UNIVERSIDADE FEDERAL DE SERGIPE

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

DEPARTAMENTO DE COMPUTAÇÃO

Uso de web scraping para mineração de produtos e preços em e-commerce

Trabalho de Conclusão de Curso

Wendel Lima Oliveira

São Cristóvão – Sergipe

2022

UNIVERSIDADE FEDERAL DE SERGIPE

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

DEPARTAMENTO DE COMPUTAÇÃO

Wendel Lima Oliveira

Uso de web scraping para mineração de produtos e preços em e-commerce

Trabalho de Conclusão de Curso submetido ao Departamento de Computação da Universidade Federal de Sergipe como requisito parcial para a obtenção do título de Bacharel em Engenharia de Computação.

Orientador(a): Leonardo Nogueira Matos
Coorientador(a): Thiago Dias Bispo

São Cristóvão – Sergipe

2022

Dedico este trabalho à minha família, primordialmente aos meus pais Gicélia Lima e Ginaldo Oliveira, a minha irmã Karla Kamilly, ao meu orientador Leonardo e ao meu coorientador Thiago.

Resumo

O desenvolvimento das tecnologias de informação e comunicação propiciou o aparecimento de sistemas colaborativos, como *crowdsourcing*, que permite o compartilhamento mútuo de dados, os mais diversos possíveis, gerados por *smartphones* de uma comunidade de usuários. Um destes sistemas mais populares é o *Waze* que permite identificar rotas de trânsito livres de congestionamento usando informações de geolocalização de aparelhos celulares. O LudiiPrice é um aplicativo *crowdsourcing* em desenvolvimento na UFS, sob coordenação dos orientadores desta monografia, que consiste em um buscador de preços de produtos de itens de consumo pessoal e gênero alimentício minerados a partir de notas fiscais eletrônicas obtidas pelo código QR fotografado por *smartphones*. Os dados das notas fiscais são armazenados em uma base de dados, usada para responder às consultas realizadas pelos usuários. O objetivo deste trabalho é realizar o povoamento automático desta base de dados usando um *web crawler* para extrair dados de preços de itens publicados em portais de *e-commerce*. Os dados minerados são processados usando técnicas de Processamento de Linguagem Natural e tem como objetivo a sugestão para cada item inserido através da nota fiscal. O crawler e as sugestões são requisitados através de uma API, que foi construída utilizando o *framework* Django que utiliza Python como linguagem. Como resultado do trabalho os *crawlers* são realizados em três *e-commerce* que populam uma base e sugere itens extraídos do *crawler* para cada produto presente na nota fiscal inserida no aplicativo LudiiPrice.

Palavras-chave: *Crowdsourcing*. *Crawler*. Processamento de linguagem natural. *e-commerce*.

Abstract

The development of information and communication technologies has led to the emergence of collaborative systems, such as crowdsourcing, which allows the mutual sharing of data, as diverse as possible, generated by smartphones of a community of users. One of these most popular systems is Waze which allows you to identify congestion-free transit routes using geolocation information from mobile devices. LudiiPrice is a crowdsourcing application that is being developed at UFS under the coordination of the supervisors of this monograph and consists of a price finder for personal consumables and groceries obtained by QR code photographed by smartphones. Invoice data is stored in a database that is used to respond to user queries. The objective of this work is to automatically populate this database using a web crawler to extract price data from products published in e-commerce portals. The data obtained is processed using Natural Language Processing techniques and aims to suggest each item entered through the invoice. The crawler and the suggestions are requested through an API, which was built using the Django framework that uses Python as its language. As a result of the work, crawlers are performed in three e-commerce that populate a base and suggest items extracted from crawler for each product present in the invoice inserted in the LudiiPrice application.

Keywords: Crowdsourcing. Crawler. Natural Language Processing. e-commerce.

Lista de ilustrações

Lista de quadros

Lista de tabelas

Lista de códigos

Lista de abreviaturas e siglas

HMTL	Linguagem de Marcação de HiperTexto
PLN	Processamento de Linguagem Natural
TF-IDF	*Term frequency - Inverse Document Frequency*
CSS	*Cascading Style Sheets*
EAN	*European Article Number*
UFS	Universidade Federal de Sergipe

Sumário

1

Introdução

O Brasil é um país de dimensões continentais rico em diversidades. Diversidades climáticas, ambientais, culturais e sociais. Historicamente se caracterizou por também ser um país rico e desigual, cuja maior parte da riqueza produzida é concentrada em uma parcela pequena da sociedade. Os anos recentes foram marcados por crises econômicas repetidas que tornaram ainda mais acirradas as desigualdades e vulnerável a sustentabilidade do trabalhador. Olhando por este ângulo, é também um país de oportunidades para o surgimento de ideias criativas para proteger a população das instabilidades da economia. Este foi o mote para a criação de um aplicativo para monitoramento de preços desenvolvido no Departamento de Computação da UFS, chamado LudiiPrice. O LudiiPrice deve permitir que seus usuários conheçam os preços praticados nos diversos supermercados da região onde vive, de modo que pode programar como fazer suas compras com a maior economia possível.

Obter os dados dos preços praticados nos supermercados é um problema desafiador. Eles existem mas não estão facilmente acessíveis. Uma primeira abordagem para obtê-los é indo procurá-los diretamente na fonte, isto é, nos bancos de dados dos fiscos estaduais. Tendo em vista que a maioria dos estabelecimentos emitem a nota fiscal eletrônica, as secretarias estaduais da fazenda possuem, portanto, todo o conteúdo necessário para levar à população a informação sobre os preços. O problema é que estas bases são privadas, estando protegidas de acesso por usuários não autorizados. Por outro lado, embora o acesso a elas seja privado, o acesso aos itens de uma nota fiscal eletrônica é público. Isto dá uma pista sobre o que pode ser feito para conseguir tais dados. Pode-se usar um mecanismo de *crowdsourcing* (BRABHAM, 2013). *Crowdsourcing* é uma forma colaborativa de levantamento e compartilhamento de dados visando o bem estar de uma comunidade de usuários. Neste caso, em particular, os usuários poderiam fotografar o código QR de um cupom fiscal, que remete à nota fiscal eletrônica hospedada no sítio web do fisco estadual, e compartilhar com os demais. Assim, se muitos usuários fizerem o mesmo, os dados almejados estariam ao alcance de todos.

Embora a solução proposta anteriormente seja válida, ela não é realizável. Isto é, em termos práticos, é inviável. Há duas razões principais para tanto. Primeiro, é necessário existir um grande e permanente engajamento dos usuários. O ato de fotografar o código QR dos cupons fiscais é uma ação deliberada e repetitiva, precisaria ser repetida a cada compra. Poucos usuários estariam motivados a fazê-la e, com poucos usuários, o levantamento colaborativo não funciona. Segundo, a comparação direta dos produtos pelo código empregado no fisco (código EAN) não permite comparar produtos similares vendidos em estabelecimentos diferentes. Há supermercados que operam com marcas próprias, que não são comercializadas pelos seus concorrentes. Produtos destas marcas não podem ser comparados com outros similares de marcas distintas, se o critério de comparação for o código EAN.

Outra forma de obter os dados procurados é buscá-los em sistemas *e-commerce*. Esta foi a solução adotada neste trabalho. Grandes redes varejistas e atacadistas permitem que seus clientes possam comprar sem sair de casa usando seus sistemas de venda pela Internet. Assim, os dados almejados podem ser obtido em outra fonte, que não as bases de dados do fisco, afinal eles estão duplicados e nem todos os acessos estão protegidos.

1.1 LudiiPrice

O projeto do LudiiPrice foi dividido em etapas, sendo as três iniciais: construção do aplicativo, criação da API de serviços e *web scraping* de produtos para preenchimento inicial da base de sugestões. As duas primeiras etapas foram desenvolvidas em outros dois TCC, Adam[1] e Rodrigo[2], já a etapa de *web scraping* foi o foco deste trabalho. Outros dois trabalhos vêm sendo feitos, que são relacionados a busca de produtos em cartazes nos *e-commerce* e detecção de marcas de produtos por meio de técnicas de aprendizado de máquina.

1.2 Objetivos

1.2.1 Objetivo geral

Desenvolver um *web scraping* para obter dados de *e-commerce* de produtos de itens pessoal e alimentício e sugeri-los para cada produtos oriundo das notas fiscais.

1.2.2 Objetivos específicos

- Criar um *web scraping* para busca de produtos e preços nos endereços eletrônicos selecionados.

[1] <https://ri.ufs.br/jspui/handle/riufs/15903>
[2] <https://ri.ufs.br/jspui/handle/riufs/15905>

- Criar base de dados de vinculação do código EAN a produtos obtidos através de *web scraping*, utilizando técnicas de Processamento de Linguagem Natural.
- Criar base de dados de sugestão de produtos obtidos através de *web scraping*, onde cada produto inserido na nota fiscal no aplicativo LudiiPrice tenha uma sugestão correspondente, utilizando técnicas de Processamento de Linguagem Natural.
- Desenvolver a API para agendamento do *web scraping* e sugestão de produtos extraídos para os produtos inseridos a partir das notas fiscais.

1.3 Metodologia

Este trabalho, portanto, visa povoar uma base de dados de sugestão inicialmente vazia ou desatualizada de forma automática, ou seja, de modo independente da participação de qualquer usuário. Isso se dá através de algumas etapas descritas nas seções seguintes.

1.3.1 Busca por endereços eletrônicos regionais

Nesta etapa foram selecionamos alguns endereços eletrônicos e os *websites* do Atacadão[3], Bompreço[4] e Mercantil Rodrigues[5] são ótimas escolhas, pois trazem um formato interessante para extração dos dados, visto que a estrutura HTML é bem comportada e possuem elementos que facilitam a interpretação e extração dos dados.

1.3.2 Extração de produtos usando *web scraping*

A extração de dados através do *web scraping* tem bastante significância, visto que a obtenção e possível atualização dos produtos se dá pela extração dessas informações. O estudo de modelos de *web scraping* é essencial para criar um modelo que solucione problemas específicos. Tendo em vista inicialmente a estrutura do código HTML dos *e-commerce* escolhidos, será criado um modelo de *web scraping* que atenda aos requisitos necessários para extração dos dados por meio da navegação no código usando elementos e atributos presentes. Na Figura 1 é mostrado um exemplo de elementos presentes em um código HTML.

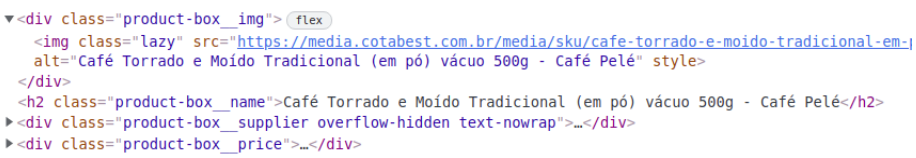

Figura 1 – Exemplo de elementos presentes no código HTML.

[3] <https://www.atacadao.com.br/>
[4] <https://www.bompreco.com.br/>
[5] <https://www.delivery.mercantilrodrigues.com.br/>

Alguns *websites* apresentam mecanismos de carregamento dinâmico de conteúdo através de *scripts*, ou seja, parte de seu conteúdo é carregado sob demanda a depender da interação do usuário. Visto isso, é preciso implementar um tipo de *web scraping* que consiga extrair os dados levando em consideração o carregamento dinâmico da página, simulando, inclusive, ações que usuário precisaria desempenhar para ter acesso aos dados desejados. Por esta razão, adotamos o Selenium[6], ferramenta que permite simular um navegador web de forma automatizada.

1.3.3 Classificação dos produtos

A classificação é um dos focos do trabalho onde visa relacionar os produtos extraídos através do *scraping* com produtos das notas fiscais inseridos no aplicativo LudiiPrice. Essa etapa visa compreender técnicas e ferramentas que auxiliam na classificação e aplicação da melhor melhor para relacionar os produtos extraídos.

1.3.4 Criação da API

A API é uma etapa importante para simplificação e agrupamento das tarefas realizadas neste trabalho, onde estará presente as etapas de extração e sugestão dos produtos e preços.

