à estrutura simples, em que cada classe conta com somente um representante.

Este trabalho está organizado da seguinte forma: conceitos teóricos, bem como a metodologia, ferramentas e banco de dados utilizados são apresentados na Seção intitulada “Materiais e métodos”. Os resultados obtidos, bem como as análises dos mesmos são apresentados na Seção “Análise e discussão dos resultados”. Uma recapitulação do trabalho com os principais resultados, conclusões e sugestões dos próximos passos são apresentados na Seção “Conclusão”.

2. Materiais e métodos

2.1. Banco de dados

Os algoritmos utilizados foram implementados em linguagem *Python*, por ser uma linguagem de código aberto, com muitas bibliotecas apropriadas para análise de dados que são constantemente atualizadas.

A base de dados utilizada para os experimentos deste trabalho foi cedida pela empresa *Omnilogic Inteligência*, parceira deste trabalho. Tal base de dados é composta por amostras com textos não estruturados, com o título de ofertas cadastradas em uma plataforma de comércio eletrônico. No total, são 394 amostras, divididas em 34 classes. Porém, trata-se de uma base de dados desbalanceada, sendo que 13 classes possuem apenas uma amostra, conforme apresentado na Tabela 1.

Tabela 1: Informações sobre a distribuição de amostras por classe

Número de amostras por classes	Número de classes
1	13
2	5
3 a 8	10
20 a 60	5
Mais de 100	1

Para os testes descritos neste trabalho foram utilizadas somente as classes com mais de 20 amostras por classe, já que cada classe necessita de amostras o suficiente para que a base de dados seja dividida em bases de treino e teste e que seja feita a escolha das amostras usadas como os representantes de cada classe. As classes usadas foram: Suporte para Galão de Água (113 amostras), Pêndulo (60 amostras), Cesta de Supermercado (51 amostras), Defumador de Ambiente (49 amostras), Limpador Magnético (32 amostras) e Pedra Esotérica (20 amostras).

A base de dados foi dividida em conjunto de treino e conjunto de teste, seguindo a proporção 70/30, sendo que o conjunto de treino ficou com 227 amostras e o conjunto de teste ficou com 98 amostras.

2.2. Redes Siamesas

A Rede Neural Siamesa foi apresentada no trabalho de Bromley, J. et al. (1994), em que foi utilizada para comparar imagens de assinaturas escritas em um *tablet*. A rede siamesa é um tipo de algoritmo *few shot learning* capaz de aprender relações entre as amostras com poucos exemplos de treinamento. É uma rede neural de treinamento supervisionado, que recebe em suas entradas duas amostras diferentes e pode ser usada para medir a similaridade entre elas.

Na etapa de treinamento da rede é necessário formar os pares de amostras e o alvo, já que o aprendizado é supervisionado. Para os pares de amostras da mesma classe, o alvo terá o valor 1, e para os pares de amostras de classes diferentes o alvo terá valor 0. Ao final de todo o processo, a rede siamesa gera um valor de similaridade entre 0 e 1, que pode ser usado para determinar se um par de amostras pertence a uma mesma classe ou não.

A Figura 1 (Chaves, A. G. S. et al. 2021) mostra a arquitetura simplificada de uma rede siamesa. Ela é formada por duas sub-redes idênticas, com a mesma estrutura de camadas e que possuem os mesmos pesos e parâmetros, tendo o mesmo comportamento no tratamento das amostras recebidas.

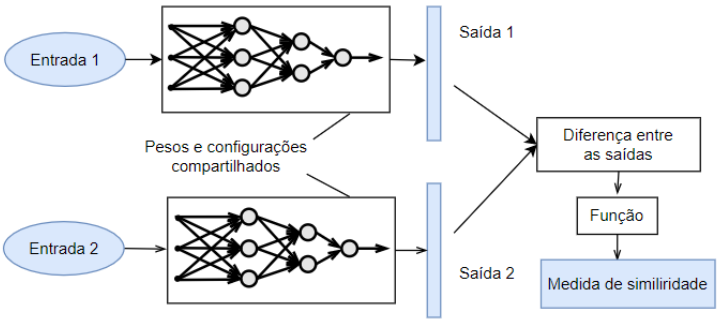


Figura 1: Arquitetura simplificada de uma rede siamesa. Fonte: Chaves, A. G. S. et al. (2021)

As entradas de cada sub-rede são diferentes e, após processá-las, cada sub-rede realiza uma operação que leva essas entradas para um espaço multidimensional de similaridade semântica, criado pela rede siamesa. A saída de cada sub-rede é um vetor com tamanho igual ao número de neurônios da camada de saída da sub-rede.

