Universidade Estadual de Londrina

WALLACY SEBASTIAN APARECIDO JERONIMO DE ALMEIDA

ALGORITMO INTELIGENTE PARA OTIMIZAÇÃO DE BUSCAS EM FERRAMENTAS DE E-COMMERCE

LONDRINA

2024

WALLACY SEBASTIAN APARECIDO JERONIMO DE ALMEIDA

ALGORITMO INTELIGENTE PARA OTIMIZAÇÃO DE BUSCAS EM FERRAMENTAS DE E-COMMERCE

Trabalho de Conclusão de Curso apresentado ao curso de Bacharelado em Ciência da Computação da Universidade Estadual de Londrina para obtenção do título de Bacharel em Ciência da Computação.

Orientador: Prof(a). Dr(a). Helen Cristina de Mattos Senefonte

LONDRINA

2024

Ficha de identificação da obra elaborada pelo autor, através do Programa de Geração Automática do Sistema de Bibliotecas da UEL

Sobrenome, Nome.
Título do Trabalho : Subtitulo do Trabalho / Nome Sobrenome. - Londrina, 2017.
100 f. : il.

Orientador: Nome do Orientador Sobrenome do Orientador.
Coorientador: Nome Coorientador Sobrenome Coorientador.
Dissertação (Mestrado em Ciência da Computação) - Universidade Estadual de Londrina, Centro de Ciências Exatas, Programa de Pós-Graduação em Ciência da Computação, 2017.
Inclui bibliografia.

1. Assunto 1 - Tese. 2. Assunto 2 - Tese. 3. Assunto 3 - Tese. 4. Assunto 4 - Tese. I. Sobrenome do Orientador, Nome do Orientador. II. Sobrenome Coorientador, Nome Coorientador. III. Universidade Estadual de Londrina. Centro de Ciências Exatas. Programa de Pós-Graduação em Ciência da Computação. IV. Título.

WALLACY SEBASTIAN APARECIDO JERONIMO DE ALMEIDA

ALGORITMO INTELIGENTE PARA OTIMIZAÇÃO DE BUSCAS EM FERRAMENTAS DE E-COMMERCE

Trabalho de Conclusão de Curso apresentado ao curso de Bacharelado em Ciência da Computação da Universidade Estadual de Londrina para obtenção do título de Bacharel em Ciência da Computação.

BANCA EXAMINADORA

Orientador: Prof(a). Dr(a). Helen Cristina de Mattos Senefonte
Universidade Estadual de Londrina – UEL

Prof. Dr. Vitor Valério de S. Campos
Universidade Estadual de Londrina – UEL

Prof(a). Dr(a). Vanessa Matias Leite
Universidade Estadual de Londrina – UEL

Londrina, 6 de maio de 2024.

Este trabalho é dedicado às crianças adultas
que, quando pequenas, sonharam em se
tornar cientistas.

AGRADECIMENTOS

Primeiramente, meus agradecimentos principais são para Deus, pois diante de todas as circunstâncias, sejam difíceis ou não, Ele sempre foi onipresente e me concedeu força, saúde, ânimo e paz para continuar a graduação. Aos meus familiares que me apoiaram, especialmente à minha mãe, Elizangela Aparecida Jeronimo, que me proporcionou apoio, recursos e motivação para que eu continuasse a minha graduação. À minha namorada, Talita Lopes Buranello, que com amor, contribuiu excepcionalmente para que eu continuasse consistente e obstinado, me incentivou com alegria e me trouxe paz. À minha orientadora, Helen Cristina de Mattos Senefonte, que se disponibilizou durante o desenvolvimento deste trabalho, com paciência, para que o mesmo fosse finalizado. À Universidade Estadual de Londrina (UEL) e ao Departamento de Computação (DC) por todos os recursos disponibilizados e professores devidamente capacitados.

"Sacrifícios agradáveis a Deus são o espírito quebrantado; coração compungido e contrito, não o desprezarás, ó Deus. (Bíblia Sagrada, Salmos 51, 17))

ALMEIDA, W. S. A. J.. **Algoritmo inteligente para otimização de buscas em ferramentas de e-commerce**. 2024. 38f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) – Universidade Estadual de Londrina – UEL, Londrina, 2024.

RESUMO

A utilização de sistemas ERPs (*Enterprise Resource Planning*), desde a sua criação, tem favorecido o aumento da produtividade nas empresas, desde microempresas até empresas de grande porte que possuem um grande número de dados para gerir, e há um aumento constante em sua busca. Atrelado a isso, muitos empresários do setor de varejo recorreram a esta solução para gerir melhor as suas vendas e aumentar a comercialização de seus produtos através de plataformas online, como *e-commerces* e *marketplaces*. No entanto, para que suas vendas sejam bem-sucedidas, é necessário que os produtos estejam em outras plataformas de modo ágil e efetiva. Nesta perspectiva, este estudo tem o objetivo de proporcionar uma solução automatizada para a categorização de produtos utilizando o nome destes, visando facilitar a busca de categorias de produtos no *marketplace* Mercado Livre.

Palavras-chave: Mercado Livre, *Paragraph Vector*, *Doc2Vec*, Categorização de produtos, PLN

ALMEIDA, W. S. A. J.. **Intelligent algorithm for search optimization in e-commerce tools**. 2024. 38p. Final Project (Bachelor of Science in Computer Science) – State University of Londrina, Londrina, 2024.

ABSTRACT

Enterprise Resource Planning (ERP) system was created in 1990s, and since their creation, the use of them has enhanced the productivity in many organizations, ranging from small to large companies, even those with a vast amount of data to manage, and there is a constant growth in their adoption. Related to this, many retail's entrepreneurs take advantage from ERPs to manage and increase their sales through online platforms, such as e-commerce websites and marketplaces. However, to be successful in their sales, the products need to be in other platforms in an fast and effective way. From this perspective, this study aims to provide an automated solution to categorize products based on their names, enhancing the search for product categories at Mercado Livre marketplace.

Keywords: Mercado Libre, Paragraph Vector, Doc2Vec, Product categorization, NLP

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ABNT	Associação Brasileira de Normas Técnicas
BNDES	Banco Nacional de Desenvolvimento Econômico e Social
IBGE	Instituto Nacional de Geografia e Estatística
IBICT	Instituto Brasileiro de Informação em Ciência e Tecnologia
NBR	Norma Brasileira

SUMÁRIO

1 INTRODUÇÃO

A Inteligência Artificial (IA) ganhou importância nos últimos anos perante o público em geral, através de produtos como *ChatGPT* da *OpenAI* e *Bard* do *Google*. Conforme explica o artigo pela *World Economic Forum*, *COVID-19 increased the use of AI. Here's why it's here to stay* [1], o surgimento de produtos como estes são resultados do incentivo dado pelas empresas das maiores áreas econômicas do mundo, como as áreas da saúde, educação e varejo, para que reduzissem custos e melhorassem o trabalho remoto durante a pandemia. Após este período, ainda há um crescente uso de IAs no dia-a-dia das pessoas, não só através dos produtos mencionados como em outros produtos do mercado, para reconhecimento de imagens e vídeos, processamento de textos, automatização de tarefas, análises no mercado financeiro, dentre muitas outras utilizações [1].

Quando observamos mais de perto a área do varejo, vemos o quanto a pandemia influenciou no crescimento dos *e-commerces* (termo usado para definir um comércio eletrônico). Durante este período, as ruas estavam com circulação restrita, os estabelecimentos possuíam limite de pessoas e muitas empresas do varejo necessitaram de uma mudança estratégica de venda, levando estas a vender pela Internet, seja através de *marketplaces* - modelo de negócio que possibilita a venda através de uma plataforma utilizada por vários lojistas -, de lojas virtuais, de redes sociais ou de aplicativos como *WhatsApp* [2]. E com o aumento do uso de *e-commerces*, naturalmente aumentaram também os problemas enfrentados pelos lojistas e pelos usuários que compram.

