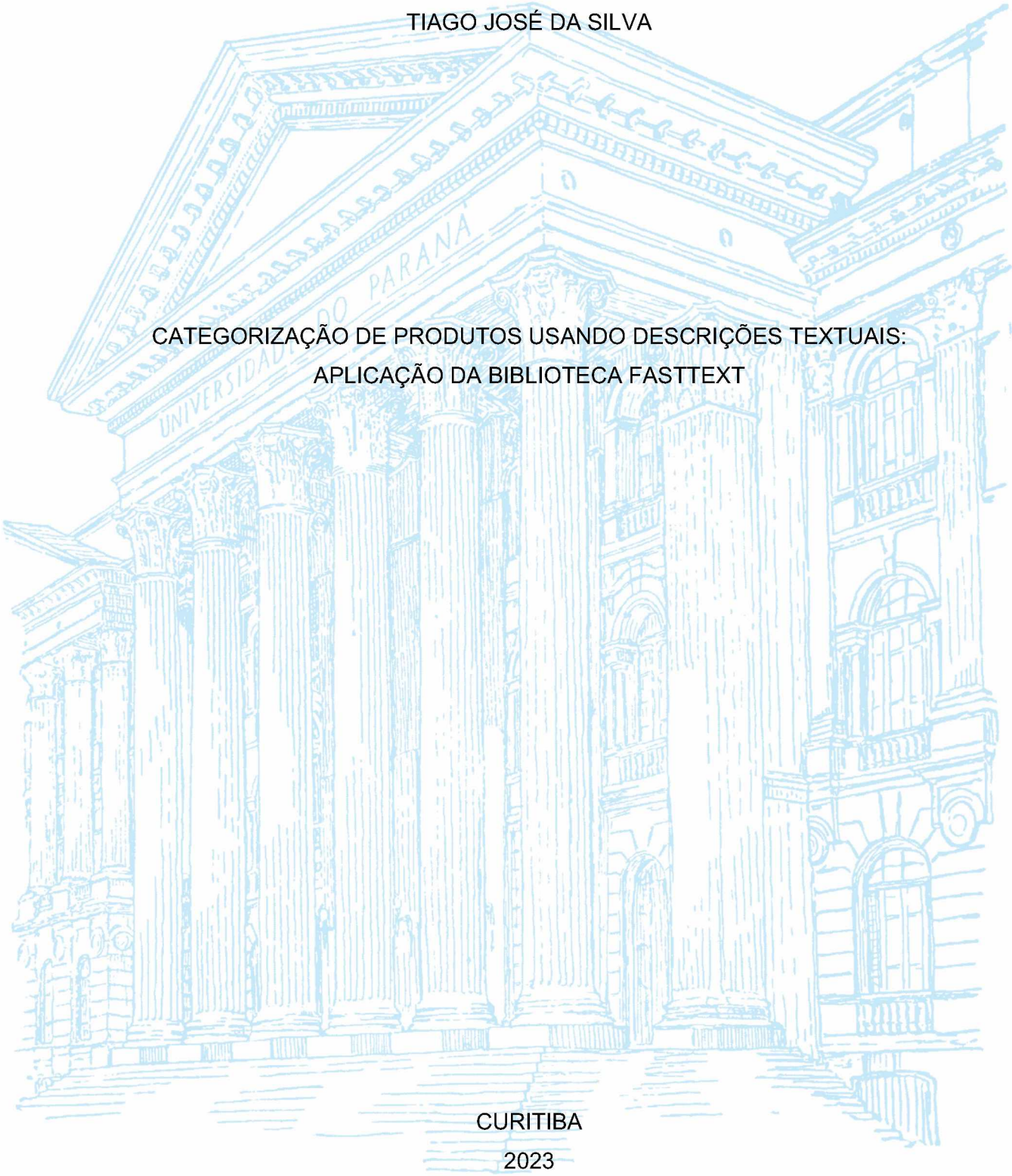
UNIVERSIDADE FEDERAL DO PARANÁ

TIAGO JOSÉ DA SILVA



CATEGORIZAÇÃO DE PRODUTOS USANDO DESCRIÇÕES TEXTUAIS: APLICAÇÃO DA BIBLIOTECA FASTTEXT

CURITIBA

2023

TIAGO JOSÉ DA SILVA

CATEGORIZAÇÃO DE PRODUTOS USANDO DESCRIÇÕES TEXTUAIS: APLICAÇÃO DA BIBLIOTECA FASTTEXT

Trabalho de Conclusão de Curso apresentado ao curso de Pós-Graduação em Inteligência Artificial Aplicada, Setor de Educação Profissional e Tecnológica, Universidade Federal do Paraná, como requisito parcial à obtenção do título de Especialista em Inteligência Artificial Aplicada.

Orientador: Prof. Dr. Razer Anthom Nizer Rojas Montaño.

CURITIBA

2023

MINISTÉRIO DA EDUCAÇÃO
SETOR DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
UNIVERSIDADE FEDERAL DO PARANÁ
PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO
CURSO DE PÓS-GRADUAÇÃO INTELIGÊNCIA ARTIFICIAL APLICADA - 40001016348E1

TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação INTELIGÊNCIA ARTIFICIAL APLICADA da Universidade Federal do Paraná foram convocados para realizar a arguição da Monografia de Especialização de **TIAGO JOSÉ DA SILVA** intitulada: **Categorização De Produtos Usando Descrições Textuais: Aplicação Da Biblioteca Fasttext**, que após terem inquirido o aluno e realizada a avaliação do trabalho, são de parecer pela sua APROVAÇÃO no rito de defesa.

A outorga do título de especialista está sujeita à homologação pelo colegiado, ao atendimento de todas as indicações e correções solicitadas pela banca e ao pleno atendimento das demandas regimentais do Programa de Pós-Graduação.

Curitiba, 22 de Junho de 2023.