Nesse espaço de similaridade semântica é calculada a distância entre os dois vetores (um de cada sub-rede) e esse cálculo pode ser feito por um neurônio ou por outra rede neural. Usando esse valor de distância é calculado o valor de similaridade, que é retornado pela rede siamesa. Após o cálculo da perda, os pesos das sub-redes são atualizados via *backpropagation.*

A perda é calculada usando alguma função de penalização, como a de entropia cruzada binária (*binary crossentropy*, em inglês), dada pela Equação 1, usando o valor de similaridade retornado pela rede siamesa e o valor passado como alvo para as duas entradas (x_1 e x_2). Considerando que, no problema analisado neste artigo, os alvos são 0 e 1, Y é o alvo que foi atribuído ao par de amostras (valor real), e P é a probabilidade de o par de amostras pertencer à classe analisada.

$$\text{Perda}(\text{sim}(x_1, x_2), Y(x_1, x_2)) = -Y*\log(P) - (1-Y)*\log(1-P) \qquad \text{Equação (1)}$$

Após ter um modelo de rede siamesa treinado, ele pode ser utilizado para tarefas de classificação *few-shot learning* pois o modelo aprendeu a diferenciar se as amostras recebidas pertencem à mesma classe ou não, sendo que amostras pertencentes a mesma classe terão um valor mais próximo de 1. Usando o valor de similaridade e um limite definido para arredondamento do valor para 0 ou 1 é possível determinar se as amostras recebidas na entrada da rede siamesa são da mesma classe ou não.

A aplicação das redes siamesas como classificador *few-shot learning* foi feita para classificação de novas amostras de produtos que tenham poucas amostras por classe e não são conhecidas pelo classificador em operação.

As sub-redes utilizadas na estrutura da rede siamesa foram propostas com uma única camada intermediária, com 20 neurônios, sendo que este número foi variado de 5 a

100, até chegar no número ótimo. Após a camada intermediária existe uma camada responsável por calcular a distância ponto a ponto das amostras no novo espaço de similaridade semântica. Essa camada utiliza as saídas de cada sub-rede, que tem a dimensão do número de neurônios da camada de saída de cada uma. Em seguida, a saída desta camada alimenta um único neurônio que calcula o valor de similaridade entre as entradas, tendo como resultado um valor entre 0 e 1.

2.3. *Ensembles*

Um *ensemble*, ou comitê de decisão, é entendido como a fusão de vários modelos treinados que, quando combinados, tem o objetivo de melhorar o desempenho de um modelo simples (González, S. and García, S. and Del Ser, J. and Rokach, L. and Herrera, F. 2020). Usando as previsões combinadas, o modelo final faz previsões melhores, conseguindo generalizar suas previsões e considerar informações vindas de partes específicas da base de dados.

No contexto deste trabalho, os *ensembles* serão utilizados para lidar com o problema da escolha do melhor representante de cada classe. Uma abordagem possível para este problema é o uso do centroide de cada classe. Porém, essa abordagem não envolve nenhuma escolha e, sim, a criação de uma nova amostra para ser usada como representante. Utilizando um *ensemble* é possível escolher *n* representantes para cada classe e ter uma representação melhor da classe, considerando diversos pontos de sua distribuição no espaço.

3. Análise e discussão dos resultados

Para montagem dos pares de dados de treinamento enviados para a rede siamesa, para cada amostra da base de dados, foi formado um par da mesma classe e um par de classe diferente, sendo que estes elementos foram escolhidos aleatoriamente.

Para a montagem dos pares de teste, no caso da rede siamesa simples, cada amostra foi escolhida como primeiro elemento e foi selecionada uma amostra de mesma classe e uma amostra de classe diferente para formar pares. todas as amostras restantes da base de dados foram colocadas como segundo elemento do par.