Desde que os sistemas *ERP* (*Enterprise Resource Planning*, ou Sistema de Gestão Integrado) surgiram na década de 90, as empresas têm recebido bem a solução para gerir os negócios [3], e com os *e-commerces* não foi diferente. Muitos *e-commerces* recorreram aos *ERPs* para gerenciar seus negócios, buscando não só ferramentas básicas (como controle de estoque, preços, entre outros) para os seus produtos, mas também uma forma de potencializar suas vendas através de integrações com outros sistemas [4]. Algumas das integrações de interesse foram os *marketplaces*: a empresa coloca seu produto à venda em seu *e-commerce*, e ao mesmo tempo em algum *marketplace*, sendo da responsabilidade do *ERP* gerenciar automaticamente os estoques das duas plataformas.

Junto às necessidades de automação devido ao grande volume de produtos de algumas empresas, surgiu também a necessidade de ferramentas e técnicas que trouxessem mais praticidade ao transmitir produtos para *marketplaces*. Em grande parte das vezes, os produtos necessitam ser classificados em categorias, como "*Smartphones*", "Papelaria"e "Higiene", para que se adequem ao *marketplace* escolhido. No entanto, as categorias que um lojista atribui podem ter nomes diferentes das categorias de *marketplace*, sendo necessária a intervenção humana para categorizar manualmente. O Mercado Livre possui

ferramentas onde, através da busca do nome de um produto, obtém-se sugestões de categorias correspondes. Mas muitas vezes - devido à complexidade da linguagem humana, ou até uma escrita incorreta - as sugestões são incorretas, levando à categorização manual de produtos e impedindo a automação desta tarefa.

A necessidade de representar o nome de um produto para que um computador entenda e categorize de forma correta é uma tarefa da área de Redes Neurais, mais especificamente do Processamento de Linguagem Natural, uma subárea que possui um conjunto de técnicas para lidar com a linguística utilizando a máquina, sendo o PLN a área que possibilita a classificação de textos e sentenças em linguagem humana.

Este estudo foi focado em empresas que possuem *e-commerces*, e visa facilitar a categorização de produtos baseado nos nomes destes, para que integrações com *ERPs* e o *marketplace* Mercado Livre [5] tenham maior precisão ao classificar produtos em categorias do Mercado Livre.

Os objetivos deste trabalho são: Extrair informações de produtos e categorias, obtidas através do Mercado Livre, para entender a estrutura destes dados; e utilizar algoritmos e técnicas de PLN para processar estes dados e obter um algoritmo que preveja categorias do Mercado Livre, tendo como entrada somente os nomes dos produtos.

As próximas seções descrevem como o estudo aconteceu e quais recursos foram utilizados. A seção 2, que vem a seguir, apresenta uma série de conceitos, técnicas e algoritmos que foram úteis para elucidar o estudo e executar o objetivo proposto, como os conceitos de Redes Neurais e Processamento de Linguagem Natural, os algoritmos *Word2Vec* e *Paragraph Vector*, e as técnicas para representação de palavras e métrica de similaridade. Já a seção 3 apresenta os meios e a forma como as categorias dos produtos foram buscadas e tratadas antes de utilizá-los no algoritmo proposto.

A seção 4 mostra como os dados foram preparados e como o algoritmo escolhido foi utilizado para atingir o objetivo proposto, e os resultados obtidos com a realização dos testes experimentais estão na seção 5. Por fim, a seção 6 possui discussões acerca da importância dos resultados para as empresas e o quanto isso impacta nos processos de seus negócios.

2 FUNDAMENTAÇÃO TEÓRICA

A relação entre as palavras torna um texto menos ou mais compreensível, conforme a vivência que temos. Ao representar esta relação a uma máquina, há de se considerar que não há experiências vividas, apenas as palavras disponíveis em um texto. Para isso, existem um conjunto de técnicas que permitem representar palavras e textos em uma máquina de acordo com a linguagem humana.

Nesta seção, apresentaremos: os tipos de aprendizado que uma máquina pode obter e como podemos representar este aprendizado; de quais maneiras podemos representar palavras; quais algoritmos podemos utilizar para compreenda um texto, classifique-o e ainda o produza; e como podemos medir a classificação de um texto, já que este estudo tem como propósito classificar produtos em categorias do Mercado Livre.

2.1 Aprendizado de Máquina

O aprendizado de máquina é uma área da inteligência artificial que reúne técnicas e algoritmos com o objetivo de imitar o aprendizado humano em uma máquina, baseado nas observações que são passadas. Esta área possui diversos algoritmos de aprendizado, sendo estes divididos em três categorias [6]:

- **Aprendizado supervisionado**: este tipo de aprendizado tenta inferir padrões de acordo com um conjunto de dados já rotulados, utilizados para treino, como por exemplo aprender sobre textos classificados como "Notícia", "Esportes", "Receita" e "Romance", e classificar novos textos de acordo com o que já foi aprendido.
- **Aprendizado não supervisionado**: quando não se tem um conjunto de dados rotulados, o aprendizado identifica padrões nos dados e estes são agrupados conforme suas semelhanças, possibilitando assim novas detecções de padrões. Por exemplo, é possível realizar o aprendizado de textos de "Notícia" sem classificá-los, obtendo assim agrupamentos não vistos antes, como "Notícia de Política", "Notícia de Famosos" e "Notícia de Economia".
- **Aprendizado semi supervisionado**: acontece quando se possui apenas alguns dados rotulados. É mais interessante utilizar aprendizado não supervisionado e semi supervisionado quando a quantidade de dados é grande, pois pode ser trabalhoso rotular cada observação.

As redes neurais são uma área do aprendizado de máquina, onde as técnicas de aprendizado são inspiradas na maneira como o cérebro funciona, e busca representar os

dados de forma correta, de acordo com a realidade, sendo assim muito utilizada para predições.

2.2 Processamento de Linguagem Natural (PLN)

A representação da linguagem humana sempre foi necessária para que o ser humano pudesse se comunicar através de outros meios que não fossem a fala. A escrita surgiu como um meio de registrar a fala do ser humano, tornando assim mais eficiente a transmissão de ideias. As tecnologias evoluíram, novas linguagens foram criadas e ferramentas foram desenvolvidas para facilitar a vida humana. Algumas destas ferramentas ainda não se encaixam totalmente com a realidade da linguagem humana, pois as linguagens evoluem a cada dia, e muitas das palavras presentes nas linguagens são altamente ambíguas, tornando assim a representação destas palavras uma tarefa complexa [6]. O PLN entra com o objetivo de solucionar problemas que envolvem linguagem natural, de modo que as palavras ou textos sejam traduzidos para uma máquina entender significados, contextos, diálogos, e se comunique traduzindo para a linguagem humana novamente [7] [6].

O PLN também abre possibilidades para ser utilizado na compreensão de textos, classificação de documentos, análise de sentimentos, entre outros, e abrange áreas do conhecimento como a linguística, o aprendizado profundo (que também é chamado de redes neurais), o aprendizado de máquina e a inteligência artificial [7], conforme apresentado na Figura 1. Entretanto, é desafiador. Além dos desafios de lidar com uma linguagem ambígua e com entradas de tamanhos variados em um sistema sem regras definidas, não temos clareza ao definir, por exemplo, a similaridade entre palavras, a representação destas e dos seus diversos significados, a representação de contextos e até a criação de sentenças baseadas em outros documentos.

Tendo em vista estes desafios e os esforços feitos utilizando tecnologias cada vez mais robustas para realizar grandes tarefas, esta área pode ser definida como um conjunto de técnicas e algoritmos que tem como entrada, ou produz como saída, dados que representam a linguagem natural [6]. Algumas dessas técnicas e algoritmos foram explorados neste trabalho, a fim de que o objetivo principal fosse alcançado. Técnicas para a representação de palavras e para metrificar similaridades entre palavras e textos foram exploradas a seguir, como também algoritmos para processamento de sentenças e textos.