RAZER ANTHOM NIZER ROJAS MONTAÑO
Presidente da Banca Examinadora

JAIME WOJCIECHOWSKI
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

Categorização de produtos usando descrições textuais: aplicação da biblioteca FastText

Tiago José da Silva
Setor de Educação Profissional e Tecnológica
Universidade Ferdaral do Paraná
Curitiba, Brasil
tiago.silva1@ufpr.br

Razer Anthom Nizer Rojas Montaño
Setor de Educação Profissional e Tecnológica
Universidade Ferdaral do Paraná
Curitiba, Brasil
razer@ufpr.br

***Resumo*—Este artigo apresenta uma análise do uso do *FastText*, uma técnica de *Machine Learning*, para a categorização de produtos de um *e-commerce* brasileiro do segmento de moda e esporte, mais especificamente a empresa *Netshoes*. O objetivo deste trabalho é avaliar a eficácia do *FastText* quando aplicado a um conjunto de dados privados fornecido pela empresa. Quatro combinações de recursos textuais foram testadas durante os treinamentos com a finalidade de achar o melhor conjunto de atributos para criação do modelo de classificação. Além disso, testou-se a eficácia da segmentação de textos maiores - como descrições de produtos - onde os treinamentos foram realizados utilizando trechos do texto ao invés do texto completo. Os resultados obtidos indicam que o *FastText*, quando treinado com recursos relativos ao nome, descrição, marca, peso e dimensões do produto, alcançou o melhor desempenho. A segmentação da descrição do produto não mostrou melhorar os resultados, sugerindo que esta técnica pode não ser a mais adequada para os dados analisados neste estudo.**

***Palavras-chave*—Aprendizado de Máquina, categorização de produtos, e-commerce, FastText, segmentação de texto, análise de dados, micro f1, macro f1.**

***Abstract*—This article presents an analysis of the use of FastText, a Machine Learning technique, for the categorization of products of a Brazilian e-commerce in the fashion and sports segment, more specifically the company Netshoes. The aim of this work is to assess the efficacy of FastText when applied to a private dataset provided by the company. Four combinations of textual features were tested during the training process in order to find the best set of attributes for the creation of the classification model. In addition, the effectiveness of segmenting larger texts - such as product descriptions - was tested, where the training was carried out using excerpts from the text instead of the full text. The obtained results indicate that FastText, when trained with features related to the product's name, description, brand, weight, and dimensions, achieved the best performance. The segmentation of the product description did not show improvement in the results, suggesting that this technique may not be the most suitable for the data analyzed in this study.**

***Index Terms*—Machine Learning, product categorization, e-commerce, FastText, text segmentation, data analysis, micro f1, macro f1.**

I. Desenvolvimento

Lojas de comércio eletrônico costumam oferecer um abrangente leque de produtos e serviços em suas plataformas. Esse portfólio tem crescido constantemente durante os últimos anos impulsionado pela populariação do uso de equipamentos eletrônicos, como *smartphones* [1], por uma maior aceitação de clientes no uso da tecnologia e pela pandemia de COVID-19 que levou governos a impedir a locomoção física pelas cidades a fim de diminuir as taxas de contágio [2]. Segundo estudo da FGV[1] [3], que analisou os resultados de 745 empresas de diferentes segmentos, o *e-commerce* brasileiro foi responsável por 21,2% das vendas de varejo durante Junho de 2021, antes da pandemia de COVID 19, no entanto, o percentual era de 9,2%.

Com portfólios de produtos cada vez maiores, a dificuldade em categorizar corretamente produtos cadastrados ganhou relevância. Empresas de *e-commerce* como *Magazine Luiza*, *Netshoes*, *Mercado Livre* e a *Amazon*, costumam categorizar os produtos vendidos em suas plataformas utilizando taxonomias predefinidas e adequadas às suas estratégias de venda. Os dados utilizados neste artigo utilizam uma taxonomia de dois níveis: **departamento** e **tipo de produto**. A Figura 1 demonstra a estrutura taxonômica utilizada.

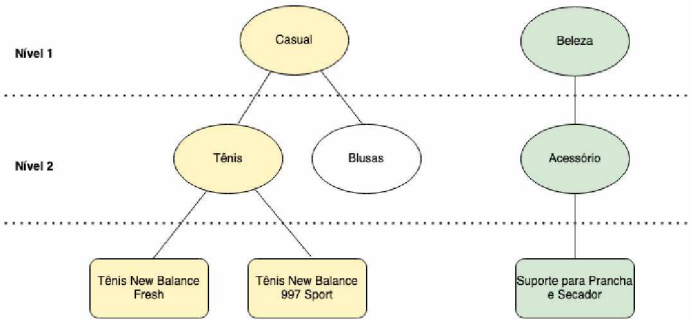


Figure 1. Exemplo de árvore de taxonomia com 2 níveis

Empresas de comércio eletrônico utilizam a categorização de produtos para dispô-los de forma organizada e agrupada em suas plataformas virtuais, o que gera melhorias de usabilidade para usuários e cria a possibilidade de análises de dados, até então não possíveis, por parte de tomadores de decisão.

No entanto, a atividade de categorização de produtos ainda é uma etapa executada por profissionais humanos em muitas empresas, o que gera demora na publicação de produtos em suas plataformas de venda digital [4], e consequente perda de vendas.

[1]Fundação Getúlio Vargas

Abordagens baseadas em tecnologia, como algoritmos de aprendizagem de máquina e processamento de linguagem natural, estão se tornando cada vez mais relevantes para automatizar o processo de categorização de produtos. A proposta deste artigo é testar a técnica de aprendizagem de máquina *FastText*[2] e avaliar o resultado de suas classificações após seus respectivos treinamentos utilizando uma base de dados com informações de produtos fornecida pela *Netshoes*.

FastText, como estabelecido por Joulin et al. [5], é uma técnica de classificação textual baseada em dois princípios - o modelo *Continuous Bag-of-Words (CBOW)* [6] e a representação de palavras por meio de n-gramas de caracteres, conhecida como *Subword Information* [7]. O conceito *CBOW*, tem como alicerce a previsão da palavra seguinte em um texto baseando-se na palavra atual. Para alcançar isso, uma rede neural é treinada, cuja camada de saída proporciona a distribuição de probabilidade da próxima palavra.

Esta rede neural, contendo três camadas, é treinada com a ajuda de um vetor *one-hot*, que atua como entrada. O objetivo final é otimizar o *embedding* das palavras, assegurando que palavras semanticamente similares estejam próximas umas das outras no espaço n-dimensional gerado.

Após o treinamento dos *embeddings*, uma segunda rede neural é configurada para inferir uma palavra com base em seu contexto. O *input* desta segunda rede é alimentado com os vetores das palavras de contexto e a camada de *output* representa a palavra inferida.