1.4 Estrutura do Documento

Para facilitar a navegação e melhor entendimento, este documento está estruturado em cinco capítulos, que são:

- Capítulo 1 - Introdução:

  Corresponde ao presente capítulo, que compreende os objetivos deste trabalho.

- Capítulo 2 - Fundamentação Teórica:

  Contém os conceitos centrais que guiam este trabalho com base em publicações relevantes para a área de estudo em questão.

- Capítulo 3 - Trabalhos Relacionados:

  Aborda os trabalhos encontrados a partir da revisão sistemática, bem como a pesquisa de mercado feita sobre o objeto de estudo deste trabalho.

- Capítulo 4 - Métodos

  Descreve os métodos usados para realização dos experimentos e criação da base de dados de produtos sugeridos.

[6] <https://www.selenium.dev/>

- Capítulo 5 - Resultados Obtidos:

  Mostra as etapas dos trabalhos realizados com seus respectivos resultados.

2

Fundamentação Teórica

Neste capítulo será discutido o uso de técnicas para extração de dados da *web* e classificação das descrições dos produtos em relação a tabela de códigos EAN.

2.1 Web Crawler

Web Crawling é um importante método para coletar dados da internet. Um *web crawler* é um programa que percorre automaticamente links na web baixando documentos e seguindo os links de uma página para outra (DHENAKARAN; SAMBANTHAN, 2011). Um *crawler* é uma ferramenta de busca e extração que precisa resolver dois problemas: primeiro, ter uma boa estratégia de *crawling*, isto é, uma estratégia para decidir quais serão as próximas páginas baixadas. Em segundo lugar, ele precisa ter uma arquitetura de sistema altamente otimizada para baixar um número grande de páginas no menor tempo possível, enquanto se mantém robusto contra travamentos, gerenciável e atento com recursos e servidores web (SHKAPENYUK; SUEL, 2002).

Os *crawlers* têm diversas funções, sendo uma delas: criar uma réplica de todas as páginas visitadas que são posteriormente processadas por alguma ferramenta de busca que irá extrair dados relevantes de cada página *web* visitada. O trabalho das ferramentas de busca é armazenar informações sobre várias páginas da web, que eles recuperam da Word Wide Web (WWW). Essas páginas são recuperadas por um web crawler que é um navegador automatizado que segue cada link que encontra (KAUSAR; DHAKA; SINGH, 2013).

2.1.1 Funcionamento do Web Crawler

No funcionamento do crawler existe basicamente três componentes principai, como mostra a Figura 2: uma fronteira que armazena a lista de links a serem visitados, o *Page Downloader* que baixa páginas da *web* e o *Web repository* que recebe páginas da *web* de um

crawler e as armazena no banco de dados (UDAPURE; KALE; DHARMIK, 2014). Os processos serão resumidos abaixo.

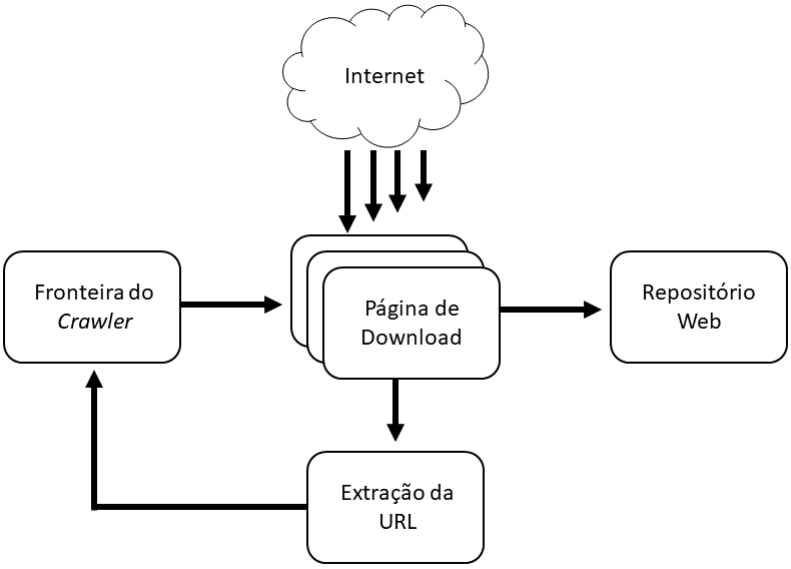

Figura 2 – Arquitetura de um *web crawler*

Fonte: Adaptado de (UDAPURE; KALE; DHARMIK, 2014)

Fronteira do *Crawler*: Contém uma lista de links não visitados. A fronteira normalmente é implementada usando uma fila FIFO, onde o crawler começa pelo início da fila e cada nova URL encontrada é adicionada ao final da fila (LEVENE; POULOVASSILIS, 2004). O ciclo de busca e extração do *link* continua até que a fronteira esteja vazia ou alguma outra condição faça com que ele pare (AVRAAM; ANAGNOSTOPOULOS, 2011).

Página de download: O principal trabalho do *page downloader* é baixar a página da internet correspondente aos *links* recuperados do *crawler frontier*. Para isso, o *page downloader* requer um cliente HTTP para enviar a solicitação HTTP e ler a resposta. Deve haver um período de tempo limite que deve ser definido pelo cliente, a fim de garantir que não levará tempo desnecessário para obtenção da página HTML (UDAPURE; KALE; DHARMIK, 2014).

Repositório *Web*: É usado para armazenar e gerenciar um grande conjunto de dados (HIRAI et al., 2011). No caso do *crawler*, esse conjunto de dados são páginas da web. O repositório armazena apenas páginas HTML padrão. Todas as outras mídias e tipos de documentos são ignorados pelo rastreador (CHO; GARCIA-MOLINA, 1999).

2.2 Processamento de Linguagem Natural (PLN)

Processamento de Linguagem Natural é uma área de pesquisa e aplicação que explora como os computadores podem ser usados para compreender e manipular texto ou fala em

linguagem natural humana. Aplicações de PLN incluem uma série de campos de estudos, como tradução automática, processamento e resumo automático de texto, recuperação de informações em diversos idiomas, reconhecimento de voz, entre outras aplicações (CHOWDHURY, 2003).

Tradicionalmente a análise de PLN é dividida em estágios, olhando principalmente a distinção linguística, sendo eles: sintaxe, semântica e a pragmática. Em (INDURKHYA; DAMERAU, 2010), é proposta uma divisão mais elaborada sobre as etapas do processamento de linguagem natural.

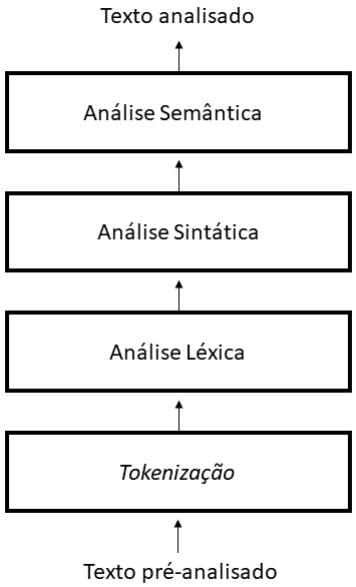

Figura 3 – Estados da análise de PLN.

- **Processamento de texto**: é notável quem em certos idiomas as palavras não são delimitadas por espaços como é na língua inglesa, por exemplo. Nessa etapa o principal objetivo é criar os *tokens* que divide as palavras, usando técnicas de processamento de texto.
- **Análise Léxica**: no item anterior foi feita a *tokenização*, mas sabendo que as palavras não são atômicas, também é necessário analisar cada palavra usando técnicas que estuda a morfologia e fonologia.
- **Análise Sintática**: nesta parte, extrair o significado de uma frase é uma questão fundamental. Em uma abordagem de PLN é geralmente determinada a estrutura sintática ou gramatical de cada frase.
- **Análise Semântica**: se refere a análise dos significados das palavras, expressões fixas, frases inteiras e enunciados, ou seja, significa traduzir expressões originais em metalinguagem semântica.

Abaixo estão descritas algumas técnicas que foram usadas para transformar e comparar os produtos com base na sua descrição.

2.2.1 TF-IDF

O uso do TF-IDF espera extrair os termos dos documentos, de forma que os documentos representem linhas e os termos sejam colocados em colunas. Os termos acabam acrescentando um grande número de colunas, o que causa o problema dimensional e diminui a eficiência do algoritmo.

Para reduzir esses termos, a técnica TF-IDF é usada, técnica essa que pondera com valores menores os termos mais comuns no *corpus* e pondera com valores maiores os termos com maior frequência no documento e menor no corpus (AIZAWA, 2003). Essencialmente, o TF-IDF funciona determinando a frequência relativa de palavras em um documento específico em comparação com a proporção inversa dessa palavra em todo o corpus do documento (RAMOS, 2003).

O procedimento para implementar o TF-IDF tem diferenças em relação a suas aplicações, mas a abordagem geral funciona da seguinte maneira. Dada uma coleção de documentos D, uma palavra w e um documento individual $d \in D$, calculamos

$$w_d = f_{w,d} * \log \frac{|D|}{f_{w,D}} \tag{2.1}$$

onde $f_{w,d}$ é igual ao número de vezes que w aparece em d, $|D|$ é o tamanho do conjunto de documentos, e $f_{w,D}$ é igual ao número de documentos em que w aparece em D (SALTON; BUCKLEY, 2000).

2.2.2 Word Embedding

Construir *word embeddings* sempre gerou muito interesse para linguistas, frequentemente vistas como um espaço vetorial de baixa dimensão, onde as dimensões são recursos que potencialmente descrevem propriedades sintáticas ou semânticas (LEBRET; COLLOBERT, 2013). Em PLN, *word embedding* é um termo usado para a representação de palavras para análise de texto, normalmente na forma de um vetor de valor real que codifica o significado da palavra, de modo que as palavras que estão mais próximas no espaço vetorial devem ter significado semelhantes (JURAFSKY; MARTIN, 2018).

BERT, que significa Bidirectional Encoder Representations from Transformers, foi projetado para pré-treinar representações bidirecionais profundas de texto não rotulado, condicionando conjuntamente o contexto esquerdo e direito em todas as camadas (DEVLIN et al., 2019). Para o uso de um modelo BERT em português brasileiro pré-treinado foi usado o BERTimbau (SOUZA;

NOGUEIRA; LOTUFO, 2020) que ao lançar os modelos para a comunidade tem a a esperança de fornecer bases sólidas para futuras pesquisas de PNL.

2.3 Expressão Regular

A expressão regular é um tipo de padronização muito usada na ciência da computação. Uma expressão regular é uma notação algébrica para caracterizar um conjunto de strings. Elas são particularmente usadas para pesquisar textos, quando temos um padrão para pesquisar dentro do texto (JURAFSKY; MARTIN, 2018).

2.3.1 Padrões de expressões regulares

O tipo mais simples de expressão regular é uma sequência de caracteres simples. Delimitando a expressão regular entre barras a pesquisa é feita da seguinte forma: `/sequence_char/`. As expressões regulares são ***case sensitive***, ou seja, se diferencia entre maiúsculo e minúsculo; `/s/` é diferente de `/S/`. Isso significa que o padrão `/sapatos/` não corresponderá à palavra Sapatos. Isso pode ser resolvido

RE	**Combinação**	**Exemplo**
/sapatos/	sapatos	'Eu fui comprar sapatos.'
/a/	a	'A casa de Jorge é bonita.'

Tabela 1 – Expressões regulares de sequência simples

colocando os caracteres que pode podem conter o carácter maiúsculo ou minúsculo entre [], onde o que tiver dentro do colchetes é feita uma disjunção.

RE	**Combinação**	**Exemplo**
/[sS]apatos/	sapatos ou Sapatos	'Sapatos são feitos em pares.'
/[abc]/	a ou b ou c	'A bola é redonda.'

Tabela 2 – Expressões regulares com disjunção de caracteres

Em expressões regulares como `/[ABCDEFGHIJKLMNOPQRSTUVWXYZ]` pode ser feita uma redução para simplificar a expressão.

RE	**Combinação**	**Exemplo**
/[A-Z]/	uma letra maiúscula	'O jogo é amanhã'
/[a-z]/	uma letra minúscula	'998a877 b.'
/[0-9]/	um dígito	'O começo do capítulo era na pág. 5.'