2.3 Representação de palavras

Um dos pilares fundamentais do PLN é a representação de palavras, pois esta permite que sistemas computacionais processem textos de uma maneira semelhante à linguagem humana. A linguagem humana é simbólica e discreta, até mesmo em seus caracteres: discreto pois cada palavra tem o seu significado, e simbólico porque os signi-

Figura 1 – As diversas áreas que o PLN abrange - Fonte: Elaborado pelo autor

ficados são externos, e cabe a cada pessoa interpretar as palavras por conta própria, de acordo com as experiências de vida que esta possui [6]. A linguagem humana também é composicional: é possível formar palavras combinando letras, formar frases combinando palavras e textos combinando frases. Somando estas duas propriedades, podemos dizer que a linguagem humana é esparsa: nem toda combinação de letras e palavras irá resultar em algo que possua significado. Representar as nuances da linguagem humana requer um conjunto de técnicas, como também a evolução destas técnicas para que tenhamos cada vez mais representações fiéis à realidade linguística na qual vivemos.

2.3.1 *One-hot*

Tradicionalmente a representação de uma palavra, extraída de uma frase ou texto qualquer, era feita com um vetor do tamanho do vocabulário - conjunto de palavras - do texto, onde cada posição do vetor representa uma palavra do vocabulário. A Figura 2 exemplifica esta representação: ao lado esquerdo existem algumas palavras retiradas da frase "O tempo voa quando nos divertimos", e à direita de cada palavra há um vetor com todas as palavras da frase. O vetor é preenchido com o valor 0 para todos os elementos, e o elemento que representa a palavra assume o valor 1, sendo este elemento chamado de *hot*. Esta representação é chamada de *one-hot* [8].

O tempo voa quando nos divertimos.

	o	tempo	voa	quando	nos	divertimos
tempo	0	1	0	0	0	0
divertimos	0	0	0	0	0	1
voa	0	0	1	0	0	0

Figura 2 – Representação *one-hot* - Fonte: Elaborado pelo autor

2.3.2 *Bag-of-Words (BoW)*

Diferente da representação *One-hot* (ou *Bag-of-Single-Word*), que possibilita a leitura de uma palavra como um vetor com um dos elementos tendo o valor 1, o *BoW* permite que uma frase ou um parágrafo sejam representados por um vetor [6]. O vetor tem o tamanho do vocabulário também, com cada posição representando uma palavra, e cada palavra presente na frase é representada preenchendo o elemento correspondente do vetor com o valor 1, e as palavras não contidas são preenchidas com o valor 0, como mostra a Figura 3. Essa técnica permitiu que tarefas mais complexas de PLN fossem realizadas, como a identificação de relação entre as palavras, e assim permitiu também a captura de contextos.

1. O gato estava em cima do cachorro.
2. O cachorro estava em cima do gato.
3. O pássaro voou por cima do cachorro.

	o	gato	estava	em	cima	do	cachorro	pássaro	voou	por
1.	1	1	1	1	1	1	1	0	0	0
2.	1	1	1	1	1	1	1	0	0	0
3.	1	0	0	0	1	1	1	1	1	1

Figura 3 – Representação *bag-of-words* - Fonte: Elaborado pelo autor

2.3.3 *Word Embeddings*

Anteriormente, como mostra a representação *BoW*, as palavras eram representadas por números discretos, e assim as relações entre as palavras formavam um grafo que dava o contexto local das palavras. Em uma representação distribuída, onde o contexto vai além do próprio vetor de palavras, cada palavra é representada por um vetor de valores, onde esses valores são "padrões de ativação", e a sua semântica e o contexto de uma palavra são identificados pelos valores desse vetor, conforme a Figura 4. Esse vetor possibilita também capturar as similaridades entre as palavras, calculando e comparando a distância entre diferentes vetores. Dessa forma, é possível inferir que palavras que tendem

a aparecer em contextos similares tendem a ter o mesmo significado. Por exemplo, as palavras "celular" e "*smartphone*" aparecem em um contexto de ligações telefônicas, ou seja, com o mesmo conjunto de palavras, então tem significados (ou vetores) semelhantes, mas as palavras "celular" e "macarrão" aparecem em contextos diferentes, então possuem significados distantes e, portanto, diferentes. Essas formas de representar palavras como padrões de ativação, e de analisar os significados dessas palavras através dos lugares do texto em que elas aparecem, levam a uma nova forma de criar algoritmos e de resolver problemas do mundo real.

frio	1,05	0,2	3,6	1,15
gelado	1,14	1,3	3,33	1,14
água	1,01	0,7	3,32	1,15

Figura 4 – Representação *Word Embeddings* - Fonte: Elaborado pelo autor

2.4 Algoritmos de previsão

Para produzir sentenças, obter significados, interpretar e classificar textos, demanda-se a existência de algoritmos capazes de lidar com estes problemas. Os algoritmos a seguir são de aprendizado não supervisionado e se inspiram em *Word Embeddings*, e tanto o *Word2Vec* quando o *Paragraph Vector* são bem eficientes no que se propõem e altamente escaláveis.

2.4.1 *Word2Vec*

Desenvolvido por Tomáš Mikolov et. al. [9] em 2013 e classificado como um algoritmo popular que utiliza *Word Embeddings*, o *Word2Vec* introduz uma técnica para a representação de palavras envolvendo as palavras ao redor da palavra alvo, considerando o contexto do alvo, seja para prever as palavras ao redor da palavra central quanto para prever a palavra central utilizando as palavras ao redor dela. A Figura 5 demonstra um exemplo do funcionamento deste algoritmo. Para isso, considerando um grande volume de texto, cada palavra possui um valor numérico de acordo com sua relação contextual. Comparado à representação *one-hot*, ao considerar o contexto das palavras temos menos limitações para resolver problemas mais complexos, possibilitando uma representação mais rica e precisa em um PLN.

2.4.1.1 Modelos para representação

O *Word2Vec* é dividido em dois modelos:

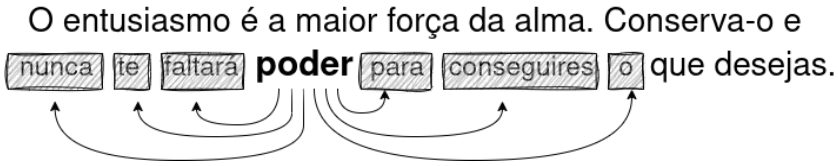
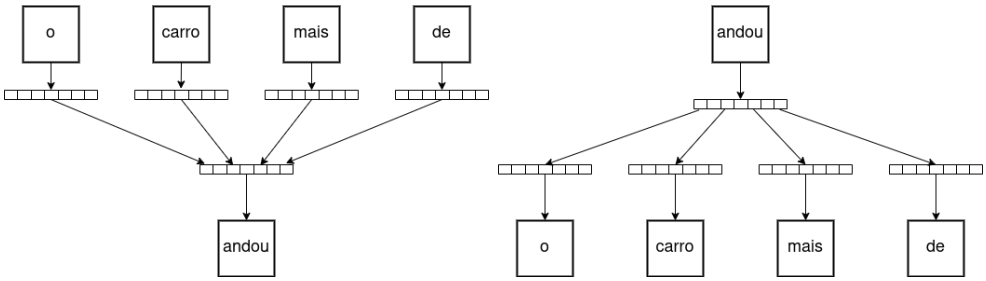


Figura 5 – Representação de como o *Word2Vec* considera uma janela de três palavras e o contexto da palavra central - Fonte: Elaborado pelo autor



(a) Modelo *CBOW* (b) Modelo *Skip-gram*

Figura 6 – Comparação entre os modelos *CBOW* e *Skip-gram* - Fonte: Elaborado pelo autor

CBOW (Continuous Bag-of-Words): a Figura 6(a) exibe o funcionamento deste modelo. o objetivo é prever uma palavra em uma frase a partir das palavras ao redor. Através da identificação de padrões de ocorrência entre as palavras, a rede neural obtém as relações semânticas entre as palavras ao redor e a palavra alvo [9].