FastText tem sido uma ferramenta eficaz em diversas aplicações, incluindo a classificação de produtos, conforme relatado por Yu et al. [8].

A. Descrição dos dados

A base de dados utilizada neste artigo contém 13.200.902 registros, onde cada registro representa uma variante de um produto. No contexto deste artigo, o termo *produto* refere-se a um item específico que está à venda, identificado por atributos como marca e modelo, por exemplo, o *Tênis Asics modelo Gel Impression 11*. O termo *variante* refere-se às variações de um produto, que podem incluir cor, sabor ou tamanho. Por exemplo, o mesmo modelo de tênis *Asics* pode ter variantes em cores como vermelho, azul e preto, e tamanhos como 38, 39, 40, 41 e 42.

Os atributos disponíveis para cada registro são descritos na tabela I.

A categorição do produto é definida pelo conjunto de campos formado por *department* e *productType*.

B. Métodos

Neste artigo, foram exploradas duas estratégias de arquitetura, utilizando a biblioteca *FastText*, que serão chamadas de *Arquitetura Singular* e *Arquitetura Segmentada*.

- **Arquitetura Singular:** A primeira estratégia analisada centrou-se na classificação de um rótulo formado pela concatenação dos atributos *department* e *productType*. A motivação por trás desta abordagem era averiguar a capacidade de um único modelo classificar os produtos em dois níveis distintos.
- **Arquitetura Segmentada:** Foi construído um modelo primário para lidar com a classificação do primeiro nível (departamento), e subsequentemente, modelos secundários — um para cada departamento — destinados a classificar o segundo nível (tipo de produto).

Para cada uma dessas estratégias, foram selecionados quatro agrupamentos de texto para treinamento, buscando discernir qual conjunto traria maior contribuição para a tarefa de classificação. Os conjuntos englobam:

- Nome do produto e descrição completa;
- Nome e a primeira terça parte da descrição;
- Nome, descrição completa, marca, peso e volume;
- Nome, primeira terça parte da descrição, marca, peso e volume.

Para cada um desses agrupamentos de texto, foram conduzidos testes com diversas configurações de hiperparâmetros no *FastText*. O objetivo central era localizar o conjunto de hiperparâmetros que culminasse no melhor desempenho para os modelos.

Após a realização de todos os testes, o critério que determinou a escolha do modelo de arquitetura, conjunto de texto e o conjunto de hiperparâmetros foi o desempenho na métrica *Macro-Average F1* (*macF1*). A configuração que alcançou o maior valor nesta métrica foi a selecionada para o treinamento do modelo final.

O processo metodológico aplicado a este estudo é descrito pelas etapas enumeradas:

1) Montagem da base de dados
2) Pré-processamento dos dados
3) Preparação para treinamento
4) Treinamento
5) Predições

1) Montagem da base de dados

Para reduzir a quantidade de dados processados e aumentar a variabilidade nos dados de treinamento, foram considerados apenas os *produtos* neste trabalho, excluindo suas variações. Por exemplo, um produto com 5 variações de cor e 10 variações de tamanho seria representado por 50 registros na base de dados, todos com informações idênticas, como nome, descrição, medidas físicas e outras características referentes ao produto. Esta duplicidade não só aumentaria o tempo de análise e processamento, como também comprometeria a qualidade do modelo gerado.

O atributo *sku*, presente na base de dados original, foi utilizado para a tarefa de extração dos produtos. Graças ao seu formato, *XXX-XXXX-XXX-XX*, é possível identificar produtos iguais e suas variantes. Os dois primeiros blocos do atributo *sku* são suficientes para identificar a qual produto um registro se refere. Produtos iguais, desconsiderando suas variações, possuem a mesma sequência de caracteres nos dois primeiros blocos do atributo *sku*. Deste modo, foi gerada uma nova base de dados composta por um único registro para cada produto

[2]https://fasttext.cc/

Table I
ATRIBUTOS DA BASE DE DADOS ORIGINAL

Atributo	Descrição	Tipo de dado
_id	Identificador único para cada registro.	String
sku	*Sku*, que é a sigla em inglês para *Stock Keeping Unit* (Unidade de Manutenção de Estoque em português), é o código que identifica a variante do produto na loja e em todos os demais fluxos sistêmicos. O *sku* segue o formato XXX-XXXX-XXX-XX, onde cada *X* é um caractere alfa numérico, cada hífen separa um bloco de caracteres e cada bloco de caracteres revela uma informação sobre o produto e a variante em questão. O primeiro bloco representa a marca do produto, o segundo o modelo, o terceiro a variante de cor ou sabor, e o último bloco representa a variante de tamanho.	*String*
name	Nome do produto, muitas vezes descrevendo algumas outras características do produto.	*String*
description	Descrição do produto muitas vezes escrita de forma mais extensa e detalhada, geralmente contendo suas características, funcionalidades e benefícios.	*String*
department	Categoria ao qual o produto pertence. Utilizado para agrupar produtos semelhantes em categorias maiores, como eletrônicos, vestuário, alimentos, etc.	*String*
productType	Subcategoria ao qual o produto pertence. Utilizado para agrupar produtos semelhantes em subcategorias pertencentes à categoria principal. *Celulares*, pode ser uma subcategoria de *Eletrônicos*, por exemplo.	*String*
brand	Marca do produto.	*String*
color	Se aplicável, possui a cor da variante em forma de texto. Às vezes representa uma combinação de cores concatenadas com o sinal de soma (+). Exemplo de valores possíveis: Amarelo, Azul+Amarelo. Este campo não é utilizado para produtos com o campo *flavor* preenchido.	*String*
flavor	Se aplicável, possui o sabor da variante em forma de texto. Este campo não é utilizado para variantes com o campo *color* preenchido.	*String*
size	Tamanho da variante. Pode estar representado em várias escalas a depender do tipo de produto. Calçados costumam ter tamanhos representados em números, roupas, por sua vez, costumam ter seus tamanhos representados em letras.	String
gender	Gênero do público ao qual a variante é destinada.	*String*
listPriceInCents	Preço sugerido para venda da variante sem a aplicação de descontos promocionais. O valor é representado em centavos. R$159,22, por exemplo, é representado por 15.922 centavos.	*Integer*
salePriceInCents	Preço de venda da variante, após a aplicação dos descontos vigentes. Assim como o atributo *listPriceInCents*, esse campo também é representado em centavos.	*Integer*
weightInGrams	Medida física que representa o peso do produto, sem sua embalagem, medido em gramas.	*Double*
depthInCm	Medida física que representa a profundidade da embalagem do produto em centímetros.	*Double*
widthInCm	Medida física que representa a largura da embalagem do produto em centímetros.	*Double*
heightInCm	Medida física que representa a altura da embalagem do produto em centímetros.	*Double*
createdDate	Data de cadastro da variante.	*Date*

com seus respectivos atributos. Nesta nova estrutura de dados, os atributos referentes à variante, como *color*, *flavor*, *size*, *gender* e *priceInCents*, foram desconsiderados.