Tabela 3 – Expressões regulares com disjunção e abreviação

Para casos de elementos opcionais é necessário o uso de ? logo após o carácter ou disjunção de caracteres. E para casos de múltiplos carácteres é necessário o uso de * .

RE	**Combinação**	**Exemplo**
/sapatos?/	sapato ou sapatos	'O sapato é azul.'
/[0-9][0-9]*/	qualquer número	'O rapaz estava na posição 1000 da fila.'

Tabela 4 – Expressões regulares com multiplicidade e caracteres opcionais

2.4 N-gram

Um N-gram é uma sequência de N *tokens* de um documento mais longo (CAVNAR, 1994).

Assim, podemos representar a palavra "DOCUMENTO"em sequências de caracteres, como:

```
uni-gram:  D, O, C, U, M, E, N, T, O;
bi-gram:   DO, OC, UM, ME, EN, NT, TO;
tri-gram:  DOC, OCU, CUM, UME, MEN, ENT, NTO.
```

O n-gram tem várias aplicações, mas neste trabalho o importante é apenas a decomposição das palavras em sequências de caracteres.

2.5 Similaridade do Cosseno

O cálculo de similaridade é um componente básico para muitos aplicativos de mineração de texto (LI; HAN, 2013). Quando os documentos são representados como vetores de termos, a similaridade de dois documentos corresponde à correlação entre os vetores. A similaridade de cosseno é uma das medidas de similaridade mais populares aplicadas a documentos de texto, como em inúmeras aplicações de recuperação de informação (HUANG, 2008).

Tendo dois documentos representados em forma de vetores $\vec{v_a}$ e $\vec{v_b}$, a similaridade do cosseno é

$$cos\_sim(\vec{v_a}, \vec{v_b}) = \frac{\vec{v_a}.\vec{v_b}}{|\vec{v_a}|.|\vec{v_b}|} \tag{2.2}$$

A similaridade do cosseno, como outras ferramentas de medida, são importantes para calcular aproximação entre vetores que representam documentos, e assim, inferir algo sobre a proximidade entre eles.

2.6 Distância de Edição

Editar distância permite a quantificação da intuição sobre a similaridade das *strings*. Mais formalmente, a distância mínima de edição entre duas strings é definida como o número mínimo de operações de edição (operações como inserção, exclusão, substituição) necessárias para transformar uma *string* em outra (JURAFSKY; MARTIN, 2018).

Cada operação pode ter um peso atribuído a si. A distância de Levenshtein entre duas sequências é o fator de ponderação mais simples em que cada uma das três operações tem um custo de 1 (LEVENSHTEIN, 1966).

2.6.1 Algoritmo de Distância de Edição Mínima

Dadas duas strings, a string de origem X de comprimento n e a string de destino Y de comprimento m, definiremos $D[i, j]$ como a distância de edição entre $X[1..i]$ e $Y[1..j]$, ou seja, os primeiros i caracteres de X e os primeiros j caracteres de Y. A distância de edição entre X e Y é, portanto, $D[n, m]$ (WAGNER; FISCHER, 1974).

Levando em conta que i varia de 0 até n e j varia de 0 até m. O valor de $D[i, j]$ é calculado tomando o mínimo dos três caminhos possíveis através da matriz que chegam lá:

$$D[i,j] = min \begin{cases} D[i-1,j] + del\_cost(source[i]) \\ D[i,j-1] + ins\_cost(target[j]) \\ D[i-1,j-1] + sub\_cost(source[i], target[j]) \end{cases} \tag{2.3}$$

O Quadro 1 mostra a comparação entre as *strings* telhado e antenado usando a técnica de distância de edição como base para o calculo das operações.

Quadro 1 – Matriz das operações

-	**-**	**A**	**N**	**T**	**E**	**N**	**A**	**D**	**O**
-	0	1	2	3	4	5	6	7	8
T	1	1	2	2	3	4	5	6	7
E	2	2	2	3	2	3	4	5	6
L	3	3	3	3	3	3	4	5	6
H	4	4	4	4	4	4	4	5	6
A	5	4	5	5	5	5	4	5	6
D	6	5	5	6	6	6	5	4	5
O	7	6	6	6	7	7	6	5	4

O quadro mostra os valores das distâncias de acordo com cada letra de cada palavra em relação a outra, onde na última posição da matriz ($D[n, m]$) mostra a quantidade mínima de operações para transformar `telhado` em `antenado`, que são no mínimo 4 operações.

2.7 Coeficiente de Jaccard

O coeficiente de Jaccard mede a similaridade usando a intersecção e união dos objetos. Para documentos de texto, o coeficiente de Jaccard compara o peso da soma dos termos compartilhados com o peso da soma dos termos que estão presentes em qualquer um dos dois documentos (LI; HAN, 2013).

Tendo dois documentos representados em forma de vetores $\vec{v_a}$ e $\vec{v_b}$, o coeficiente de Jaccard é

$$jac\_coef(\vec{v_a}, \vec{v_b}) = \frac{\vec{v_a}.\vec{v_b}}{|\vec{v_a}|^2.|\vec{v_b}|^2 - \vec{v_a}.\vec{v_b}} \tag{2.4}$$

O coeficiente de Jaccard varia entre 0 e 1. É 1 quando $\vec{v_a} = \vec{v_b}$ e 0 quando $\vec{v_a}$ e $\vec{v_b}$ são disjuntos.

3

Trabalhos Relacionados

O objetivo deste capítulo é citar trabalhos ou ferramentas que de alguma forma façam ou utilizem base de dados de produtos usando suas informações para uso em aplicações próprias ou repasse para clientes, e extração de dados em *e-commerce* com o uso de *web crawler*. As pesquisas foram feitas usando *google advanced search*, *google scholar* e a base de dados do INPI (Instituto Nacional da Propriedade Industrial).

Para tentar encontrar trabalhos relacionados foi usado o *google advanced search* com os seguintes conjuntos de palavras: banco de dados da tabela ean, *web scraping* de dados de produtos de *e-commerce*, banco de dados OR tabela ean, dados OR tabela ean OR produtos OR ecommerce, web data extraction for content aggregation; o mesmo foi usado para o INPI e para o *google scholar*, mas feita a pesquisa em inglês. Não foram obtidos trabalhos ou ferramentas com a mesma proposta deste trabalho, mas sim, ferramentas relacionadas.

Na seção 3.1 é mostrado um trabalho relacionado com bastante ênfase na extração de dados de *e-commerce* e nas subseções 3.2.1 e 3.2.2, foram mencionadas as empresa Bluesoft e GS1 Brasil, respectivamente, que realizam trabalhos relacionados a gestão da rede de fornecedores, varejistas e comerciantes; tendo um catálogo de produtos que fornece informações como: código EAN, preço médio, marca, descrição e muito mais. Na subseção 3.2.3, foi mencionado um artigo que tem como parte do trabalho extração de dados de *e-commerce*. Usa técnicas de *scraping* para obter os dados na *web*

3.1 Web Data Extraction From E-Commerce Websites

Na tese de doutorado (VIIKMAA, 2016), é mostrado um modelo de extração de dados com o uso de *web crawler*, ou seja, busca endereços eletrônicos e através deles são realizadas buscas por outros sites e produtos, assim é feita uma varredura em um maior número de *e-commerce*. O modelo navega por meio dos elementos e atributos inclusos no HTML das páginas *web*, onde

encontra as informações dos produtos e outros *e-commerce*.

É introduzido o uso do sistema de extração de informações `ZedBot`, que permite a extração de dados altamente estruturados de páginas *web*. Atende os principais requisitos definidos para o sistema moderno de extração de dados: é independente de plataforma, possui um sistema de geração de sub rotinas semiautomático e possui interface de usuário que facilita o uso. O *web crawler* projetado permite que a extração seja realizada em todas as partes do site sem interação humana.

3.2 Ferramentas Relacionadas

3.2.1 Base de dados Bluesoft Cosmos

A empresa Bluesoft como mencionada acima, trabalha com a gestão de redes que trabalham com o comércio de diversos produtos, sendo assim, a empresa apresenta um catálogo variado de produtos e suas informações. A Bluesoft Cosmos trabalha justamente com a distribuição destas informações através de uma API e que pode ser usada de forma gratuita ou paga, havendo limitações das requisições. A estrutura básica usada é da seguinte forma.

```
{
    "description": "FEIJAO CARIOCA PETRUS 1K T1",
    "gtin": "7898937265035",
    "thumbnail": "https://api.cosmos.bluesoft.com.br/...",
    ...
    "ncm": {
        "code": "07133399",
        "description": "Outros",
    ...
}
```

Código 1 – Estrutura dos produtos da Bluesoft.

No exemplo mostrado acima foi deixada apenas as informações mais relevantes para este trabalho, onde relaciona uma descrição de um produto a um único código GTIN/EAN. Essa API tem diversos serviços para montar a requisição, sendo assim, é possível realizar a consulta através do código EAN, retornando um ou mais produtos relacionados, através do código NCM ou usando a descrição do produto.

3.2.2 Base de dados GS1 Brasil

A GS1 é uma empresa que está presente em vários países, dentre eles o Brasil. A GS1 é conhecida pela criação e consulta de códigos de barras de forma global entre indústria e varejo. A GS1 também ajuda empresas na identificação de produtos virtualmente e melhora a experiência do consumidor, compartilhando informações confiáveis de produtos.

Os produtos na GS1 tem como campos marca, descrição, link para imagem, código EAN e peso. A empresa trabalha de forma parecida com a Bluesoft que foi citada acima, mas com o diferencial da parte referente a criação de novos códigos de barras.

3.2.3 Web Scraping em E-Commerce

Essa subseção usa como base (ONYENWE et al., 2021) onde foi usada técnicas de *scraping* para extrair dados de um *e-commerce*, mas com o intuito diferente desse trabalho. Para a extração de dados o autor leva em conta três principais etapas que são: mapeamento das páginas web, desenvolvimento do web crawler e processamento dos dados obtidos. Na etapa de processamento dos dados obtidos é usada a biblioteca do Python *BeautifulSoup()*, que tem como base a extração de texto de páginas HMTL e XML. Essas etapas são mostradas e explicadas abaixo.

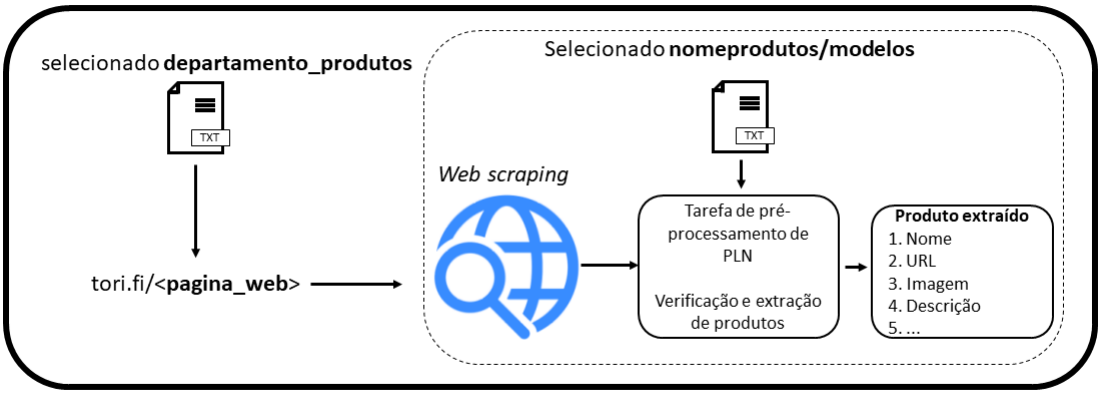

Figura 4 – Arquitetura de um *web crawler* usada por (ONYENWE et al., 2021)

- **Mapeando de páginas web**: produtos de comércio eletrônico normalmente são categorizados em diferentes departamentos. Com o estudo do HTML da página usando um navegador dá para ter noção de como é o comportamento padrão da página e com isso é possível a extração de dados com *tags* específicas. Na imagem abaixo mostra um exemplo de *tags* extraídas pelo autor.