Skip-gram: este modelo visa prever as palavras ao redor a partir de uma palavra, de acordo com o exemplo apresentado na Figura 6(b). Essa abordagem permite que o *Word2Vec* aprenda a identificar as palavras mais relevantes para a palavra central, obtendo informações sobre seu possível significado e seu uso no texto [9].

2.4.1.2 Utilizações e considerações

Além disso, o *Word2Vec* pode ser utilizado em diversas áreas do PLN, para análise de sentimentos, classificação de textos, resumo de textos e até traduções automáticas. Mas é importante notar que o *Word2Vec* também tem suas limitações, como a eficiência computacional: enquanto o *Skip-gram* possui uma representação melhor das palavras, o *CBOW* é mais rápido para treinar o modelo [6] [9]. O ruído também deve ser tratado: alguns textos podem possuir palavras que só aparecem uma vez, como palavras incorretas e palavras sem muita relevância para o texto. É crucial realizar um tratamento para o texto antes de utilizá-lo para treinar um modelo.

2.4.2 *Paragraph Vector*

A representação de palavras utilizando a técnica *Word Embeddings* é simples e eficiente, por isso tornou o *Word2Vec* tão popular. Mas ainda possui muitas desvantagens,

como por exemplo a perda da ordem das palavras em uma frase. Duas frases com as mesmas palavras podem ser consideradas frases iguais, ou seja, com o mesmo significado. Outro exemplo é inferência incorreta da similaridade entre as palavras. Por mais que a técnica *Word Embeddings* capture a similaridade das palavras baseadas em seu contexto, ela não captura os sinônimos. Então os sinônimos "fria" e "gelada" que são mais próximos na linguagem humana, utilizando *Word Embeddings*, "água", "fria" e "gelada" têm a mesma distância em relação à similaridade.

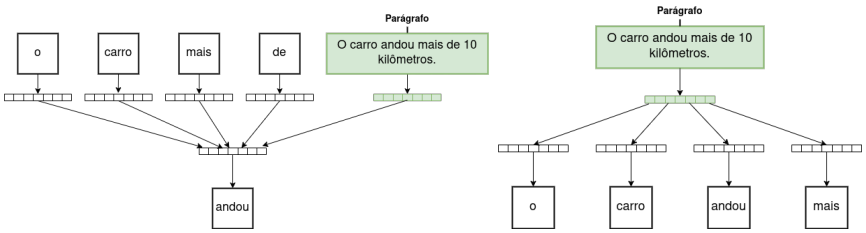
O algoritmo *Paragraph Vector*, desenvolvido por Quoc V. Le [10] em 2014, resolve alguns destes problemas, pois consegue ser aplicado em textos de tamanhos variados, sejam frases até textos grandes, obtendo assim um contexto daquele texto. Além disso, armazena a ordem das palavras, mesmo em um pequeno contexto, sabendo então diferenciar duas frases que utilizam as mesmas palavras. Inspirado no *Word2Vec*, cada palavra é representada por um vetor, com o adicional de que cada parágrafo também é representado por um vetor, e assim armazenando o contexto.

2.4.2.1 Modelos para representação

Da mesma forma que o *Word2Vec*, o *Paragraph Vector* possui dois modelos:

Distributed Memory Model of Paragraph Vectors (PV-DM): este modelo prevê de acordo com as palavras ao redor. Além das palavras, o parágrafo também é considerado como vetor, e é utilizado na previsão, concatenando todos os vetores e prevendo uma palavra, conforme exposto na Figura 7(a). Este vetor funciona como uma memória, que guarda todas as palavras e sabe o que está faltando [10].

Distributed Bag of Words of Paragraph Vector (PV-DBOW): já este modelo, como mostra a Figura 7(b), ignora as palavras de contexto e tenta prever palavras utilizando somente o vetor de parágrafo [10].



(a) Modelo *PV-DM* (b) Modelo *PV-DBOW*

Figura 7 – Comparação entre os modelos *PV-DM* e *PV-DBOW* - Fonte: Elaborado pelo autor

2.4.2.2 Utilizações e considerações

Como o *Word2Vec*, o *Paragraph Vector* pode ser utilizado também para análise de sentimentos, classificação de textos, resumo de textos e traduções automáticas. Para a

maioria dos casos, o *PV-DM* funciona melhor que o *PV-DBOW*, no entanto é recomendado que se utilize *PV-DBOW* combinado com *PV-DM*, pois se obtém melhores resultados, de acordo com o desenvolvedor [10]. O funcionamento conjunto deste dois algoritmos está exemplificado na Figura 8.

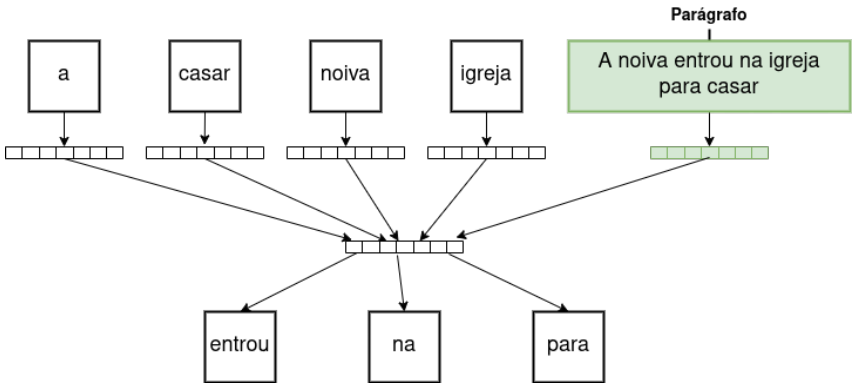


Figura 8 – Representação da combinação *PV-DM PV-DBOW* - Fonte: Elaborado pelo autor

2.5 Métricas de similaridade

Para a classificação de textos, é importante que a máquina saiba diferenciar um texto do outro, ou identificar textos semelhantes que pertençam à mesma classe. Através de técnicas de comparação, podemos dizer o quão distante dois textos estão entre si. Comparar através da linguagem natural pura é desafiador, então torna-se interessante utilizar representações de palavras e textos que se adaptem aos objetivos finais, como identificar similaridades. Especificamente para resolver problemas de classificação de textos, onde as palavras e os textos são representados como vetores numéricos, torna-se interessante utilizar a técnica a seguir.

2.5.1 Similaridade de Cosseno

A Similaridade de Cosseno, é uma técnica amplamente usada para identificar a similaridade entre dois vetores [11], obtendo um valor que varia entre -1 e 1, onde -1 significa que um vetor é totalmente oposto ao outro, e 1 significa que um vetor é totalmente similar ao outro. Pode-se obter também o valor 0, onde não há similaridade [8].

Para obter o valor de similaridade, utilizamos a seguinte equação:

$$\text{sim}_{\cos}(x, y) = \frac{\sum x_i y_i}{\sqrt{(\sum x_i^2)(\sum y_i^2)}} \tag{2.1}$$

onde

x e y são vetores numéricos,

x_i e y_i são elementos do vetor,
e $i \in$ N.

3 CATEGORIZAÇÃO DE PRODUTOS

Ao categorizar um produto no Mercado Livre, o produto é associado também às categorias "pai", e portanto, pode ser classificado em uma categoria que abrange várias subcategorias e vários tipos de produto.

De acordo com o que foi capturado na *API (Application Programming Interface)* do Mercado Livre pelos *endpoints* <https://api.mercadolibre.com/sites/MLB/categories> e <https://api.mercadolibre.com/categories/$CATEGORIA_ID> - onde *$CATEGORIA_ID* é o identificador de uma categoria - [5], e foi analisado e processado, as categorias são representadas em árvore, tendo as categorias do topo como raízes, e no momento em que foi escrito este trabalho, o grau máximo desta árvore é 7. A Figura 9 exibe os níveis das categorias do Mercado Livre.