Ademais, os atributos representantes das dimensões do produto, *depthInCm*, *widthInCm* e *heightInCm*, presentes na base de dados original, foram transformados em um único atributo, *volume*, na base de dados resultante. Esta transformação foi realizada para simplificar a representação dimensional dos produtos e potencializar a eficácia do modelo de aprendizado de máquina. O volume foi calculado multiplicando as medidas de altura, largura e profundidade de cada produto. Este processo resultou em uma base de dados com menos redundância, o que contribui para a eficiência do modelo [9].

O atributo *description* foi segmentado em 3 novos atributos - *desc_segmentation_1*, *desc_segmentation_2* e *desc_segmentation_3*, cada um contendo uma terça parte do seu conteúdo original. O objetivo foi testar a abordagem citada por Eli Cortez et al [4], onde modelos treinados utilizando apenas as primeiras palavras da descrição são mais efetivos que outros modelos gerados a partir da descrição integral.

Também foi adicionado o atributo *department_productType* ao conjunto de dados. Este atributo é uma concatenação dos atributos *department* e *productType*, separados por duas cerquilhas (##). Por exemplo, um produto do departamento *Casual* e tipo de produto *Sapatênis* terá como valor do atributo *department_productType* o texto *Casual##Sapatênis*. Este novo atributo facilitará os treinamentos e geração de métricas.

Por fim, foram removidos 31.877 registros por possuírem dados faltantes. A opção por removê-los se deu por conta deste volume representar apenas 1,2% da base total.

O conjunto de dados resultante possui 2.672.918 produtos e está descrita na tabela II.

2) Pré-processamento dos dados

Neste artigo, os seguintes passos de pré-processamento foram aplicados aos atributos *name*, *description*, *brand*, *desc_segmentation_1*, *desc_segmentation_2* e *desc_sementation_3*:

- ***Tokenização*:** A *tokenização*, também conhecida como segmentação de palavras, quebra a sequência de caracteres em um texto localizando o limite de cada palavra, ou seja, os pontos onde uma palavra termina e outra

Table II
ATRIBUTOS DA BASE DE DADOS RESULTANTE

Atributo	Descrição	Tipo de dado
name	Nome do produto, muitas vezes descrevendo algumas outras características.	String
description	Descrição do produto muitas vezes escrita de forma mais extensa e detalhada, geralmente contendo suas características, funcionalidades e benefícios.	*String*
department	Categoria ao qual o produto pertence. Utilizado para agrupar produtos semelhantes em categorias maiores, como Eletrônicos, Vestuário, Alimentos, etc.	*String*
productType	Subcategoria ao qual o produto pertence. Utilizado para agrupar produtos semelhantes em subcategorias pertencentes à categoria principal. *Celulares*, pode ser uma subcategoria de *Eletrônicos*, por exemplo.	*String*
brand	Marca do produto.	*String*
weightInGrams	Medida física que representa o peso do produto, sem sua embalagem, representado em gramas.	*Double*
volume	Medida física resultante da multiplicação dos atributos *weighInGrams*, *depthInCm* e *heightInCm*, descritos na tabela I.	*Double*
desc_segmentation_1	Primeira terça parte da descrição do produto.	*String*
desc_segmentation_2	Segunda terça parte da descrição do produto.	*String*
desc_segmentation_3	Terceira terça parte da descrição do produto.	*String*
department_productType	*String* formada pelo valor do campo *department* seguido por '##' e, por fim, concatenado com o valor do campo *productType*. Esta coluna busca facilitar os treinamentos e testes.	*String*

começa, e as extrai [10]. Para fins de linguística computacional, as palavras assim identificadas são frequentemente chamadas de *tokens* [11].

- **Conversão para minúsculo:** Todos os *tokens* foram convertidos para letras minúsculas. Isso garante que palavras iguais, mas com diferentes combinações de letras maiúsculas e minúsculas, sejam tratadas como a mesma palavra pelo modelo.
- **Remoção de *stopwords*:** *Stopwords* são palavras que não contém informações significativas e são frequentemente removidas do texto para melhorar a eficiência dos modelos de aprendizado de máquina [11]. Para esta tarefa foi utilizada a biblioteca *NLTK* (*Natural Language Toolkit*)[3]. Pelo fato de algumas marcas de produtos utilizarem palavras consideradas *stopwords*, como artigos e preposições, em seus nomes, essa etapa não foi aplicada ao atributo *brand*.
- **Remoção de números e pontuação:** Utilizando expressão regular, todos os números e pontuações presentes nos *tokens* foram removidos. Isso foi feito para que o modelo se concentre nas palavras e não seja afetado por estes caracteres. Pelo fato de algumas marcas de produtos utilizarem pontuações em seus nomes, essa etapa não foi aplicada ao atributo *brand*.
- **Remoção de acentuação:** A acentuação dos *tokens* foi removida utilizando a biblioteca *Unidecode*[4]. Isso garante que palavras escritas com e sem acentos sejam compreendidas como iguais pelo modelo.
- ***Stemming* das palavras:** Também em uso da biblioteca *NLTK*, os *tokens* de palavras inflexionadas ou derivadas foram reduzidos ao seus radicais. Foi utilizada a abordagem *RSLP* (Removedor de sufixos da Língua Portuguesa), proposta por Viviane Moreira Orengo e Christian Huyck [12]. Por exemplo, os tokens *correndo* e *correr* passam a ser representados pelo *token corr* após a aplicação desta técnica.