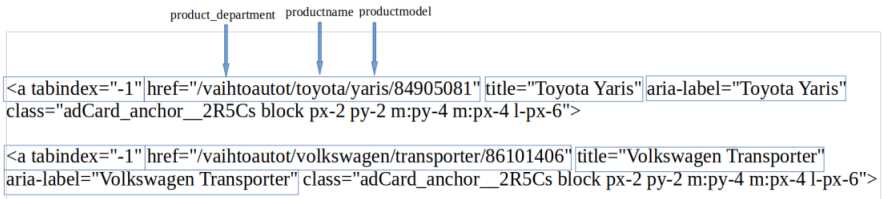

Figura 5 – Exemplo de código fonte extraído do *website*.

- **Desenvolvimento do web crawler**: essa etapa é divida em seis partes, sendo elas:

1 - Faça a requisição de uma URL específica.

2 - Baixe o conteúdo que é retornado.

3 - Identifique os atributos que facilitam a navegação pelo código HTML e que ajudam a encontrar os elementos desejados dentro da página *web*.

4 - Verifique se os produtos/modelos estão nos nomes dos produtos.

5 - Extraia o conteúdo desses elementos e faça a analise.

6 - Descarte a página da web

- **Processamento dos dados obtidos**: é usada como base a função *BeautifulSoup()* e com ela pode se analisar e extrair tags específicas. Na imagem abaixo o autor mostra um pequeno exemplo do uso da função.

```
...
r = requests.get(hre_)
soup = BeautifulSoup(r.content, "html.parser")
for div in soup._ndAll("div", class ="date-cat-container"):
  for div in div._ndAll("div", class = "date image"):
...
```

Figura 6 – Exemplo de código do *web crawler.*

4
Método

Neste capítulo será descrito os métodos usados para obtenção dos dados por meio do *web crawler* e classificação através de técnicas de vetorização e comparação vetorial. Na seção 4.1 é discutida a abordagem de atribuição de código EAN para produtos extraídos dos *e-commerce*, já na seção 4.2 segue a abordagem de sugestões de produtos extraídos dos *e-commerce* para cada produto das notas fiscais inseridas no aplicativo LudiiPrice.

4.1 Experimentos para atribuição do código EAN

Os experimentos descritos a seguir foram realizados para detectar a viabilidade de classificação de produtos com textos bem comportados, ou seja, cuja descrição do produto seja completa.

No primeiro momento foi pensado em usar uma base de dados que contém o código EAN presente para cada produto, mas como não existe uma base oficial disponibilizada para estudo, onde possa ser encontrada facilmente produtos, a alternativa foi comprar uma base de dados não oficial que contém o EAN e a descrição para cada produto.

4.1.1 Base de dados comprada

A base comprada[1] possui em sua composição um campo para o código EAN, código NCM (Nomenclatura Comum do Mercosul) e descrições com tipos de representações alternativos, sendo elas: contendo acentuação, maiúscula, minúscula.

[1] <https://produto.mercadolivre.com.br/MLB-776990445-banco-de-dados-prod-supermercado-79-mil-produtos-fotos-_JM>

codbar	descricao	desc_sem_acento	desc_upper	desc_upper_sem_acent	desc_lower	desc_lower_sem_acento	ncm
7892840001407	Biscoito Caseiro Du Bebel 140Gr	Biscoito Caseiro Du Bebel 1	BISCOITO CASEIRO DU BEB	BISCOITO CASEIRO DU BEB	biscoito caseiro du bebel 1	biscoito caseiro du bebel 1	19053100
7896524702017	Limpa Vidros Suprema 500Ml	Limpa Vidros Suprema 50	LIMPA VIDROS SUPREMA 5	LIMPA VIDROS SUPREMA 5	limpa vidros suprema 500	limpa vidros suprema 500	34022000
7891098010438	Chá Leão 15 Saq Carqueja	Cha Leao 15 Saq Carqueja	CHÁ LEÃO 15 SAQ CARQU	CHA LEAO 15 SAQ CARQU	chá leão 15 saq carqueja	cha leao 15 saq carqueja	00902100
7899060504640	Carne Moida Bov.Cong Frivasa 500G	Carne Moida Bov.Cong Friv	CARNE MOIDA BOV.CONG F	CARNE MOIDA BOV.CONG F	carne moida bov.cong friv	carne moida bov.cong friv	02012090
7891164002848	Alcatra Temp Churrasco Facil Aurora 800Gr	Alcatra Temp Churrasco FA	ALCATRA TEMP CHURRASC	ALCATRA TEMP CHURRASC	alcatra temp churrasco fa	alcatra temp churrasco fa	02013000
7894351056457	Carpaccio Minerva 340Gr	Carpaccio Minerva 340Gr	CARPACCIO MINERVA 340	CARPACCIO MINERVA 340	carpaccio minerva 340gr	carpaccio minerva 340gr	02023000
7896079902269	Carpaccio Salmao Damm 100G	Carpaccio Salmao Damm 1	CARPACCIO SALMAO DAMM	CARPACCIO SALMAO DAMM	carpaccio salmao damm 1	carpaccio salmao damm 1	02023000
7896079902382	Carpaccio Haddock Damm 100 G	Carpaccio Haddock Damm	CARPACCIO HADDOCK DAM	CARPACCIO HADDOCK DAM	carpaccio haddock damm	carpaccio haddock damm	02023000
7899567209536	Carne Moida Friboi 500 Gr	Carne Moida Friboi 500 Gr	CARNE MOIDA FRIBOI 500	CARNE MOIDA FRIBOI 500	carne moida friboi 500 gr	carne moida friboi 500 gr	02023000
7891164002879	Costela Suina Churrasco Facil Aurora 800	Costela Suina Churrasco FA	COSTELA SUINA CHURRAS	COSTELA SUINA CHURRAS	costela suina churrasco fa	costela suina churrasco fa	02031200
7893000768215	Costela Grill Sadia 800Gr	Costela Grill Sadia 800Gr	COSTELA GRILL SADIA 800	COSTELA GRILL SADIA 800	costela grill sadia 800gr	costela grill sadia 800gr	02031900
7894904009596	Costelinha Seara Suina Temperada 1Kg (Kg 1.000)	Costelinha Seara Suina Te	COSTELINHA SEARA SUINA	COSTELINHA SEARA SUINA	costelinha seara suina tem	costelinha seara suina tem	02031900
7894904070572	Lombo Suino Seara Ao Molho Barbecue 1Kg	Lombo Suino Seara Ao Mol	LOMBO SUINO SEARA AO M	LOMBO SUINO SEARA AO M	lombo suino seara ao molh	lombo suino seara ao molh	02032200
7891164001087	Costela Aurora Especial 1Kg (Kg 1.000)	Costela Aurora Especial 1K	COSTELA AURORA ESPECIA	COSTELA AURORA ESPECIA	costela aurora especial 1k	costela aurora especial 1k	02032900
7891164001193	Filezinho Suino Aurora 1Kg	Filezinho Suino Aurora 1Kg	FILEZINHO SUINO AURORA	FILEZINHO SUINO AURORA	filezinho suino aurora 1kg	filezinho suino aurora 1kg	02032900
7891164028206	Picanha Suina Aurora : 101481	Picanha Suina Aurora : 1	PICANHA SUINA AURORA :	PICANHA SUINA AURORA :	picanha suina aurora : 10	picanha suina aurora : 10	02032900
7893000010192	Bisteca Sem Tempero Sadia 1Kg	Bisteca Sem Tempero Sad	BISTECA SEM TEMPERO SA	BISTECA SEM TEMPERO SA	bisteca sem tempero sadia	bisteca sem tempero sadia	02032900
7893000018761	Carne Sadia Congelada De Suino Bisteca C/Osso 1Kg (Un	Carne Sadia Congelada De	CARNE SADIA CONGELADA	CARNE SADIA CONGELADA	carne sadia congelada de	carne sadia congelada de	02032900
7893000084742	Pernil Facil Desossado Sadia	Pernil Facil Desossado Sad	PERNIL FACIL DESOSSADO	PERNIL FACIL DESOSSADO	pernil facil desossado sadia	pernil facil desossado sadia	02032900
7893000517059	Costela Sadia Congelada Kg (Un 1.000)	Costela Sadia Congelada K	COSTELA SADIA CONGELA	COSTELA SADIA CONGELA	costela sadia congelada kg	costela sadia congelada kg	02032900
7894904001767	Bisteca Suina Seara Cong Kg (Un 4.000)	Bisteca Suina Seara Cong K	BISTECA SUINA SEARA CO	BISTECA SUINA SEARA CO	bisteca suina seara cong k	bisteca suina seara cong k	02032900
7896314344489	Copa Lombo Pamplona	Copa Lombo Pamplona	COPA LOMBO PAMPLONA	COPA LOMBO PAMPLONA	copa lombo pamplona	copa lombo pamplona	02032900
7891515016715	Peito Perdigao Chester Phv Kg	Peito Perdigao Chester Phv	PEITO PERDIGAO CHESTER	PEITO PERDIGAO CHESTER	peito perdigao chester phv	peito perdigao chester phv	02071200
7893000081147	Frango Facil Inteiro Alho/Ceb/Ervas Sadia	Frango Facil Inteiro Alho/C	FRANGO FACIL INTEIRO AL	FRANGO FACIL INTEIRO AL	frango facil inteiro alho/ceb	frango facil inteiro alho/ceb	02071200
7891515503659	File Peito Perdigao Sassami 1Kg (Un 1.000)	File Peito Perdigao Sassam	FILE PEITO PERDIGAO SAS	FILE PEITO PERDIGAO SAS	file peito perdigao sassam	file peito perdigao sassam	02071300
7891515614096	Coxinha Da Asa Bandeja 1 Kg Perdigao	Coxinha Da Asa Bandeja 1	COXINHA DA ASA BANDEJA	COXINHA DA ASA BANDEJA	coxinha da asa bandeja 1	coxinha da asa bandeja 1	02071300
7891515614126	Coração De Frango Perdigao 1Kg	Coracao De Frango Perdiga	CORAÇÃO DE FRANGO PER	CORACAO DE FRANGO PER	coração de frango perdiga	coracao de frango perdiga	02071300

Figura 7 – Trecho da base comprada.

Os produtos contidos na base comprada foram formados a partir de dados extraídos de comércios físicos e eletrônicos, onde contém a tabela e fotos dos produtos. A tabela contá com um pouco mais de oitenta mil produtos variados.

4.1.2 Vetorização TF-IDF

Os primeiros testes foram feitos usando dados extraídos manualmente dos *e-commerce* para assim ser feita a vetorização usando TF-IDF.

Os parâmetros foram validados manualmente através da avaliação da qualidade da comparação entre produtos. Como parâmetro do TF-IDF foi usado a separação do vocabulário usando n-gramas de 1 a 3 caracteres. O pre-processamento das descrições foi feito filtrando-se caracteres especiais, acentuação e priorizando o uso de letra minúscula. O vocabulário foi construído a partir da base comprada descrita na seção 4.1.1.

Um exemplo de vetorização usando TF-IDF é mostrado nas Figuras 8 e 9 para as descrições:

1-`CHA LEAO 15 SAQ CARQUEJA`

2-`CARNE MOIDA FRIBOI 500 GR`

Descrição	Contador (n-gram)
1	(1,1,1,0,0,0,0,0,1,1,1,1,0,0,0,0,1,1,1,0,0,0,1,1)
2	(0,0,0,1,1,1,1,1,0,0,0,0,1,1,1,1,0,0,0,1,1,1,0,0)

Figura 8 – Contador n-gram das descrições.

Descrição	Vetor (TF-IDF)
1	(0.29, 0.29, 0.29, 0, 0, 0, 0, 0, 0.29, 0.29, 0.29, 0.29, 0, 0, 0, 0, 0.29, 0.29, 0.29, 0, 0, 0, 0.29, 0.29)
2	(0, 0, 0, 0.29, 0.29, 0.29, 0.29, 0.29, 0, 0, 0, 0, 0.29, 0.29, 0.29, 0.29, 0, 0, 0, 0.29, 0.29, 0.29, 0, 0)

Figura 9 – Vetor após uso do TF-IDF nas descrições.

Na Figura 8, é mostrado o contador de n-gram presentes em cada descrição, baseado no dicionário criado, já na Figura 9, é mostrado o vetor normalizado após aplicação do TF-IDF.