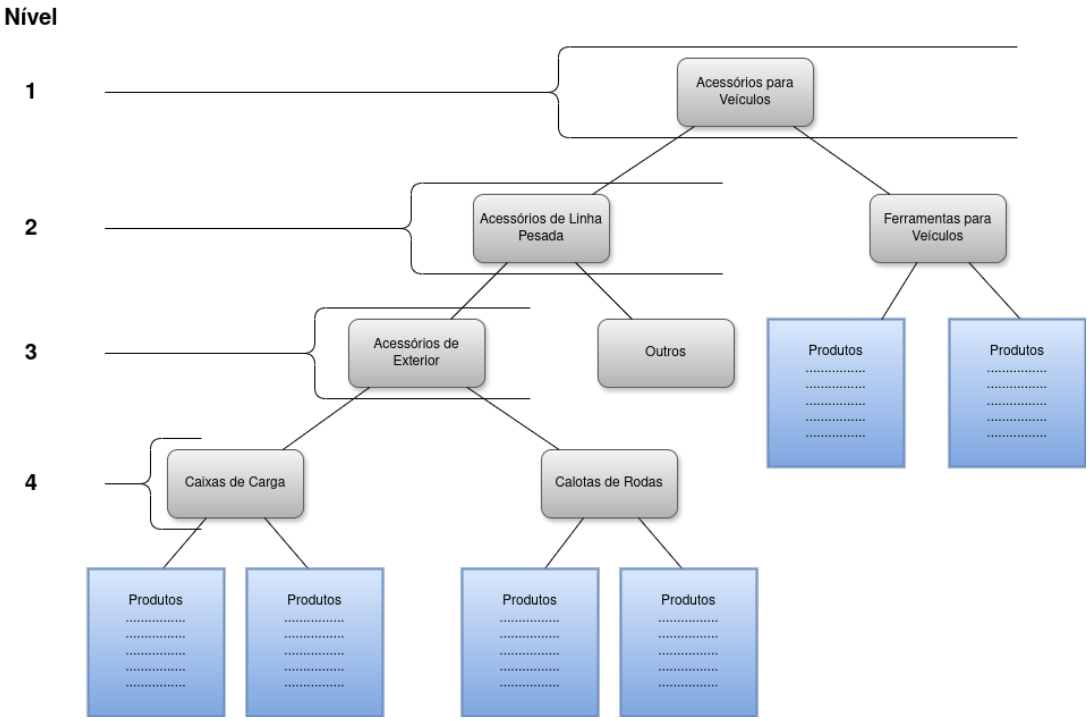


Figura 9 – Representação das categorias do Mercado Livre - Fonte: Elaborado pelo autor

O processo de categorização de produtos envolve um *dataset* com nomes de produtos, onde cada produto está associado com o identificador de uma categoria e de todas as categorias acima desta. Posteriormente, se realiza o pré-processamento dos dados, eliminando possíveis ruídos e normalizando o *dataset*. Em seguida, o treinamento é efetuado utilizando o algoritmo *Paragraph Vector*.

3.1 Composição dos dados

Como foi dito, o *dataset* de entrada do algoritmo possui as informações dos produtos combinadas com as informações das categorias associadas a cada produto. Na tabela 1, há uma demonstração de como os dados são representados.

Tabela 1 – Tabela com os campos utilizados

Nome do produto	**Categoria**	**Categoria n-1**	**...**	**Categoria n-5**	**Categoria n-6**	**Categoria n-7**
Kit 20 Botão Painel Instrumentos Volare V5 V6 V8 W6 W7 W8 W9	MLB456390	MLB5672	...	MLB456390	NULO	NULO
Cozinha Completa 5 Pç C/ Armário Balcão Mp3715 Veneza Gb Pta Cor Preto	MLB439421	MLB1574	...	MLB454701	MLB439421	NULO
Receptor Sdr Usb Dvb-t Sdr+dab+fm Rtl 2832u + R820t2 Uhf Vhf	MLB100074	MLB1648	...	NULO	NULO	NULO

Além das informações já expostas na tabela, é importante salientar que:

- O tamanho do *dataset* é a quantidade de produtos disponíveis para treino, ou seja, cada observação possui um produto diferente;
- Podem haver produtos com nomes repetidos, apesar de cada produto possuir um identificador único, não exposto no *dataset* para treino;
- Cada nome de produto é associado a uma categoria especifica e às categorias superiores a esta;
- Nem todos os níveis de categorias estão preenchidos para cada produto, pois nem todas as categorias específicas estão no sétimo nível.

3.2 Pré-processamento de Dados

Observando novamente a tabela 1, vemos que o nome do produto é proveniente do preenchimento livre por parte dos vendedores da plataforma Mercado Livre, e possui palavras abreviadas, números e caracteres especiais, o que aumenta o tempo de treinamento e diminui a precisão das previsões. Portanto, antes do treinamento é necessário remover estas palavras que aumentam o vocabulário do modelo e causam ruído.

3.2.1 Padronização de nomes de produtos

Para padronizar o nome dos produtos, são realizados alguns procedimentos, nesta ordem:

1. o nome dos produtos são convertidos para letras minúsculas;
2. os acentos são removidos, para a diminuição do vocabulário;
3. palavras abreviadas, como "c/" e "p/", correspondentes a "com" e "para", e palavras que não agregam significado (*stop words*) como artigos, conjunções e preposições são removidas, pois atrapalham na composição do vocabulário de treinamento;
4. números e caracteres especiais também são removidos.

3.2.2 Normalização e remoção de possíveis ruídos

Além dos caracteres que causam ruído, sabe-se que um *dataset* contendo informações de produtos do Mercado Livre pode conter inconsistências, pois os vendedores são livres para escolher as categorias dos produtos. Portanto, alguns produtos podem estar em duas categorias diferentes, ocasionando assim a perda do significado real da categoria. Também é possível que uma categoria possua poucos produtos, gerando inconsistências na fase de treinamento.

Dito isso, é interessante realizar mais dois processos, que são descritos a seguir:

- Remoção de produtos em categorias com quantidades menores que 50 produtos (normalização);
- Remoção de nomes de produtos duplicados que pertencem, ao mesmo tempo, à categorias diferentes, pois produtos desse tipo podem gerar ambiguidade na classificação, e portanto, é tratado como um ruído.

3.3 Treinamento

Após as etapas de pré-processamento, os nomes dos produtos passam por um processo de "tokenização", adaptado para que o algoritmo *Paragraph Vector* processe os dados. Então, utiliza-se este *dataset* para o treinamento do *Paragraph Vector*.

Cada nome de produto, representado por um vetor de *tokens*, é tratado como uma sentença, e é marcado com o identificador de categoria associado a ele, tratado como identificador de documento, possibilitando assim um treinamento supervisionado.

No início do treinamento, é escolhido um tamanho de vetor para representar as associações de palavras, e então o algoritmo *Paragraph Vector* realiza uma inicialização aleatória preenchendo o vetor com número entre $[-\frac{1}{2d}, \frac{1}{2d}]$, o mesmo utilizado pelo *Word2Vec*, onde d é o número de dimensões.

Após isso, se inserem os nomes dos produtos "tokenizados", associados com suas respectivas categorias. De acordo com o parâmetro *min_count*, palavras que possuem uma contagem inferior a este parâmetro são ignoradas pelo algoritmo [12]. A partir deste momento, o *Paragraph Vector* constrói o vocabulário de palavras com suas associações.

3.4 Previsão

Com o modelo criado, é possível verificar quais categorias são mais semelhantes entre si, obter o vocabulário criado, inferir os valores vetoriais de uma sentença e também obter os vetores que representam cada categoria. Estes dois últimos são úteis para o objetivo proposto, pois é possível inferir os valores vetoriais de um nome de produto e comparar com os valores vetoriais de cada categoria.

Essa comparação é realizada através de uma métrica de similaridade, a Similaridade de Cosseno, que possui como entrada dois vetores numéricos e retorna um número entre $[-1, 1]$, onde -1 indica que não é similar e 1 indica que é similar, como já foi explicado. O resultado com mais similaridade é a categoria prevista.