Após os processos descritos de *tokenização* e tratamento dos *tokens* gerados, os atributos resultantes destes processos foram combinados a fim de gerar novos atributos, cada qual representando uma combinação de atributos a ser testada pelos processos de treinamento. As combinações feitas estão representadas na tabela III e foram adicionadas ao conjunto de dados da tabela resultante.

3) Preparação para o treinamento

O conjunto de dados utilizado neste artigo apresenta um notável desbalanceamento entre as classes de nível 2, *productType*, como demonstrado na Figura 2. O desbalanceamento de dados é um problema comum em tarefas de classificação, podendo levar o modelo de aprendizado de máquina a desenvolver um viés em direção às classes majoritárias, prejudicando o desempenho nas classes minoritárias. Para mitigar este problema, foi adotada a estratégia de subamostragem [13] na etapa de extração dos dados para treinamento.

Primeiramente, foi calculada a frequência de ocorrências de cada categoria no atributo *department_productType* e em seguida, a mediana dessas frequências. A mediana foi utilizada como um limiar para definir a quantidade máxima de ocorrências permitidas em cada classe de *department_productType*. Com este procedimento, as classes majoritárias foram subamostradas, resultando em um conjunto de dados mais equilibrado para o treinamento, conforme ilustrado na Figura 3.

O conjunto de dados extraído utilizando a técnica de subamostragem resultou em 259.086 registros.

A divisão dos dados para treinamento e teste foi feita utilizando a técnica *holdout*, que envolve a reserva de uma porção dos dados para teste e a utilização do restante para treinamento [14]. Neste artigo, dividimos os dados em 80% para treinamento e 20% para teste, mantendo a proporção das categorias em ambas as divisões para garantir uma distribuição

[3]https://www.nltk.org/
[4]https://pypi.org/project/Unidecode/

Table III
NOVOS ATRIBUTOS GERADOS A PARTIR DE COMBINAÇÕES DE FEATURES TEXTUAIS

Novo atributo	Atributos combinados
tokens_name_description	*name + description*
tokens_name_seg1	*name + desc_segmentation_1*
tokens_name_description_brand_weight_volume	*name + description + brand + weight + volume*
tokens_name_seg1_brand_weight_volume	*name + desc_segmentation_1 + brand + weight + volume*

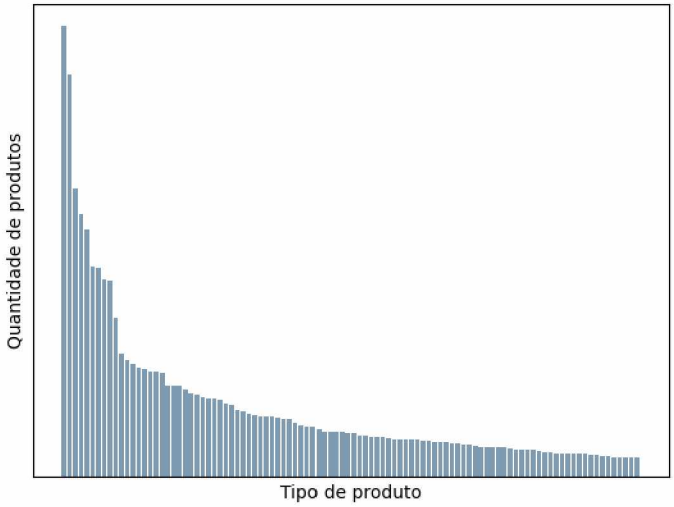


Figure 2. Distribuição das classes de Nível 2 - Tipo de Produto - antes do balanceamento

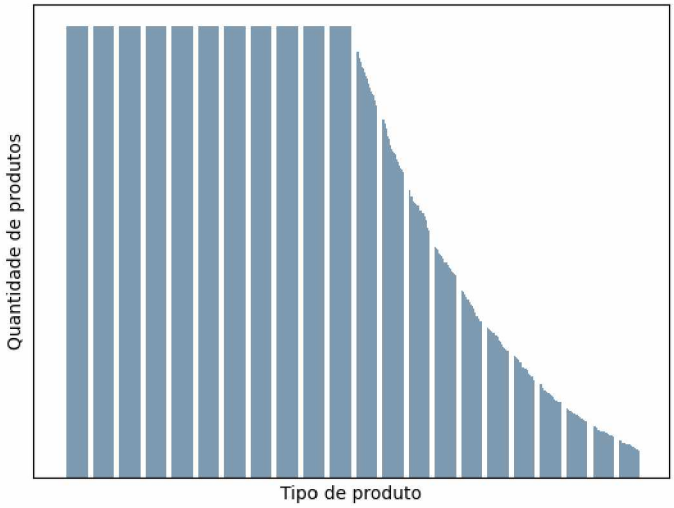


Figure 3. Distribuição das classes de Nível 2 - Tipo de Produto - após balanceamento

representativa.

Devido ao desbalanceamento inicial dos dados, optou-se por utilizar métricas de avaliação mais adequadas a essa situação: a *Macro-averaged F1* (*MacF1*) e a *Micro-averaged F1* (*MicF1*) [15]. A métrica *MacF1* é calculada a partir da média harmônica das pontuações F1 para cada classe, sem considerar a proporção entre elas, o que a torna sensível ao desempenho do modelo nas classes minoritárias. Por outro lado, a *MicF1* é calculada a partir da média harmônica das precisões e revocações totais, dando igual peso a todas as classes, independentemente de seu tamanho.

$$\text{MacF1} = \frac{1}{N}\sum_{i=1}^{N} 2 \cdot \frac{P_i \cdot R_i}{P_i + R_i} \tag{1}$$

Onde:

- **N** é o número total de classes;
- $\mathbf{P_i}$ é a precisão da classe;
- $\mathbf{R_i}$ é a revocação da classe;

$$\text{MicF1} = 2 \cdot \frac{P \cdot R}{P + R} \tag{2}$$

Onde:

- **P** é a precisão média de todas as classes;
- **R** é a revocação média de todas as classe;

4) Treinamento

Os hiperparâmetros testados durante a utilização do *FastText*, com seus respectivos significados [16], são:

- ***dim*:** Especifica a dimensionalidade dos vetores de palavras que serão aprendidos pelo modelo. Em termos simples, *dim* controla o número de variáveis que o modelo utilizará para representar cada palavra. Um valor maior para *dim* pode capturar mais informações sobre cada palavra, mas também pode tornar o modelo mais complexo e demorado para treinar.
- ***epoch*:** Determina o número de vezes que o algoritmo percorre todo o conjunto de dados durante o treinamento. Cada *epoch* é uma iteração completa pelos exemplos de treinamento. Um número maior de *epochs* permite que o modelo aprenda os padrões nos dados mais profundamente, mas também aumenta o risco de sobreajuste.
- ***wordNgram*:** Controla o número de *n-gramas* de palavras que o modelo considerará durante o treinamento. Um *n-grama* é um conjunto contínuo de *n* palavras dentro de um dado texto. Por exemplo, em um 2-grama (bigrama), o modelo considera pares de palavras consecutivas. Isso pode ajudar o modelo a aprender relações contextuais entre palavras que estão próximas umas das outras.
- ***loss*:** Define a função de perda a ser minimizada durante o treinamento. A função de perda quantifica o quão distante estão as previsões do modelo em relação aos valores reais. A biblioteca *FastText* suporta várias funções de

perda, como *ova* (*one-vs-all*), *ns* (*Negative Sampling*), *hs* (*Hierarchical Softmax*) e *softmax*.

- ***minCount***: Estipula o número mínimo de vezes que uma palavra deve aparecer nos dados de treinamento para que seja considerada pelo modelo. Isso pode ser útil para eliminar palavras raras que podem causar ruído nos dados.
- ***lr***: Conhecido como taxa de aprendizado, este hiperparâmetro controla a velocidade com que o modelo aprende. Uma taxa de aprendizado maior pode fazer com que o modelo aprenda mais rápido, mas também pode levar a um aprendizado instável. Por outro lado, uma taxa de aprendizado baixa pode fazer com que o modelo aprenda devagar, mas de forma mais estável.
- ***bucket***: Determina o número de *buckets* usados para armazenar *hashes* de *n-gramas*. Em *FastText*, um *bucket* é uma espécie de compartimento para armazenar *n-gramas* com *hashes* semelhantes, uma técnica usada para lidar com a enorme quantidade de *n-gramas* únicos e para manter a computação viável. Quando se define um alto valor para o *bucket*, mais *n-gramas* podem ser armazenados, o que pode melhorar a qualidade do modelo, mas ao mesmo tempo, requisitar mais memória.

Desta forma, para cada um dos atributos textuais descritos na tabela IV, realizou-se uma busca em *grid*, usando-se uma combinação destes hiperparâmetros. Os valores utilizados para cada hiperparâmetro são os descritos na tabela V.

Table IV
ATRIBUTOS UTILIZADOS NO TREINAMENTO DE MODELOS

Conjunto	Atributo
1	*tokens_name_description*
2	*tokens_name_seg1*
3	*tokens_name_description_brand_weight_volume*
4	*tokens_name_seg1_brand_weight_volume*

Table V
HIPERPARÂMETROS UTILIZADOS

Hiperparâmetro	Valores
dim	50 e 100
epoch	25 e 50
wordNgram	1 e 2
loss	*ova*, *ns*, *softmax*, *hs*
minCount	1 e 3
ws	3, 5 e 7
lr	0.1, 0.5 e 1
bucket	100000 e 200000

Este processo, que busca o melhor conjunto de hiperparâmetros para cada conjunto de texto representado pelos atributos testados, foi realizado para ambas as arquiteturas estudadas, a Arquitetura Singular e a Arquitetura Segmentada. Desta forma é possível definir a melhor estratégia de arquitetura, o melhor corpo de texto e os melhores hiperparâmetros em termos de *medida macro-average f1* (*MacF1*).

Este abrangente exercício de ajuste de hiperparâmetros ajudou a garantir que os modelos fossem treinados de maneira eficiente, gerando a melhor performance possível dada a complexidade e especificidades dos dados.

5) Predições

Considerando as predições feitas utilizando a base de testes para os dois modelos de arquitetura propostos, Singular e Segmentado, os melhores resultados considerando os 4 conjuntos de dados são expostos nas tabelas VI e VII.

Table VI
MÉTRICAS MICRO F1 E MACRO F1 DA CATEGORIZAÇÃO DE PRODUTOS UTILIZANDO A ARQUITETURA SINGULAR

Conjunto 1		Conjunto 2		Conjunto 3		Conjunto 4	
mic	**mac**	**mic**	**mac**	**mic**	**mac**	**mic**	**mac**
0.800	0.806	0.803	0.798	0.824	**0.825**	0.826	0.820

^{mic} Métrica MicF1
^{mac} Métrica MacF1

Table VII
MÉTRICAS MICRO F1 E MACRO F1 DA CATEGORIZAÇÃO DE PRODUTOS UTILIZANDO A ARQUITETURA SEGMENTADA

Conjunto 1		Conjunto 2		Conjunto 3		Conjunto 4	
mic	**mac**	**mic**	**mac**	**mic**	**mac**	**mic**	**mac**
0.716	0.713	0.707	0.704	0.727	**0.722**	0.723	0.718

^{mic} Métrica MicF1
^{mac} Métrica MacF1

O conjunto 3 foi o conjunto que resultou no melhor valor para a métrica *MacF1* em ambas as arquiteturas, como mostrado na figura 4, por essa razão ele foi utilizado para treinamento do modelo final. Os hiperparâmetros responsáveis pelas melhores métricas *MacF1* são os exibidos na tabela VIII.

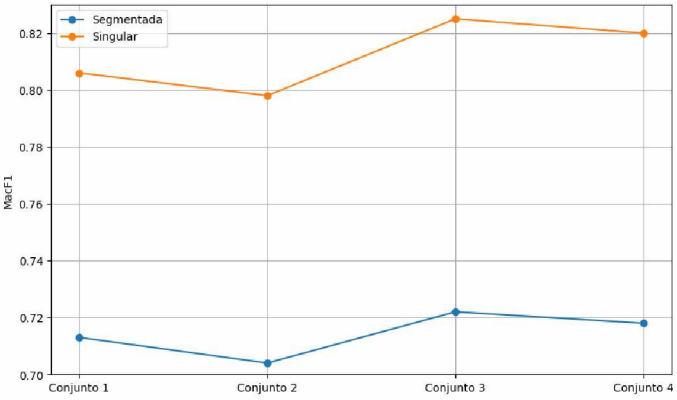


Figure 4. Comparação da métrica MacF1 entre os conjuntos de texto para Arquiteturas Singular e Segmentada

C. Tecnologias

Para esta pesquisa foi utilizado um computador de uso pessoal Apple MacBook Pro Processador M1 de 8 núcleos a 2.3 GHz, 16 Gb de Ram e 512 Gb de armazenamento SSD.