4.1.3 Web scraping da base de dados

Após os testes de vetorização foi introduzido o uso dos itens extraídos a partir do *web crawler*. O método de extração inicial foi feito usando como base categorias escolhidas em cada *e-commerce*, onde percorria cada URL correspondente as categorias e logo após era extraída a primeira página de produtos que era mostrada.

Figura 10 – Categorias no Atacadão.

Figura 11 – URL da categoria Mercearia no Atacadão.

Outro método escolhido para extração foi usar a pesquisa por texto disponibilizada pelo *e-commerce*, onde foi elencado categorias gerais e com isso foi realizada a pesquisa e extração dos dados. Também foi realizado *scraping* em dois *e-commerce*, onde o primeiro serve como base para pesquisa e o outro realiza a pesquisa pela busca disponibilizada no *website* e extrai os produtos.

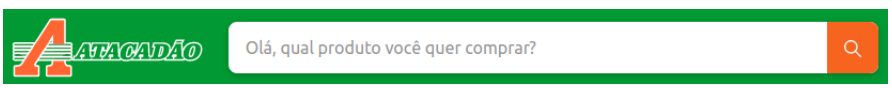

Figura 12 – Campo de pesquisa de produtos no Atacadão.

Ao longo dos testes foi necessário o uso do Selenium para servir como intermediador dos passos anteriores a extração dos dados. Os *e-commerce* escolhidos no projeto usam um método de mostrar os produtos ao longo do carregamento da página, ou seja, os sites podem carregar a

página primeiramente e só depois carregar os produtos. *Scrapy*, sozinho, tenta extrair os dados após a página ser carregada, podendo assim, não conseguir extrair nenhum dado de produto.

```
####options for Chrome Driver####
options = webdriver.ChromeOptions()
options.add_argument("headless")
options.add_argument("window-size=1920,1080")

driver = webdriver.Chrome(chrome_options=options,
    executable_path=ChromeDriverManager().install())

driver.get('https://www.siteaqui.com.br/')

wait = WebDriverWait(driver, 30)
```

Código 2 – Configuração do Selenium usando o driver do Chrome.

Com o uso do *Selenium* é possível que uma página ao carregar fique esperando até que encontre um determinado elemento **HTML**, como: `div`, `a`, `body`, `spam` ou até mesmo um atributo pertencente a um elemento, como: `href`, `id`, `class`,entre outros.

O *Selenium* foi configurado para sempre que for usar um elemento **HTML** através do mesmo ou por meio algum de seus atributos, verifica primeiro a existência por meio de uma variável de espera (`wait`), que tem um quantidade de tempo máximo de espera, para só assim acessar o determinado elemento.

```
####Uso do wait para espera de um elemento.
#espera
wait.until(EC.presence_of_element_located((By.CSS_SELECTOR,
    "span#state")))
#execucao de comando
driver.find_element_by_css_selector("span#state").click()
```

Código 3 – Uso do *wait* em um elemento HMTL.

4.1.4 Vetorização com o BERT

Um segundo método para vetorização das descrições dos produtos foi usando *word embeddings* com o modelo pré-treinado BERT (Bidirectional Encoder Representations from Transformers). O uso foi baseado na sua versão pré-treinada para português do Brasil, BERTimbau[2].

O uso desse método foi descontinuado devido a pequena distância entre produtos semelhantes e não semelhantes em relação ao vetores TF-IDF. A distância entre vetores será mencionada em detalhes na seção 4.1.5.

[2] <https://huggingface.co/neuralmind/bert-base-portuguese-cased>

4.1.5 Sugestão de produtos mais semelhantes

O método escolhido para checar a semelhança entre as descrições foi avaliação da similaridade do cosseno: os vetores são normalizados e a similaridade do cosseno calculada entre cada par de vetor TF-IDF.

Quanto menor o ângulo entre os vetores das descrições dos produtos, maior a similaridade entre eles. Podemos inferir, neste cenário, que as descrições dos respectivos produtos possuem grafias semelhantes, ou seja, contém mais palavras em comum que diferentes em relação ao vocabulário. O raciocínio inverso pode ser aplicado para vetores com ângulos maiores: suas descrições divergem e, portando, são dissimilares.

Conforme explicado anteriormente, a similaridade do cosseno se compreende no intervalo $[-1, 1]$ (numa interpretação livre: nada similar a muito similar). Entretanto, a natureza dos vetores TF-IDF gerados nos experimentos aqui descritos produz valores de similaridade compreendidas sempre no intervalo $[0, 1]$.

4.1.6 Criação e classificação de uma base de dados de atribuição

Para a criação da base de dados de atribuição do código EAN foi preciso determinar um limiar de aceitação para a similaridade do cosseno. Esse limiar mínimo de aceitação é de suma importância para mostrar confiabilidade das descrições atribuídas aos seus determinados códigos EAN.

Na hipótese inicial se estabeleceu um limiar de confiabilidade na vinculação das descrições. Na prática, o referido limiar consiste da distância mínima de aceitação para considerarmos, no contexto deste experimento, as descrições do produto como iguais. Vale relembrar que, na similaridade do cosseno, quanto mais próxima ao valor 1, mais semelhantes são as descrições comparadas. Dada essa hipótese foi montada uma divisão de classes de acordo com o valor da similaridade do cosseno.

Nesse ponto, analisando os resultados obtidos, foi visto a necessidade de uma outra abordagem para sugestão de produtos, pois com a conclusão dos experimentos foi visto a inconfiabilidade de um limiar para atribuição do código EAN. A nova abordagem está descrita na seção 4.2.

4.2 Experimentos para sugestão de produtos

Visto a necessidade de uma outra forma para prover a sugestão de produtos foi criado um novo método. A partir das notas fiscais contidas no LudiiPrice é construído o vocabulário e a cada nova inserção de nota fiscal é sugerido para cada produto um outro produto da base de dados extraída do *scraping* que possui uma maior semelhança seguindo os critérios que serão descritos nas seções seguintes.

4.2.1 Obtenção da base dados

O *crawler* para cada e-commerce foi desenvolvido seguindo os seguintes passos: configuração do Selenium, inserção do CEP para localização da loja física mais próxima, obtenção das URLs e extração dos produtos para cada URL anteriormente obtida. O Selenium mencionado na seção 4.1.3 foi mantido, mas com algumas diferenças devido à inserção de novas etapas.

Os e-commerce selecionados usam o CEP ou localidade para selecionar o comércio físico mais próximo, no caso do **Atacadão** ele usa um sistema de selecionar estado e cidade, já no **BomPreço** e **Mercantil Rodrigues** é usado o CEP. Para padronizar a busca por CEP teve que ser feita uma adaptação no **Atacadão** onde primeiro é feita uma busca por CEP em uma API dos correios e depois é usado o *Selenium* para fazer a escolha do estado e cidade a partir do resultado extraído da API. A inserção do CEP depende da disponibilidade dos e-commerce, ou seja, caso não haja uma resposta para aquele determinado CEP a variável de espera (`wait`) vai chegar até o seu tempo limite e lançar uma exceção.

```
1  time.sleep(5)
2  wait.until(EC.presence_of_element_located((By.CSS_SELECTOR,
       "span#select2-input-state-container")))
3  driver.find_element_by_css_selector("span#select2-input-state-container").click()
4
5  wait.until(EC.presence_of_element_located((By.XPATH, "//li[text()='"+ uf +"']")))
6  driver.find_element_by_xpath("//li[text()='"+ uf +"']").click()
7
8  wait.until(EC.presence_of_element_located((By.CSS_SELECTOR,
       "#select2-input-city-container")))
9  driver.find_element_by_css_selector("#select2-input-city-container").click()
10
11 wait.until(EC.presence_of_element_located((By.XPATH, "//li[text()='"+ loc +"']")))
12 driver.find_element_by_xpath("//li[text()='"+ loc +"']").click()
13
14 wait.until(EC.presence_of_element_located((By.CSS_SELECTOR, "input[value='Confirmar
       minha escolha']")))
15 driver.find_element_by_css_selector("input[value='Confirmar minha
       escolha']").click()
```

Código 4 – Inserção do CEP no Atacadão.

```
1  time.sleep(5)
2  wait.until(EC.presence_of_element_located((By.CLASS_NAME, "vtex-cep")))
3  driver.find_element_by_css_selector("input.vtex-cep").send_keys(self.cep)
4  driver.find_element_by_css_selector("button.vtex-whiteLB").click()
```

Código 5 – Inserção do CEP no Bompreço.

Conforme podemos observar comparando os Códigos 5 e 6, a diferença na quantidade de instruções é significativa, fato que é justificado em razão do Atacação não disponibilizar funcionalidade de inserção de CEP para seleção da região desejada pelo usuário.

O método usado para extração das URLs foi baseado nas categorias existentes nos *e-commerce*, onde após ser inserido o CEP é feito uma varredura nas categorias que são colocadas

em um vetor que será percorrido posteriormente onde irá ser feita a extração dos dados dos produtos.

```
wait.until(EC.presence_of_element_located((By.CSS_SELECTOR,
    "div.category a")))
all_links = driver.find_elements_by_css_selector("div.category a")
```

Código 6 – Extração das URLs dos e-commerce.

O modelo de extração das URLs segue esse modelo com a mudança apenas do valor do CSS a ser localizado para cada *e-commerce*.

A última etapa e mais importante do *web crawler* é a extração dos dados dos produtos. Cada produto que é extraído do *crawler* tem os seguintes campos: `website`, `categoria`, `marca`, `descrição` e `preço`. Esses campos são tratados e divididos no *pipeline* do *crawler* e são armazenados no banco de dados utilizando o modelo de dados.

```
yield {
    'market' : 'siteaqui.com.br',
    'category': category,
    'brand': brand,
    'description': desc,
    'price': price,
}
```

Código 7 – Modelo dos dados extraídos do *crawler* para cada produto.

4.2.2 Vetorização TF-IDF

A vetorização do TF-IDF segue com os mesmos parâmetro usado na seção 4.1.2, pois já havia sido feito os ajustes necessários para o projeto em andamento.

Uma vez que os resultados com o TF-IDF se mostraram mais promissores que os resultados com o BERT nos experimentos descritos na seção 4.1, preferimos adotar somente aquela como técnica de vetorização das descrições do produtos.

4.2.3 Sugestão de produtos

A sugestão de produtos segue o mesmo critério escolhido na seção 4.1.5 com o uso da similaridade do cosseno entre par de vetores de duas tabelas.

4.2.4 Criação da base de dados

A partir das etapas descritas anteriormente foi feita uma rotulação entre os três membros presentes no projeto, onde a partir das descrições e sua sugestão foi rotulada entre 1 e 0, onde 1

corresponde a uma sugestão válida e 0 corresponde a uma sugestão não válida. Na Figura 13, é mostrada uma tabela com alguns do itens validados.

desc_nota	desc_scraping	cos_aprox	Leonardo	Thiago	Wendel	Validação
fanta pet 2l um	refrigerante uva fanta garrafa 2l	0.366383388778565	1	1	1	1
ref soda ant 2l um	pote de plastico du cheff santana 2l item sortido	0.3523713488565131	0	0	0	0
ref pep cola 350ml um	refrigerante coca-cola lata 350ml	0.37621522089617143	1	1	1	1
agua coc soc m coc un	agua de coco esterilizada sococo caixa 1l	0.3354508192123723	1	1	1	1
oleo liza 900ml um	oleo de soja liza garrafa 900ml	0.6579164643796582	1	1	1	1
vin minh alc 500ml un	vinagre de alcool minhoto frasco 500ml	0.4454455721107843	1	1	1	1
mist bolo 400g um	mistura para bolo chocolate vitamilho pacote 400g	0.4175417385250659	1	1	1	1
l cond moca 395g un	leite condensado moca de colher nestle lata 395g	0.48688471884081025	1	1	1	1
queijo parmesao un	queijo parmesao ralado vigor 100g	0.651523930683923	1	1	1	1

Figura 13 – Validação de forma empírica.

Após a validação foi criada a base de dados de produtos obtidos através do *scraping*, mostrada na Figura 14.