4 METODOLOGIA EXPERIMENTAL

4.1 *Dataset*

O *dataset* utilizado neste trabalho foi construído através de informações obtidas da própria *API* do Mercado Livre [5]. Os *endpoints* utilizados foram:

- <https://api.mercadolibre.com/sites/MLB/categories>: aqui se obtém os identificadores das categorias principais, que são as categorias raízes, e elas são úteis para buscar todas as outras categorias;
- <https://api.mercadolibre.com/categories/$CATEGORIA_ID>: utilizando o identificador *$CATEGORIA_ID*, nesta *URL* é possível obter as informações da categoria, como por exemplo as subcategorias que esta possui e o identificador de cada uma delas;
- <https://api.mercadolibre.com/sites/MLB/search?category=$CATEGORIA_ID&offset=$OFFSET>: utilizando os parâmetros *$CATEGORIA_ID* como identificado de categoria e *$OFFSET* como indicador de posição do vetor de produtos, é possível obter uma lista de produtos que uma categoria possui.

O processo de obtenção das informações acontece a partir da busca das categorias principais, com o objetivo de utilizar estas categorias como parâmetro de busca de informações. Buscando informações detalhadas de cada uma destas categorias principais, obtemos as subcategorias, onde podemos repetir o processo até chegar nas últimas subcategorias da árvore, ou seja, nos nós-folha. Por último, obtemos os produtos dessas últimas subcategorias.

Vinculando os produtos desta forma, onde cada nó da árvore possui um identificador de categoria e os nós-folha possuem informações sobre os produtos, podemos obter todas as categorias que um produto está vinculado, desde os nós-folha até o nó-raiz.

Nesta etapa do trabalho foram obtidas todas as categorias do Mercado Livre disponíveis pela *API*. Nem todas as categorias possuíam uma quantidade exata de produtos, tornando assim o conjunto de dados desbalanceado. Os registros de produtos somaram um total de 6730208 produtos e 8818 categorias. Todas estas informações foram obtidas no dia 29 de fevereiro de 2024.

Como os retornos dos *endpoints* da *API* do Mercado Livre eram em formato *JSON (JavaScript Object Notation)*, o armazenamento das informações também foi em formato

JSON, selecionando apenas as informações relevantes para uma posterior leitura. O arquivo armazenado ocupa um espaço de aproximadamente 36 MB.

Os dados coletados neste processo são, para cada categoria: o identificador de categoria, o nome da categoria e as subcategorias. E, para cada produto: o identificador de produto e o nome do produto.

Ao ler este arquivo, os dados são processados e transformados em um *dataset* organizado por colunas, descritas a seguir:

Tabela 2 – Tabela da organização das colunas

id	Identificador do produto
title	Nome do produto
categoria-especifica	Identificador da última subcategoria associada ao produto
categoria-especifica-nome	Nome referente ao identificador da coluna *categoria-especifica*
categoria-1	Identificador da categoria de nível 1 na árvore, associada ao produto
categoria-1-nome	Nome referente ao identificador da coluna *categoria-1*
categoria-2	Identificador da categoria de nível 2 na árvore, associada ao produto
categoria-2-nome	Nome referente ao identificador da coluna *categoria-2*
categoria-3	Identificador da categoria de nível 3 na árvore, associada ao produto
categoria-3-nome	Nome referente ao identificador da coluna *categoria-3*
categoria-4	Identificador da categoria de nível 4 na árvore, associada ao produto
categoria-4-nome	Nome referente ao identificador da coluna *categoria-4*
categoria-5	Identificador da categoria de nível 5 na árvore, associada ao produto
categoria-5-nome	Nome referente ao identificador da coluna *categoria-5*
categoria-6	Identificador da categoria de nível 6 na árvore, associada ao produto
categoria-6-nome	Nome referente ao identificador da coluna *categoria-6*
categoria-7	Identificador da categoria de nível 7 na árvore, associada ao produto
categoria-7-nome	Nome referente ao identificador da coluna *categoria-7*

A coluna *id*, que serve para identificar um produto e indicar a singularidade deste, e as colunas que representam os nomes das categorias, não foram utilizadas para treino.

Com o *dataset* construído, a padronização dos nomes dos produtos foi realizada, como também a normalização e remoção de ruídos. Após a remoção destes, o *dataset* foi reduzido para o tamanho de 4140328 produtos e 4008 categorias.

A partir disso, foram escolhidas, aleatoriamente, 100 categorias, com um total de 103309 produtos. Considerando que a proposta do trabalho era prever categorias utilizando somente os dados provenientes do Mercado Livre, decidiu-se que o novo *dataset* seria dividido em dois: um *dataset* de treinamento com 77,44% dos dados, e um *dataset*

de teste com 22,56%.

4.2 Configuração Experimental

Neste trabalho foram treinados diversos modelos do algoritmo *Paragraph Vector*, um para cada categoria existente que não fosse a última subcategoria, ou seja, uma categoria sem subcategorias. Cada modelo tinha o objetivo de prever qual subcategoria possuía mais similaridade com o nome do produto dado como entrada. Dessa forma, era possível identificar a categoria de um produto em diversos graus da árvore de categorias, até o nó-folha. A Figura 10 exemplifica como a previsão de categorias é realizada.

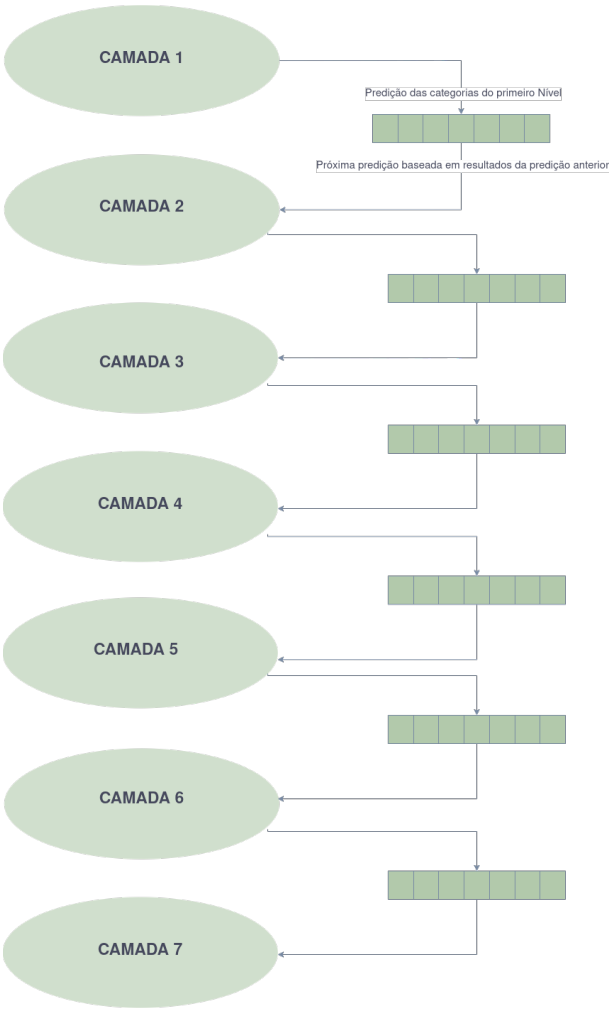


Figura 10 – Camadas de previsão de categorias do Mercado Livre - Fonte: Elaborado pelo autor

Para o treinamento de cada modelo *Paragraph Vector*, foram utilizados os seguintes parâmetros:

vector_size: é o tamanho do vetor que representa as palavras e os nomes dos produtos. Foi definido como *20*.

epochs: representa a quantidade de interações que o algoritmo irá realizar em todas as informações, para extrair informações relevantes. Foi definido como *10*.

min_count: define a contagem mínima de cada palavra para que ela seja considerada no treinamento. Isso significa que palavras que raramente aparecem nos nomes dos produtos são ignoradas. O valor deste parâmetro foi definido como 5.

dm: quando possui o valor 1, é utilizado o algoritmo *PV-DM*, caso contrário se utiliza o algoritmo *PV-DBOW*. O valor definido foi 0.

dbow_words: é aceito somente quando $dm = 0$. Quando o valor deste parâmetro é 1, considera-se o algoritmo *PV-DBOW* em conjunto com o *PV-DM*. Neste trabalho, foi definido como 1, conforme Quoc Le [10] definiu como mais eficiente.