Table VIII
MELHORES VALORES DE HIPERPARÂMETROS PARA CADA ARQUITETURA

Hiperparâmetro	Arquitetura singular	Arquitetura Segmentada
loss	*ova*	*softmax*
lr	1	1
dim	100	50
epoch	25	25
wordNgram	2	2
bucket	200000	200000
minCount	3	1
ws	7	7

As ferramentas e bibliotecas utilizadas foram:

- *Python*[5] versão 3.9.12
- *Anaconda Individual Edition*[6] versão 2021.11
- *fastText*[7] versão 0.9.2
- *Scikit-learn*[8] versão 1.0.2
- *Pandas*[9] versão 1.4.2
- *NumPy*[10] versão 1.21.5
- *NLTK*[11] versão 3.7
- *Unidecode*[12] versão 1.3.6

II. RESULTADOS E DISCUSSÕES

A partir dos modelos gerados, considerando o melhor conjunto de hiperparâmetros e o melhor conjunto de dados de treinamento para cada arquitetura, ao classificar todos os produtos da base de dados, obteve-se os resultados descritos na tabela IX.

Em comparação entre os dois modelos arquiteturais testados, o modelo de arquitetura segmentada obteve o melhor resultado ao classificar o Nível 2 do produto, conforme visto nas figuras 5 e 6.

Table IX
COMPARAÇÃO DOS RESULTADOS DA CLASSIFICAÇÃO TOTAL DE PRODUTOS OBTIDOS COM MODELOS GERADOS PELAS ARQUITETURAS SINGULAR E SEGMENTADA

	Arquitetura Singular		Arquitetura Segmentada	
	MicF1	**MacF1**	**MicF1**	**MacF1**
Nível 1	0.56	0.66	0.87	0.80
Nível 2	0.45	0.57	**0.74**	**0.74**

A. Medida de qualidade dos modelos

A medida de qualidade dos modelos utilizada neste trabalho é a F1 nas suas variantes *Micro-Average F1* (*MicF1*) e *Macro-Average F1* (*MacF1*).

[5]https://www.python.org/
[6]https://www.anaconda.com/blog/anaconda-individual-edition-2021-11
[7]https://fasttext.cc/
[8]https://scikit-learn.org
[9]https://pandas.pydata.org/
[10]https://numpy.org/
[11]https://www.nltk.org/
[12]https://pypi.org/project/Unidecode/

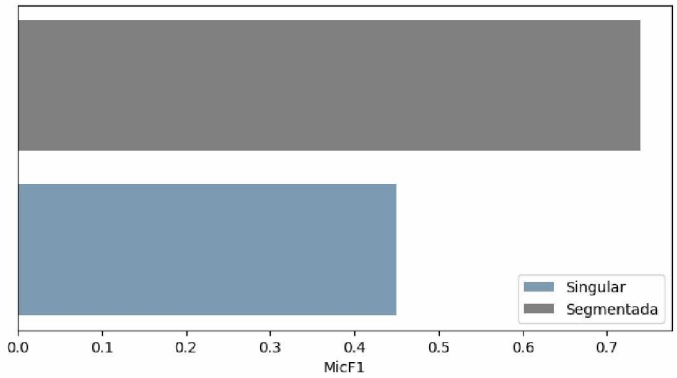


Figure 5. Comparação de métricas MicF1 após teste final para Arquiteturas Singular e Segmentada

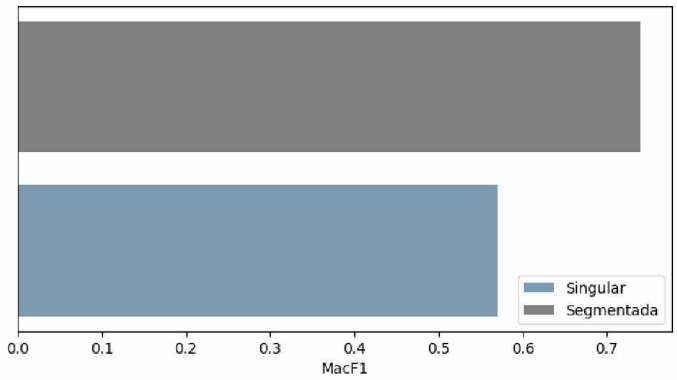


Figure 6. Comparação de métricas MacF1 após teste final para Arquiteturas Singular e Segmentada

A partir destes dados é possível afirmar que o modelo de Arquitetura Segmentada obteve melhor desempenho, com *MicF1* de 0.74 e *MacF1* de 0.74.

Também é possível concluir que não houve ganhos significativos nos modelos treinados com dados utilizando a segmentação de textos grandes. Modelos treinados com a descrição completa obtiveram melhores resultados.

B. Discussões

Embora o resultado obtido durante a etapa de treinamento indique que a Arquitetura Segmentada é a melhor abordagem, levando em consideração a métrica *MacF1*, o mesmo não se provou ao submeter todos os produtos ao teste do modelo.

Ao analisar a figura 7, nota-se que a métrica *MacF1* pouco variou para a Arquitetura Segmentada entre a fase de treinamento e o teste final. Já para a Arquitetura Singular, houve variação superior à 30%, o que pode indicar um possível *overfitting* durante a fase de treinamento. Outra possibilidade é que algum aspecto particular dos dados de teste não tenha sido adequadamente capturado pela Arquitetura Singular, afetando assim a sua capacidade de generalização.