Na abordagem com uso do centroide, foi calculado o centroide de cada classe com os dados de treinamento. Esse elemento foi colocado como o primeiro elemento do par e as amostras da base de dados de teste foram colocadas uma vez como sendo o segundo elemento do par.

No caso do *ensemble* de siamesas, primeiro foram escolhidas as amostras que seriam colocadas como representantes de cada classe, e essas amostras foram retiradas da base de teste. Essas amostras foram “fixadas” como o primeiro elemento da rede siamesa. Os pares foram montados de forma que, para cada representante, todas as outras amostras restantes na base foram colocadas uma vez como o segundo elemento do par. A Figura 2 ilustra o processo de divisão da base, formação dos pares de entrada e cálculo da similaridade entre as entradas.

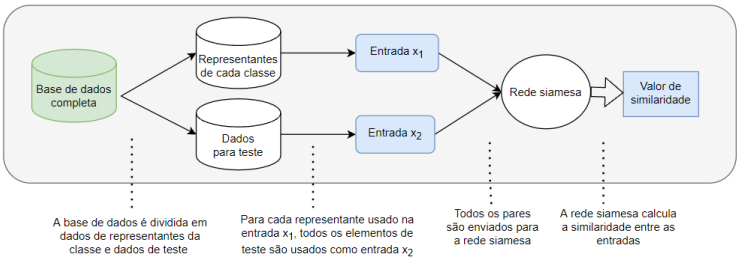


Figura 2: Etapas para obter a similaridade usando estrutura de *ensembles*. Fonte: Do Autor (2021)

A rede siamesa foi treinada com os pares de treinamento e o modelo foi salvo para ser aplicado os dois testes. A escolha dos representantes foi feita de forma aleatória em cada rodada do teste. Foram executadas cinco rodadas de teste para abordagem com *ensemble* para observação da influência da escolha dos representantes de cada classe e os resultados foram dispostos na Tabela 2.

Tabela 2: Resultados com *ensemble* variando o número de representantes

Número do teste	Acurácia (%) 3 representantes	Acurácia (%) 5 representantes
1	95,00	95,59
2	92,50	98,53
3	93,75	94,11
4	96,25	98,53
5	95,00	97,05
Média	94,50 ± 1,27	96,76 ± 1,72

A Tabela 3 exibe os valores dos resultados para os testes com estrutura simples de redes siamesas variando os representantes escolhidos, assim como os dados obtidos com estrutura simples utilizando o centroide como representante da classe e a média dos resultados com estrutura de *ensemble* para os casos com 3 e com 5 representantes. O número de representantes no ensemble foi variado para observar a influência na acurácia do algoritmo quando comparado com a rede siamesa simples, que só utiliza um representante de cada classe.

Tabela 3: Resultados para cada abordagem

Algoritmo	Acurácia (%)
Simples	93,36
Centroide	94,89
Ensemble (3 representantes)	94,50 ± 1,27
Ensemble (5 representantes)	96,76 ± 1,72

Com o uso de *ensemble* o modelo alcança acurácias maiores do que com o uso da estrutura simples. Em relação ao número de representantes, é possível perceber que a acurácia do modelo aumenta com o aumento do número de representantes. Entretanto, é necessário cuidado ao determinar *n,* o número ideal de representantes pois a base de dados é desbalanceada e a escolha de um número de representantes muito alto fará com que não seja possível aplicar esta solução para as classes com menos de *n* elementos sem uma estratégia para criar novas amostras dessas classes que possam ser utilizadas com representantes da classe.

O uso do centroide como representante de uma classe é, na verdade, uma forma de driblar o problema da escolha de representante, pois o centroide é calculado de acordo com os dados de treinamento e nenhuma amostra real da base de dados é escolhida. Porém, pode ser utilizada uma abordagem em que amostra escolhida como representante da classe será aquela que está mais próxima do centroide calculado com os dados de treinamento do modelo.

4. Conclusão

Em situações de classes desbalanceadas, mas que precisam ser consideradas, os algoritmos *few shot learning* podem ser uma escolha interessante pois aprendem a estabelecer relações entre as amostras com poucos exemplos. As aplicações voltadas para o comércio eletrônico normalmente empregam algoritmos que necessitam de um grande número de amostras de treinamento para conseguir um desempenho satisfatório. O uso de métodos *few shot learning* no processamento de linguagem natural é uma aplicação relativamente nova, visto que esses métodos são comumente aplicados na literatura em problemas de análise de imagens, como reconhecimento facial.