Categoria	Descrição	Preço	E-commerce
chocolates-doces-e-guloseimas	caixa de bombons sortidos garotices garoto 250g	12.99	bompreco
chocolates-doces-e-guloseimas	wafer com recheio e cobertura de chocolate ao leite nestle kitkat 41,5g	1.79	bompreco
chocolates-doces-e-guloseimas	caixa de bombom especialidades nestle 251g	13.99	bompreco
chocolates-doces-e-guloseimas	milho para pipoca premium yoki pacote 500g	5.29	bompreco
chocolates-doces-e-guloseimas	chocolate bis ao leite 126g	6.59	bompreco
chocolates-doces-e-guloseimas	chocolate em barra ao leite alpino nestle 90g	6.59	bompreco
chocolates-doces-e-guloseimas	pipoca para micro-ondas manteiga de cinema yoki pacote 100g	3.19	bompreco
chocolates-doces-e-guloseimas	chocolate snickers 45g	2.59	bompreco
chocolates-doces-e-guloseimas	ovinho de chocolate recheado com oreo lacta 54g	9.99	bompreco
chocolates-doces-e-guloseimas	chocolate em barra ao leite garoto 90g	6.59	bompreco
chocolates-doces-e-guloseimas	wafer com recheio e cobertura de chocolate branco nestle kitkat white 41,5g	1.79	bompreco
chocolates-doces-e-guloseimas	chocolate em barra ao leite classic nestle 90g	6.59	bompreco
chocolates-doces-e-guloseimas	chocolate bis xtra oreo 45g	2.59	bompreco
chocolates-doces-e-guloseimas	chocolate lacta diamante negro 90g	5.99	bompreco

Figura 14 – Dados de produtos extraídos do Bompreço através do *web scraping*.

4.3 Especificação da API

Neste ponto para uma integração de melhor aproveitamento foi sugerido a criação de uma API que é responsável pela execução do *web crawler* que popula a base dados e sugestão de produtos oriundos da base populada a partir da inserção de notas fiscais.

Como todo o projeto foi feito usando Python [3] como linguagem principal, o *framework* escolhido foi o Django[4].

O modelo escolhido para formato da API é mostrado na Fig. 15.

[3] <https://www.python.org/>
[4] <https://www.djangoproject.com/>

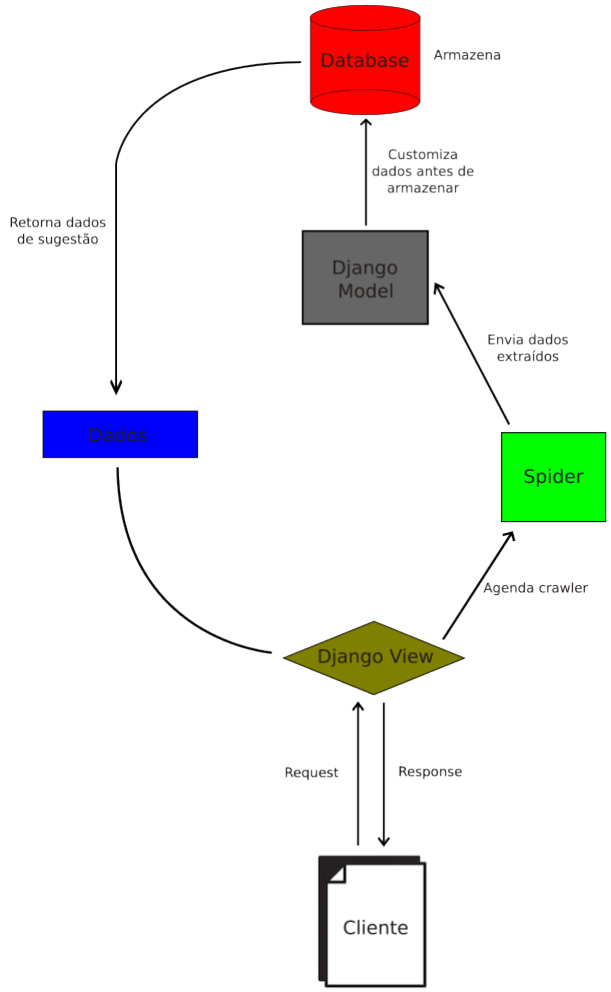

Figura 15 – Modelo da API.

- **Cliente**: O cliente é responsável pela requisição de execução do *crawler* recebendo como parâmetro o CEP e o *e-commerce*, também realiza a requisição para comparação das descrições presente na nota fiscal, que não possuem relação com o EAN de um outro produto já inserido anteriormente, em relação aos descrições extraídos anteriormente dos *crawlers*.

- **Django**: Foi o framework escolhido para criação da API, responsável por realização dos *crawlers* nos *e-commerce* e da sugestão de produtos extraídos previamente do *crawler* para cada produto inserido a partir das notas fiscais que não tenham relação com nenhum código EAN de produto cadastrado previamente no LudiiPrice. O cliente que pode ser o app `LudiiPrice` ou um *script* de início do *crawler*, onde se comunica com a *view* do Django, que possui duas URLs de requisição, correspondentes as duas opções de uso descritas anteriormente.

  A *view* responsável pelo início do *crawler* usa uma biblioteca que realiza agendamentos

```
...
urlpatterns = [
    ...
    path('api/crawl/', views.crawl, name='crawl'),
    path('api/tfidfcomp/',views.tfidfcomp, name='tfidfcomp'),
    ...
]
...
```

Código 8 – URLs permitidas para o uso pelo cliente.

com o auxílio do Scrapyd[5] que é um aplicativo para implantação e execução de *crawler* Scrapy. Ele permite que o usuário implante (carregue) seus projetos e controle seus *spiders* usando uma API JSON. O *framework* foi configurado contendo a integração com Scrapy e com a base de dados.

- **Scrapy**:

  Os *crawlers* usados nessa etapa foram aproveitados da última versão apresentas na seção 4.2.1 com alguns ajustes finos na formatação do texto.

- **Base de Dados**:

  A base de dados foi representada como o DER da Fig. 16

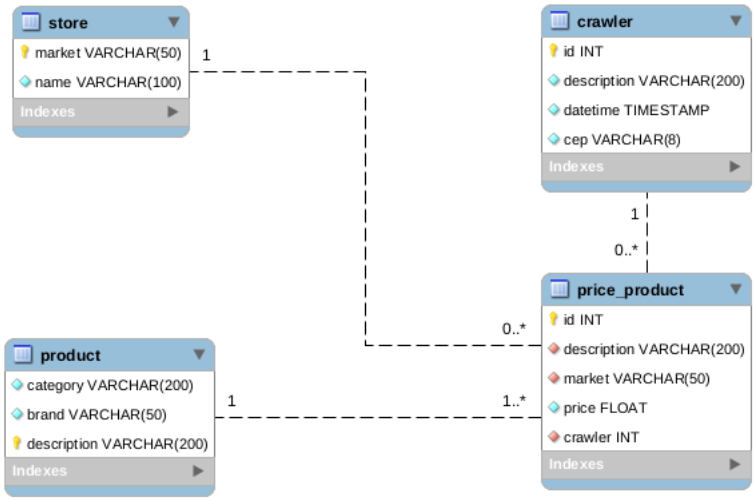

Figura 16 – Modelo da API.

O modelo é o mesmo para os e-commerce escolhidos, onde alguns possuem ou não a marca do produto. ***Product*** corresponde ao produto onde a chave primária é a descrição. ***Price product*** corresponde aos produtos que tenha mais de um preço ou possuam em mais

[5] <https://scrapyd.readthedocs.io/en/stable/>

de um e-commerce. ***Store*** são os *e-commerce* usados nos *crawlers*. ***Crawler*** corresponde as informações do crawler executado, como descrição, data e hora.

5

Resultados Obtidos

Este capítulo apresenta os resultados obtidos. Para fins de clareza de apresentação, o capítulo está seccionado em dois principais blocos semânticos que descrevem os resultados obtidos no experimentos realizados para atribuição do código EAN e de sugestão de produtos, usando TF-IDF e Word Embeddings.

5.1 Experimentos para atribuição do código EAN

Nesta seção será abordada os dois métodos usados para vetorização das descrições dos produtos, onde foram realizados experimentos para obtenção de mais informações sobre os métodos e, assim escolher o que melhor atendia ao problema.

Inicialmente os experimentos consistiam em usar um *e-commerce* como teste para extração de produtos e relacionar com a tabela de dados não oficial mencionada na seção 4.1. O *e-commerce* inicialmente usado foi o Atacadão, onde na Figura 17 é mostrado um exemplo de produto.

Figura 17 – Modelo do produto no Atacadão.

O modelo que o atacadão usa para apresentação dos dados em HTML é mostrado na Figura 18.

```
▶<div class="product-box__img">…</div> flex
  <h2 class="product-box__name">Arroz Parboilizado
  Pacote 1kg - Tio João</h2>
▶<div class="product-box__supplier overflow-hidden
text-nowrap">…</div>
▼<div class="product-box__price"> flex
  ▼<span class="js-product-box__price">
      "R$ "
      <span class="product-box__price--number">4,29
      </span>
      <span class="product-box__price--unit">/pct
      </span>
    </span>
```

Figura 18 – Modelo do HTML de um produto no Atacadão.

Na Figura 19, apresentamos alguns resultados da comparação entre a base dos produtos minerados e tabela que contém os produtos com seus respectivos códigos EAN, as colunas *brand*, *name* e *price* são correspondentes aos produtos extraídos do *e-commerce*; já as colunas *codean* e *old_name* correspondem a tabela que contém os produtos e códigos EAN; por último a coluna *cos_aprox* que corresponde o valor da similaridade do cosseno. Essa abordagem já fazia o uso do TF-IDF, utilizando como parâmetros o analisador por caracteres e *n-gram*.