4.3 Recursos Computacionais

O pré-processamento dos dados e o treinamento do modelo *Paragraph Vector* foi realizado em um *hardware* 12th Gen Intel® Core™ i5-12450H × 12 com 16 GB de RAM, com placa de vídeo integrada Intel® UHD Graphics. O sistema operacional utilizado foi Arch Linux, *rolling-release*, com *kernel* Linux na versão Linux 6.8.2-arch2-1 para arquitetura x86-64.

A implementação do algoritmo *Paragraph Vector* foi obtida pela biblioteca *Gensim*, com o nome *Doc2Vec*. Para obter a tokenização das palavras, a biblioteca *nltk* foi utilizada. E para o cálculo da similaridade, foi obtida uma função da biblioteca *scipy* que retornava a similaridade entre $[0, 2]$, onde a similaridade é alta quando o valor se aproxima de 0 e baixa quando se aproxima de 2. Isso porque o cálculo é realizado dessa forma:

$$z = 1 - \text{sim}_{\cos}(x, y) \tag{4.1}$$

onde
$\text{sim}_{\cos}(x, y)$ é o cálculo original da similaridade de cosseno,
e $z = 1 - [-1, 1] = [0, 2]$.

A linguagem de programação utilizada para os experimentos foi *Python 3.11.8*.

4.4 Métricas de Desempenho

Para quantificar os acertos nas previsões das categorias, foi necessário empregar algumas métricas, inspiradas no trabalho já realizado por Leonardo Santos Paulucio [7]. Esta matriz possui os registros de cada previsão, e avalia os verdadeiros positivos e negativos, como também os falsos positivos e negativos.

No entanto, ao escolher as métricas que mais se adequam ao objetivo proposto, temos que considerar que o *dataset* obtido é desbalanceado. Portanto, foram utilizadas as seguintes métricas de desempenho.

4.4.1 Acurácia Balanceada

Para obter uma taxa de acerto em que se consideram todos os registros de previsão, utiliza-se a acurácia balanceada, pois quando se utilizam somente os registros de previsão da própria categoria, tendo em vista que o *dataset* é desbalanceado, a taxa de acerto se torna enviesada.

A acurácia balanceada emprega duas métricas:

Sensibilidade: Também chamada de *Recall*, é a métrica que mede a taxa de predições corretas da categoria, avaliando quantas predições da categoria selecionada foram corretas (Verdadeiros Positivos - VP) e quantas foram incorretas (Falso Negativo - FN). A Equação 4.2 descreve como o cálculo é feito.

$$\text{Sensibilidade}_c = \frac{\text{VP}_c}{\text{VP}_c + \text{FN}_c} \tag{4.2}$$

onde c é a categoria selecionada.

Especificidade: Mede a taxa de rejeições corretas da categoria, considerando quantas predições de outras categorias não previram a categoria selecionada (Verdadeiro Negativo - VN) e quantas previram (Falso Positivo - FP). A Equação 4.3 mostra o cálculo desta métrica.

$$\text{Especificidade}_c = \frac{\text{VN}_c}{\text{VN}_c + \text{FP}_c} \tag{4.3}$$

onde c é a categoria selecionada.

A partir disso, a acurácia balanceada de uma categoria pode ser escrita como:

$$\text{ACB}_c = \frac{\text{Sensibilidade}_c + \text{Especificidade}_c}{2} \tag{4.4}$$

A acurácia balanceada total foi calculada como:

$$\text{ACB}_T = \frac{\sum_{c=1}^{C} \text{ACB}_c}{C} \tag{4.5}$$

onde
C é a quantidade total de categorias,
ACB_c é a acurácia balanceada da categoria c.

4.4.2 Acurácia Top-K

Esta métrica considera a predição como correta quando o vetor de saída de tamanho k, com as melhores pontuações, possui a categoria esperada. Neste trabalho, as melhores pontuações foram consideradas as mais próximas de 0, pois indicavam maior similaridade entre o nome de um produto e as categorias analisadas.

Para o objetivo proposto, esta métrica é útil pois possibilita a aplicação em um ambiente onde as categorias são sugeridas ao invés de determinadas. Dessa forma, 4 valores de k foram escolhidos: $\{1, 3, 5, 10\}$.

5 RESULTADOS

Os resultados obtidos na Figura 11 demonstram que o algoritmo obteve melhores resultados para a Camada 1, com acurácia de 92,29%. Isso se dá pois a Camada 1 possui menos categorias para classificar e mais produtos para o dataset de treino. No entanto, entre as camadas mais específicas, obteve-se uma melhora da predição na Camada 5, com acurácia de 75,34%, e uma piora na Camada 6, com valor de acurácia de 50,00%.

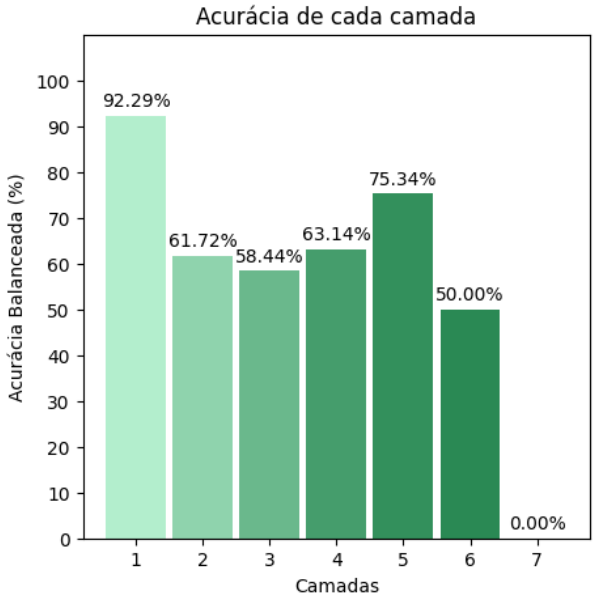


Figura 11 – Representação das categorias do Mercado Livre - Fonte: Elaborado pelo autor

Para as categorias, obteve-se resultados individuais de acurácia, destacando-se a categoria *Dobradiças de Porta-luva*, com acurácia de 99,84%, sendo esta a maior, e a categoria *Decoração de Exterior*, com acurácia de 49,09%, esta sendo a menor.

Em relação à métrica Top-K, os resultados observados indicam uma tendência de melhora da taxa de acerto do primeiro resultado, da Camada 3 até a Camada 6. Obteve-se também melhores resultados entre os 10 primeiros resultados de previsão de categorias de cada produto, como se observa na Figura 12. Vemos também que na Camada 1 a taxa de acerto do primeiro resultado não se destacou. Isso indica que os resultados de acurácia podem estar enviesados. Contudo, dada a escolha de um *dataset* desbalanceado com informações de produtos classificados pelos próprios vendedores da plataforma Mercado Livre, e a escolha aleatória de categorias e produtos para treino e teste dos modelos, considera-se que as camadas mais altas tenham resultados melhores, pois as categorias destas camadas possuem mais produtos para treino.