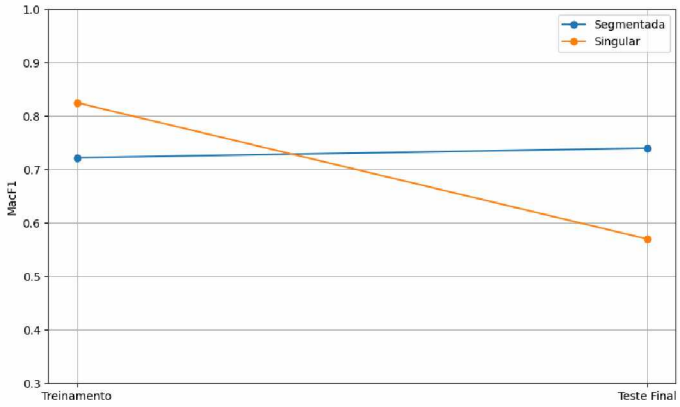


Figure 7. Comparação do desempenho alcançado pelas Arquiteturas Singular e Segmentada durante o treinamento e o teste final segundo métrica MacF1

III. Considerações Finais e Trabalhos Futuros

O presente artigo explorou a capacidade da tecnologia *FastText*, desenvolvida pelo *Facebook*, em uma tarefa específica de classificação de produtos. No entanto, é importante reconhecer que existem outras tecnologias disponíveis que podem ser utilizadas para essa mesma tarefa. Trabalhos futuros podem considerar a implementação de diferentes métodos de aprendizado de máquina e técnicas de processamento de texto para comparar e contrastar os resultados obtidos neste artigo.

Apesar das expectativas, a técnica de segmentação da descrição do produto não levou a um aumento nas métricas *MicF1* e *MacF1* do modelo em comparação ao uso do texto descritivo integral. Trabalhos futuros podem explorar outras estratégias de segmentação, inclusive a segmentação de outros atributos além da descrição, como por exemplo, o nome do produto.

Outra linha de testes possível é medir a eficiência do *FastText* ao treiná-lo com o texto original dos atributos, sem passar por etapas de pré-processamento como *tokenização*, conversão para letras minúsculas, *stemização*, remoção de acentos, números e caracteres especiais. Mesmo que essa abordagem possa resultar em um processo de treinamento mais lento, a preservação do texto integral pode oferecer ao modelo uma maior compreensão do contexto, o que potencialmente pode resultar em um desempenho mais efetivo. No entanto, esta é uma hipótese que precisa ser validada por meio de mais experimentações.

Além disso, uma área de estudo que também pode ser explorada em futuras investigações é a utilização das imagens dos produtos como fonte de informação para a tarefa de classificação. As imagens contêm uma variedade de informações visuais que podem ser úteis para a identificação do departamento e tipo do produto, complementando as informações textuais. O uso de técnicas de aprendizado profundo, como redes neurais convolucionais, permitem a extração de características visuais das imagens e, consequentemente, sugerem melhorar a eficiência da classificação.

Referências

[1] W. L. de M. Cruz, "Crescimento do e-commerce no Brasil: desenvolvimento, serviços logísticos e o impulso da pandemia de Covid-19", Geo, vol. 17, nº 1, jul. 2021.

[2] W. R. das N. Santos, A.Dib "Inovação do E-commerce Brasileiro na Pandemia", Econômica, vol. 22, nº 1, 2020.

[3] Negócios SC. A porcentagem das vendas on-line no varejo por segmento. Disponível em: https://negociossc.com.br/blog/a-porcentagem-das-vendas-on-line-no-varejo-por-segmento/ Acesso em: 20/08/2022.

[4] Cortez, Eli & Herrera, Mauro & Silva, Altigran & Moura, Edleno & Neubert, Marden. (2011). Lightweight Methods for Large-Scale Product Categorization. JASIST. 62. 1839-1848. 10.1002/asi.21586.

[5] Joulin, A., Grave, E., Bojanowski, P., & Mikolov, T. (2016). Bag of Tricks for Efficient Text Classification. ArXiv. /abs/1607.01759

[6] Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. ArXiv. /abs/1301.3781

[7] Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2016). Enriching Word Vectors with Subword Information. ArXiv. /abs/1607.04606

[8] Yu, W., Z. Sun, H. Liu, Z. Li, and Z. Zheng 2018. Multi-level deep learning based e-commerce product categorization. In The SI- GIR 2018 Workshop On eCommerce co-located with the 41st International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2018), Ann Arbor, Michigan, USA, July 12, 2018

[9] GÉRON, Aurélien. Mãos à obra: aprendizado de máquina com Scikit-Learn, Keras & TensorFlow: Conceitos, ferramentas e técnicas para a construção de sistemas inteligentes. 2ª ed., Alta Books, 2021.

[10] Palmer, D. D. (2010). Text preprocessing. In Handbook of natural language processing. Chapman and Hall/CRC, 2nd edition.

[11] Barbosa, Jardeson Leandro Nascimento; Vieira, João Paulo Albuquerque; Santos, Roney Lira de Sales; Junior, Gilvan Veras Magalhães; MUNIZ, Mariana dos Santos; Moura, Raimundo Santos. Introdução ao Processamento de Linguagem Natural usando Python. In: III ESCOLA REGIONAL DE INFORMÁTICA DO PIAUÍ. 1. Anais... SBC, 2017, v. 1. p. 336-360.

[12] Viviane Orengo and Christian Huyck. 2001. A stemming algorithm for the Portuguese language. In Proceedings of 8th International Symposium on String Processing and Information Retrieval. 186–193.

[13] BARELLA, Victor Hugo. Técnicas para o problema de dados desbalanceados em classificação hierárquica. 2015. Dissertação (Mestrado em Ciências de Computação e Matemática Computacional) - Instituto de Ciências Matemáticas e de Computação, University of São Paulo, São Carlos, 2015. doi:10.11606/D.55.2016.tde-06012016-145045. Acesso em: 2023-04-10.

[14] Raschka, S. (2018). Model Evaluation, Model Selection, and Algorithm Selection in Machine Learning. ArXiv. /abs/1811.12808

[15] BAEZA-YATES, Ricardo et al. Modern information retrieval. New York: ACM press, 1999.

[16] FastText Documentation. Disponível em: https://fasttext.cc/docs/en/options.html. Acesso em: 2023-05-09.

[17] GOMES, Manoel Aquino. Classificação de produtos com base em descrições textuais. 2021. 50 f. Dissertação (Mestrado em Informática) - Universidade Federal do Amazonas, Manaus (AM), 2021.

[18] FAYYAD, U.; PIATESKY-SHAPIRO, G.;SMYTH, P.Knowledge Discovery and Mining:Towards a UniFyng Framework© 1996.

[19] ASSUNÇÃO, J.V.C. Uma breve introdução à Mineração de Dados. 1ª ed., Novatec, 2021.