	codean	brand	categ	name	price	old_name	cos_aprox
0	7891959014612	União	Açúcar Cristal e Demerara	Açúcar Cristal pacote 1kg - União	4.69	Açucar Uniao Cristal 1Kg	0.802922302232567
1	7896894900068	Colombo	Açúcar Cristal e Demerara	Açúcar Cristal pacote 1kg - Colombo	3.63	Açucar Cristal Colombo 1Kg	0.827539005212865
2	7896894900013	Caravelas	Açúcar Cristal e Demerara	Açúcar Cristal Demerara pacote 1kg - Caravelas	8.99	Açúcar Caravelas	0.6635144631966997
3	7896006711124	Camil	Arroz Branco	Arroz tipo 1 pacote 1kg - Camil	4.79	Camil Arroz Tipo 1 2 Kg	0.6641329011246517
4	7896079431110	Namorado	Arroz Branco	Arroz tipo 1 pacote 1kg - Namorado	4.89	Arroz Namorado Tipo 1 1Kg	0.7644921621745111
5	7896038357017	Di Salerno	Arroz Arbóreo e Carnaroli	Arroz arbóreo pacote 1kg - Di Salerno	16.52	Arroz Arborio Urbano 1Kg (Un 1.000)	0.5210739235249915
6	7896213002794	Vitarella	Biscoito Salgado	Biscoito Salgado Cream Cracker pacote 400g - Vitarella	4.99	Biscoito Cream Cracker Pao Assado Vitarella 400G	0.796544683034977
7	7891203056146	Predilleto	Biscoito Salgado	Biscoito Salgado Cream Cracker pacote 400g - Predilleto	3.09	Biscoito Panco Cream Cracker 400G	0.695961372808841
8	7896213000667	Vitarella	Biscoito Salgado	Biscoito Salgado Água e Sal pacote 400g - Vitarella	4.99	Biscoito Vitarella Agua E Sal 400G	0.783136486473181
9	7896014500017	Maratá	Café Torrado e Moído	Café Torrado e Moído Tradicional (em pó) almofada 250g - Maratá	4.99	Café Caneção Almofada 250G Tradicional	0.6891748958404009
10	7891021006125	Melitta	Café Torrado e Moído	Café Torrado e Moído Tradicional (em pó) vácuo 250g - Melitta	7.88	Café Melitta Tradicional Vacuo 500G	0.7106872215449148
11	7896014500017	Mantiqueira	Café Torrado e Moído	Café Torrado e Moído Tradicional (em pó) almofada 250g - Mantiqueira	4.38	Café Caneção Almofada 250G Tradicional	0.6583424199035135
12	7896069510436	Helce	Farinha de Milho e Mandioca	Farinha de mandioca torrada pacote 1kg - Helce	4.1	Farinha De Mandioca Torrada Siamar	0.679751488412735 2
13	7896278600140	Yoki	Farinha de Milho e Mandioca	Farinha de mandioca crua e grossa pacote 1kg - Yoki	6.04	Farinha De Mandioca Crua Grossa 500	0.743645675734907
14	7896039432003	Primeira Linha	Farinha de Milho e Mandioca	Farinha de mandioca crua e grossa pacote 1kg - Primeira Linha	7.5	Farinha De Mandioca Primeira Linha 1Kg	0.7845096225438883
15	7896006744115	Camil	Feijão Carioca	Feijão Carioca pacote 1kg - Camil	7.19	Feijao Carioca Camil 1Kg (Un 1.000)	0.7749748945395255
16	7896006751113	Camil	Feijão Preto	Feijão Preto pacote 1kg - Camil	7.99	Feijao Preto Camil 1 Kg.	0.7045007043139008
17	7896062602008	Solito	Feijão Carioca	Feijão Carioca pacote 1kg - Solito	6.89	Feijao Carioca Solito 1Kg (Un 1.000)	0.7720247838298632
18	7896283800184	Jussara	Leite Integral	Leite Integral garrafa 1Litro - Jussara	4.7	Leite Jussara Max Integral 1L	0.6633815405910304
19	7898080640611	Italac	Leite Integral	Leite Integral tetra pak 1Litro - Italac	4.07	Leite Italac Integral 1L	0.6769809408291787
20	7896185312396	Shefa	Leite Integral	Leite Integral tetra pak 1Litro - Shefa	4.21	Leite Shefa 1Lt Integral	0.6328962145571888
21	7896205788040	Predilleto	Macarrão Espaguete	Macarrão Espaguete pacote 500g - Predilleto	2.16	Macarrão Espaguete	0.6916459605683548
22	7896205788040	Barilla	Massas e Molhos	Macarrão Espaguete pacote 500g - Barilla	8.9	Macarrão Espaguete	0.7246689461819535
23	7896038310197	Urbano	Macarrão Parafuso	Macarrão Parafuso de Arroz pacote 500g - Urbano	2.67	Macarrão C/Ovos Urbano Parafuso 500Gr (Un 1.000)	0.6786656607190317
24	7891080803673	Soya	Óleo de Soja	Óleo de Soja pet 900ml - Soya	8.69	Oleo De Soja Pet 900 Ml Soya	0.9570482051238107
25	7896036090336	Liza	Óleo de Soja	Óleo de Soja pet 900ml - Liza	8.69	Oleo De Soja Lata 900 Ml Liza	0.7938348744181262
26	7896018900035	Leve	Óleo de Soja	Óleo de Soja pet 900ml - Leve	8.59	Oleo Leve Soja 900Ml	0.8054777776740649
27	7896110100043	Lebre	Sal	Sal Refinado pacote 1kg - Lebre	1.58	Sal Refinado Lebre 1Kg (Un 1.000)	0.7752047657797652
28	7896035210001	Cisne	Sal	Sal Refinado pacote 1kg - Cisne	2.38	Sal Refinado Cisne 1Kg (Un 1.000)	0.7715152530199668
29	7896110100029	Norsal	Sal	Sal Refinado pacote 1kg - Norsal	1.39	Sal Refinado Norsal 1Kg (Un 1.000)	0.7621448293819244

Figura 19 – Resultado usando TF-IDF com analisador por caracteres.

Após essas etapas foi inserido mais um *e-commerce*, o Bompreço.

Figura 20 – Modelo do produto no Bompreço.

```
▼<div class="vtex-product-summary-2-x-nameContainer flex items-center
justify-center pv6"> flex
  ▼<h3 class="vtex-product-summary-2-x-productNameContainer mv0 vtex-p
  roduct-summary-2-x-nameWrapper overflow-hidden c-on-base f5">
      <span class="vtex-product-summary-2-x-productBrand vtex-product-s
      ummary-2-x-brandName t-body">Refrigerante Coca-Cola Zero sem
      Açúcar Lata 220ml</span>
    </h3>
  </div>
▼<div class="vtex-productShowCaseContainer">
  ▼<div class="vtex-productShowCaseContent">
      <span class="vtex-productShowCasePrice">R$ 1,99</span>
      <div class="vtex-productShowCasePriceKg"></div>
```

Figura 21 – Modelo do HTML de um produto no Bompreço.

Com o uso do TF-IDF mais a similaridade do cosseno para relacionar as descrições relativamente já ajustado era preciso estabelecer um limiar seguro para atribuição do código EAN. A partir disso foram realizados diversos experimentos para tentar relacionar.

A primeira alternativa após os primeiros resultados foi dividir em intervalos de **0.25** para a similaridade do cosseno. Com isso foram extraídos cerca de 1400 produtos dos dois *e-commerce* para relacionar com a tabela não oficial e detalhar em gráfico nos quatro possíveis intervalos.

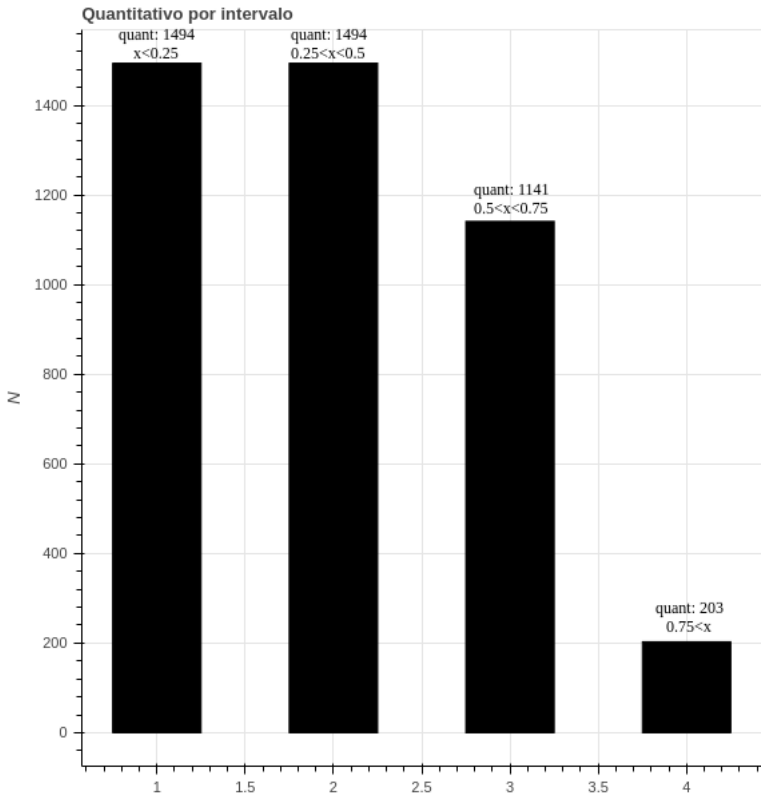

Figura 22 – Quantitativo por intervalos.

O gráfico da Figura 22 mostra o quantitativo nos quatro intervalos que compreende o valor da similaridade do cosseno entre 0 e 1, pois o TF-IDF resulta vetores com valores maiores ou iguais a zero, com normalização do módulo do vetor igual a 1.

A Tabela 5 relaciona o intervalo, quantidade e porcentagem em relação a quantidade máxima da amostragem de produtos.

Intervalo	**Quantidade**	**Porcentagem**
$[0.00 \leq x < 0.25]$	1494	100%
$[0.25 \leq x < 0.50]$	1494	100%
$[0.50 \leq x < 0.75]$	1141	76.3%
$[0.75 \leq x < 1.00]$	203	13.6%

Tabela 5 – Tabela dos dados da Figura 22.

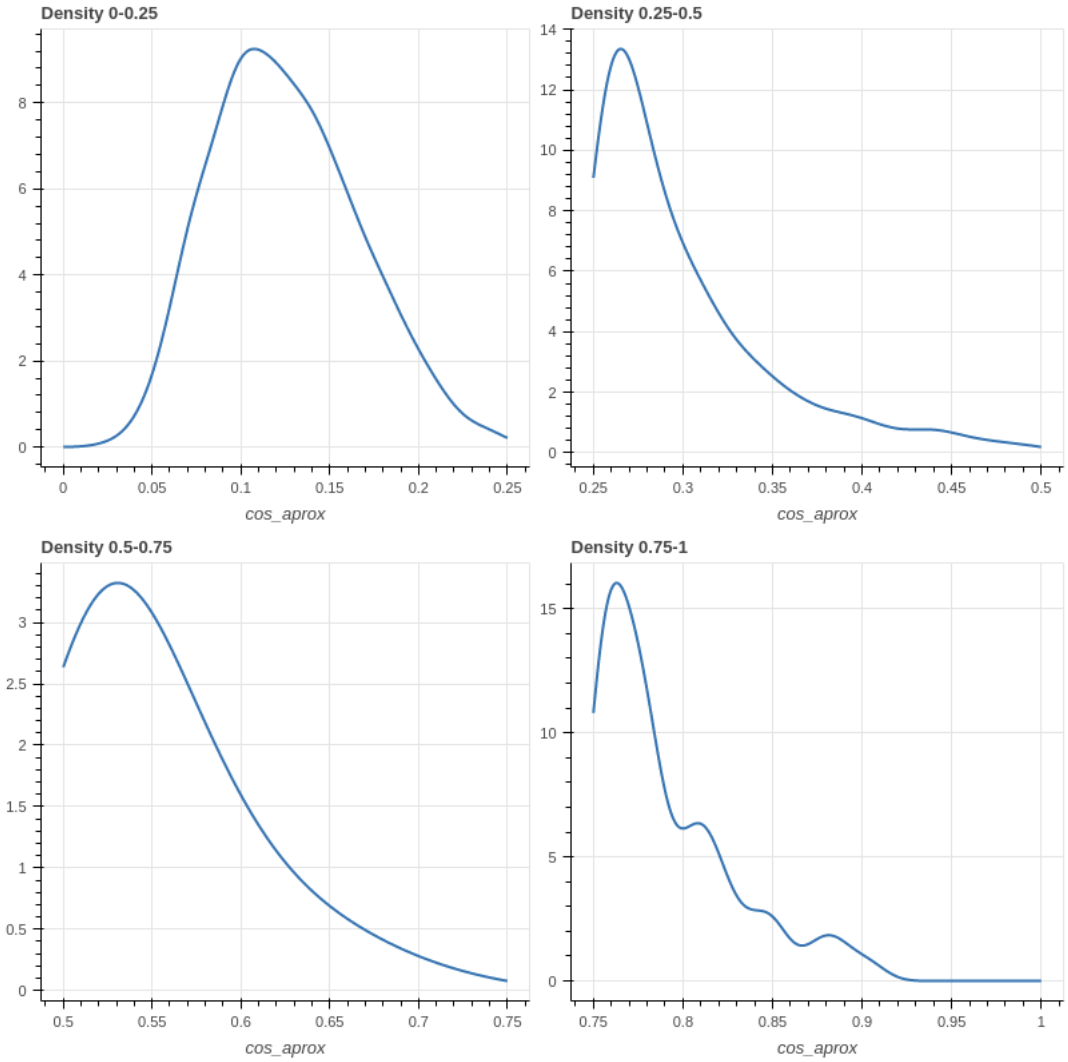

Figura 23 – Gráfico de densidade por intervalo.

Na Figura 23 é mostrado o gráfico de densidade em cada intervalo, onde o eixo x representa o valor da similaridade do cosseno e o eixo y representa a densidade.