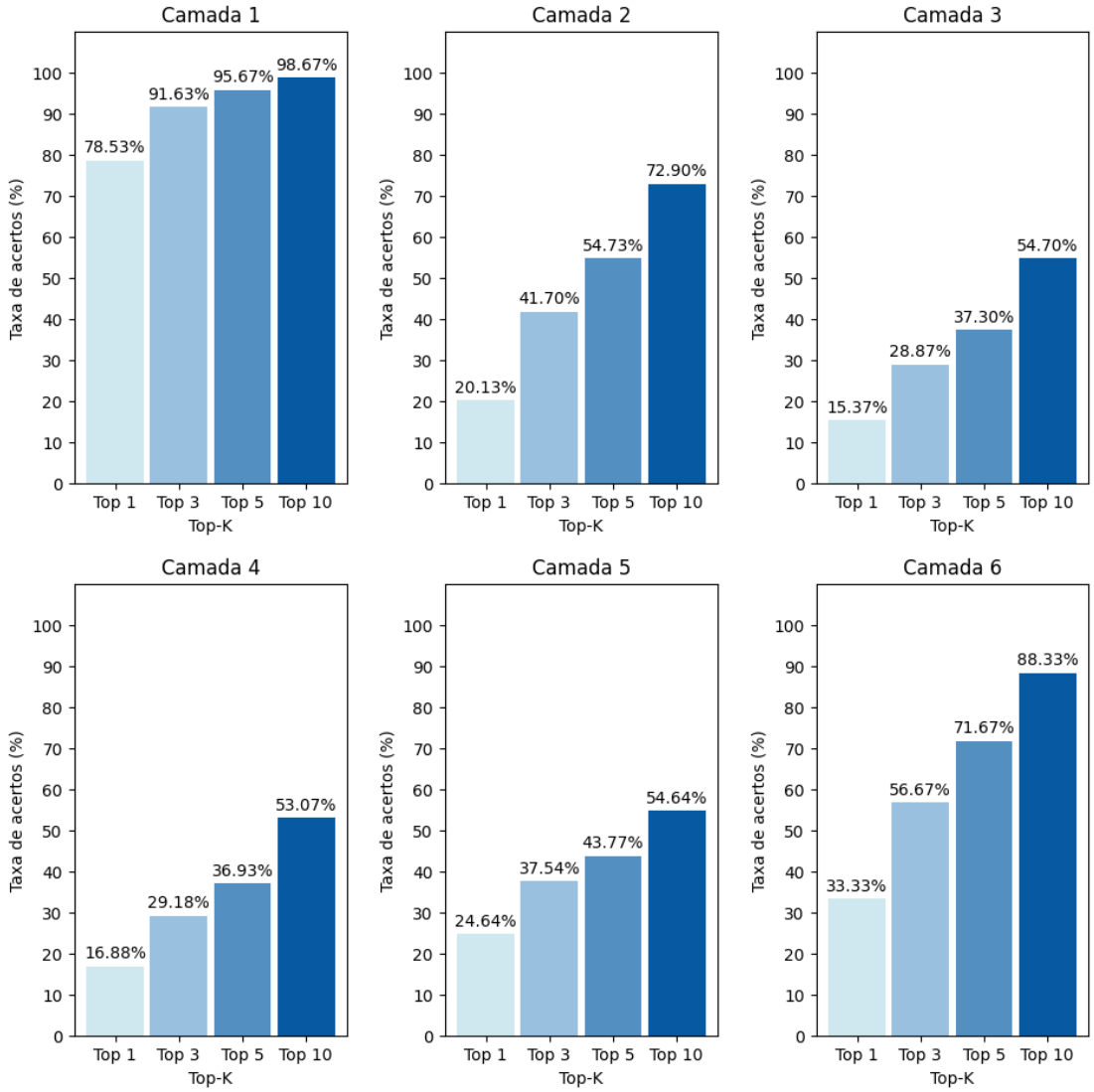


Figura 12 – Comparação entre as taxas de acerto das camadas - Fonte: Elaborado pelo autor

6 CONCLUSÃO

Sabe-se que o uso dos *marketplaces* tem influenciado diretamente em como os sites de *e-commerce* se comportam ao colocar produtos para a venda, e ao estruturar o próprio site para se adequarem às tendências do mercado. As empresas buscam inserir as tecnologias mais atuais de acordo com as suas necessidades e recursos financeiros. A automatização de algumas tarefas, como a categorização de produtos, trazem agilidade para os processos processos diários. Considerando esta necessidade, este trabalho teve como objetivo a criação de um sistema, utilizando o algoritmo *Paragraph Vector* para a categorização de produtos, com o diferencial de prever diversos níveis de categorias e subcategorias.

Nos testes experimentais, a escolha de categorias aleatórias limitadas a 100 foi necessária de acordo com os recursos computacionais disponíveis. Para uma quantidade maior de dados de treinamento, as limitações de recursos influenciam no tempo de treinamento e de teste, além de uma predição mais lenta conforme o tamanho do modelo gerado. A remoção de categorias com menos de 1000 produtos mostrou-se útil para expor resultados de modelos treinados com mais dados, proporcionando assim um ambiente controlado para observar o comportamento do algoritmo. Esta remoção pode ser evitada adicionando, sob supervisão, produtos externos verificados e classificados manualmente.

Os resultados destes testes mostram que o primeiro resultado, na maior parte dos casos, não é correto. No entanto, na camada mais específica, os 10 primeiros resultados são satisfatórios, apresentando a categoria correta. Portanto, o algoritmo *Paragraph Vector* atingiu o objetivo proposto do estudo.

Para trabalhos futuros, será relevante a realização de testes com mais produtos nas camadas subsequentes, pois se observarmos a primeira camada, os 10 primeiros resultados alcançaram um acerto de 98,67%, e esta camada se destacou em todas as métricas. Esta camada também possui mais produtos para treinamento. Além disso, será de grande importância a previsão de categorias de modo mais ágil, utilizando mais recursos computacionais e/ou métodos diferentes, como também a investigação de outros algoritmos de PLN que mais se adequem ao problema proposto.

REFERÊNCIAS

[1] FORUM, W. E. *COVID-19 increased the use of AI. Here's why it's here to stay*. 2021. <https://www.weforum.org/agenda/2021/02/covid-19-increased-use-of-ai-here-s-why-its-here-to-stay/>. [Online; acessado em 07-Setembro-2023].

[2] E-commerce Brasil. *Evolução do e-commerce: cinco fatos para entender melhor o mercado*. 2022. <https://www.ecommercebrasil.com.br/artigos/evolucao-do-e-commerce>. [Online; acessado em 07-Setembro-2023].

[3] AL-MASHARI, M. Enterprise resource planning (erp) systems: a research agenda. *Industrial Management & Data Systems*, MCB UP Ltd, v. 102, n. 3, p. 165–170, Jan 2002. ISSN 0263-5577. Disponível em: <https://doi.org/10.1108/02635570210421354>.

[4] ORACLE. *Your Complete Guide to Modern ERP*. 2017. Https://www.oracle.com/webfolder/s/assets/ebook/modern-erp/index.html. [Online; acessado em 12-Setembro-2023].

[5] LIBRE, M. *Documentação do Mercado Livre*. 2020. https://developers.mercadolivre.com.br/pt_br/guia-para-produtos. [Online; acessado em 15-Setembro-2023].

[6] GOLDBERG, Y. *Neural Network Methods for Natural Language Processing*. 1st. ed. [S.l.]: Morgan and Claypool, 2017.

[7] PAULUCIO, L. S. Categorização automática de produtos utilizando apenas o título e aprendizado profundo. In: . [S.l.: s.n.], 2022.

[8] EISENSTEIN, J. *Natural Language Processing*. [S.l.]: The MIT Press, 2019.

[9] MIKOLOV, T. et al. *Efficient Estimation of Word Representations in Vector Space*. 2013.

[10] LE, Q.; MIKOLOV, T. *Distributed Representations of Sentences and Documents*. 2014.

[11] JATNIKAA, D.; MOCH, B. A.; SURYANIA, A. A. *Word2Vec Model Analysis for Semantic Similarities in English Words*. 2019.

[12] GENSIM. *models.doc2vec - Doc2Vec Paragraph embeddings*. <https://radimrehurek.com/gensim/models/doc2vec.html>. [Online; acessado em 27-Janeiro-2024].