As redes siamesas são adequadas para o problema tratado neste artigo pois possuem poucos parâmetros para configuração, são fáceis de parametrizar e apresentam um tempo de treinamento pequeno, considerando a base de dados disponível. Além disso, foi possível diminuir a complexidade das redes utilizadas diminuindo o número de neurônios nas camadas ocultas das redes.

Foram implementadas duas abordagens utilizando as redes siamesas. No primeiro caso, foi utilizada somente uma amostra de cada classe como representante da classe. No segundo caso, foram utilizadas três amostras de cada classe como representante, visando resolver o problema da definição do melhor representante.

Como trabalhos futuros, pode ser feita a alteração na estrutura da rede siamesa, modificando o tipo de rede neural utilizado ou simplesmente adicionar mais camadas ocultas. Também pode ser criado um novo processo para montagem dos pares de treinamento. Outra possibilidade são estudos sobre o número ótimo de representantes para a estrutura de *ensemble* e como esses representantes são escolhidos, podendo ser usada uma métrica de distância e a proximidade da amostra com o centroide da classe.

5. Agradecimentos

Os autores agradecem o apoio financeiro da agência FAPEMIG e da empresa *Omnilogic Inteligência*.

6. Referências

Bromley, J. et al. (1994) “Signature verification using a “siamese” time delay neural network”. In Advances in neural information processing systems, p. 737–744, 1994.

Chaves, A. G. S. et al. (2021) “Extração de entidades de produtos utilizando técnicas de few-shot learning". In: SBAI 2021 - XV Simpósio Brasileiro de Automação Inteligente. Rio Grande, RS, Brasil: Sociedade Brasileira de Automática, 2021.

Chen, L. and Wang, F. (2013) “Preference-based clustering reviews for augmenting e-commerce recommendation". In: Knowledge-Based Systems, v. 50, p. 44–59, 09 2013.

González, S. and García, S. and Del Ser, J. and Rokach, L. and Herrera, F. (2020). “A practical tutorial on bagging and boosting based ensembles for machine learning: Algorithms, software tools, performance study, practical perspectives and opportunities”. In: Information Fusion. doi:10.1016/j.inffus.2020.07.007, 2020.

Haixiang, G. and Yijing, L. and Shang, J. and Mingyun, G. and Yuanyue, H. and Bing, G. (2017) “Learning from class-imbalanced data: Review of methods and applications”. In: Expert Systems with Applications, Volume 73, Pages 220-239, 2017, ISSN 0957-4174,

He H. and E. A. Garcia (2009) "Learning from Imbalanced Data" in IEEE Transactions on Knowledge and Data Engineering, vol. 21, no. 9, pp. 1263-1284, Sept. 2009, doi: 10.1109/TKDE.2008.239.

Jadon, S. and Garg, A. (2020) “Hands-On One-shot Learning with Python Learn”. Birmingham, Mumbai: Packt Publishing, 2020. v. 1. 1–136 p. ISSN 1098-6596. ISBN 9788578110796.

Kim, R. Y. (2020) “The impact of covid-19 on consumers: Preparing for digital sales”. In: IEEE Engineering Management Review, v. 48, n. 3, p. 212–218, 2020.

Koch, G. and Zemel, R. and Salakhutdinov, R. (2015) “Siamese neural networks for one-shot image recognition”. In: Proceedings of the 32nd International Conference on Machine Learning, v. 37, 2015.

Silva, F. C. e et al. (2020). “Classificador fuzzy-genético aplicado ao processamento de linguagem natural". In: Anais do XXIII Congresso Brasileiro de Automática, 2020.

Souza, A. de et al. (2019) “Aplicação de redes neurais siamesas na autenticação de condutores”. In: Anais do 14º Simpósio Brasileiro de Automação Inteligente (SBAI). Ouro Preto, MG, Brasil: SBAI, 2019.

Zhang, C. and Ma, Y. (2014) “Ensemble Machine Learning: Methods and Applications”. Springer New York, ISBN 1489988173, 9781489988171