Nesse momento foi mudado o foco da abordagem usada para classificação das descrições dos produtos. Foi iniciado o uso dos *word embeddings*, através do modelo BERT mencionado na seção 4.1.4.

	codean	brand	categ	name	price	old_name	cos_aprox
0	7891959014612	União	Açúcar Cristal e Demerara	Açúcar Cristal pacote 1kg - União	4.69	Açucar Uniao Cristal 1Kg	0.9216982
1	7896894900068	Colombo	Açúcar Cristal e Demerara	Açúcar Cristal pacote 1kg - Colombo	3.63	Açucar Cristal Colombo 1Kg	0.92430484
2	7896894900068	Caravelas	Açúcar Cristal e Demerara	Açúcar Cristal Demerara pacote 1kg - Caravelas	8.99	Açucar Cristal Colombo 1Kg	0.8862742
3	7896006711155	Camil	Arroz Branco	Arroz tipo 1 pacote 1kg - Camil	4.79	Arroz Camil Tipo 1 5Kg	0.9138971
4	7896079431110	Namorado	Arroz Branco	Arroz tipo 1 pacote 1kg - Namorado	4.89	Arroz Namorado Tipo 1 1Kg	0.9067005
5	7896018100480	Di Salerno	Arroz Arbóreo e Carnaroli	Arroz arbóreo pacote 1kg - Di Salerno	16.52	Macarrão Todeschini Semola Ovos 500G Ave Maria	0.8812714
6	7896213000653	Vitarella	Biscoito Salgado	Biscoito Salgado Cream Cracker pacote 400g - Vitarella	4.99	Biscoito Crean Cracker Integral 450G Vitarella	0.9526801
7	7896213002817	Predilleto	Biscoito Salgado	Biscoito Salgado Cream Cracker pacote 400g - Predilleto	3.09	Biscoito Cream Cracker Crocks Queijo Vitarella 400G	0.92245406
8	7896213000667	Vitarella	Biscoito Salgado	Biscoito Salgado Água e Sal pacote 400g - Vitarella	4.99	Biscoito Vitarella Agua E Sal 400G	0.9476436
9	7896009630033	Maratá	Café Torrado e Moído	Café Torrado e Moído Tradicional (em pó) almofada 250g - Maratá	4.99	Motta-Café Torrado E Moido 500G	0.8675552
10	7891021006309	Melitta	Café Torrado e Moído	Café Torrado e Moído Tradicional (em pó) vácuo 250g - Melitta	7.88	Café Melitta 250G Descafeinado Classico	0.87403965
11	7896009630033	Mantiqueira	Café Torrado e Moído	Café Torrado e Moído Tradicional (em pó) almofada 250g - Mantiqueira	4.38	Motta-Café Torrado E Moido 500G	0.8433973
12	7896046604554	Helce	Farinha de Milho e Mandioca	Farinha de mandioca torrada pacote 1kg - Helce	4.1	Farinha De Mandioca Crua Chinezinho 1Kg	0.9124154
13	7896046604554	Yoki	Farinha de Milho e Mandioca	Farinha de mandioca crua e grossa pacote 1kg - Yoki	6.04	Farinha De Mandioca Crua Chinezinho 1Kg	0.92387354
14	7896039432003	Primeira Linha	Farinha de Milho e Mandioca	Farinha de mandioca crua e grossa pacote 1kg - Primeira Linha	7.5	Farinha De Mandioca Primeira Linha 1Kg	0.9120572
15	7896062602053	Camil	Feijão Carioca	Feijão Carioca pacote 1kg - Camil	7.19	Feijao Carioca Vitabon 1Kg	0.8838774
16	7896006751113	Camil	Feijão Preto	Feijão Preto pacote 1kg - Camil	7.99	Feijao Preto Camil 1 Kg.	0.9136611
17	7896006746157	Solito	Feijão Carioca	Feijão Carioca pacote 1kg - Solito	6.89	Feijao Carioca Tipo 1 1Kg	0.9116212
18	7896185310019	Jussara	Leite Integral	Leite Integral garrafa 1Litro - Jussara	4.7	Leite Shefa De Saquinho 1 Litro	0.86782753
19	7896034611960	Italac	Leite Integral	Leite Integral tetra pak 1Litro - Italac	4.07	Leite Uht Integral Classic 1 Litro Parmalat	0.8653096
20	7896185312396	Shefa	Leite Integral	Leite Integral tetra pak 1Litro - Shefa	4.21	Leite Shefa 1Lt Integral	0.90627545
21	7896024740144	Predilleto	Macarrão Espaguete	Macarrão Espaguete pacote 500g - Predilleto	2.16	Macarrão Piraque Espaguete 500G	0.924938
22	7896323613026	Barilla	Massas e Molhos	Macarrão Espaguete pacote 500g - Barilla	8.9	Macarrão Pin Semolado Espaguete 500G	0.9236191
23	7896323604109	Urbano	Macarrão Parafuso	Macarrão Parafuso de Arroz pacote 500g - Urbano	2.67	Macarrão Pin Semolado Parafuso Verde 1Kg	0.90889263
24	7891080803673	Soya	Óleo de Soja	Óleo de Soja pet 900ml - Soya	8.69	Oleo De Soja Pet 900 Ml Soya	0.990657
25	7891080803673	Liza	Óleo de Soja	Óleo de Soja pet 900ml - Liza	8.69	Oleo De Soja Pet 900 Ml Soya	0.9618473

Figura 24 – Tabela do resultado do uso de Word Embeddings com Bert.

Como visto na Figura 24 a variação do cosseno (*cos_aprox*) entre as descrições é baixa em relação ao intervalo total que fica entre 0 e 1, ou seja, ficaria difícil de verificar produtos classificados corretamente ou não.

Com o uso do BERT e o TF-IDF foi feita uma correlação de Pearson, onde no eixo Y representa os Word Embeddings e no eixo X o TF-IDF. Cada produto foi escolhido aleatoriamente dentre as categorias do Atacadão e foram divididos em quatro classes.

- **Classe 1** - Produtos iguais, marcas iguais;
- **Classe 2** - Produtos iguais, marcas diferentes;
- **Classe 3** - Produtos diferentes, marcas iguais;
- **Classe 4** - Produtos diferentes, marcas diferentes;

As classes são representadas pelas seguintes cores: Classe 1 - verde; Classe 2 - azul; Classe 3 - vermelho; Classe 4 - preto.

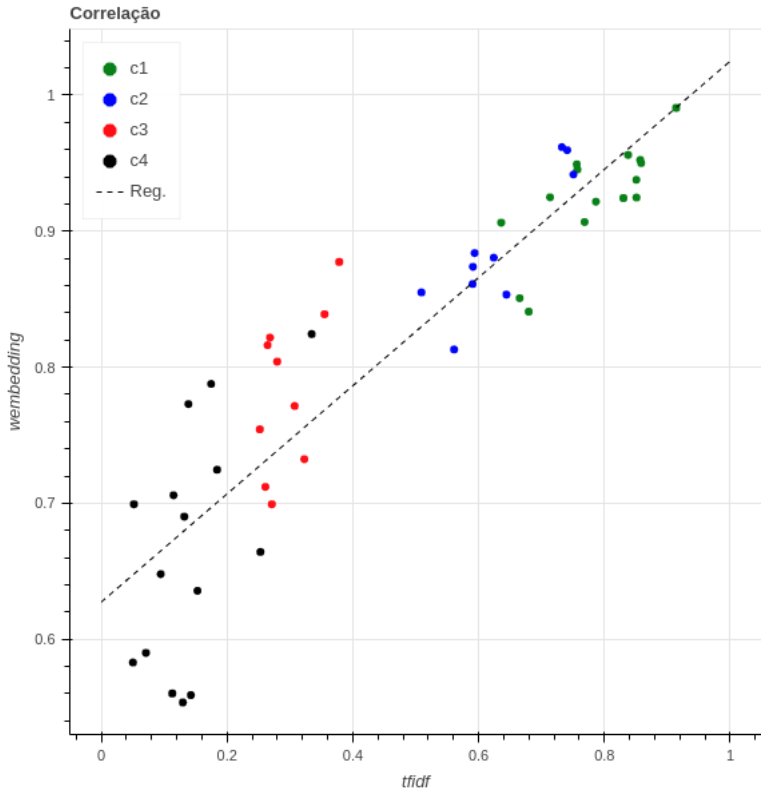

Figura 25 – Correlação de Pearson entre Word Embeddings e TF-IDF.

Com o resultado desse gráfico foi visto uma grande proximidade entre pontos pertencentes as mesmas classes. Foi definido 4 intervalos, onde cada um representaria uma classe. Cada classe possui um intervalo de 0,25 para similaridade do cosseno:

- **Classe 1** - $0.75 \leq x < 1.00$
- **Classe 2** - $0.50 \leq x < 0.75$
- **Classe 3** - $0.25 \leq x < 0.50$
- **Classe 4** - $0.00 \leq x < 0.25$

Com resultados não muito satisfatórios entre as classes 2, 3 e 4 foram realizadas mudanças nas classes, onde apenas duas classes seriam abordadas.

- **Classe 1** - Produtos iguais, marcas iguais;
- **Classe 2** - Junção das Classes 2, 3 e 4 da versão anterior;

Com o resultado da Figura 25 pareceu satisfazer, em um certo nível, os intervalos e seus respectivos significados, as abordagens seguiram esse modelo de divisão de intervalos.

As Figuras 26 e 27 correspondem ao resultado de uma amostra de aproximadamente 1200 produtos extraídos do Atacadão e Bompreço com o uso dos intervalos propostos e mencionados na seção 4.1.6.

Na Figura 26 é mostrado um gráfico muito parecido com o da Figura 22 onde agora possui divisões por classificações. A cor verde representa a **Classe 1**, a cor azul representa a **Classe 2**, a cor vermelha representa a **Classe 3** e a cor preta representa a **Classe 4**.

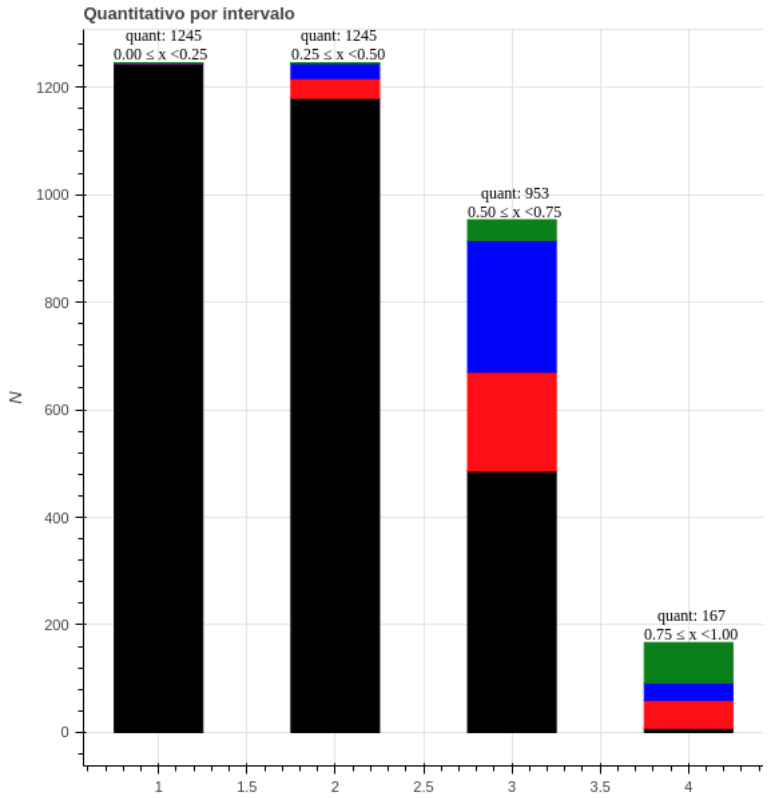

Figura 26 – Quantitativo por intervalos com classificação.

A Tabela 6 relaciona o intervalo, quantidade, porcentagem das classificações e porcentagem total em relação a quantidade máxima de produto. Os valores das porcentagens das classificações foram arrendondados com uma casa decimal.

Intervalo	**Quantidade**	**Porcentagem**	**C1**	**C2**	**C3**	**C4**
$[0.00 \leq x < 0.25]$	1245	100%	0%	0.08%	0.08%	99.8%
$[0.25 \leq x < 0.50]$	1245	100%	0.08%	2.2%	3%	94.7%
$[0.50 \leq x < 0.75]$	953	76.5%	4%	25.7%	19.3%	51%
$[0.75 \leq x < 1.00]$	167	13.4%	45%	19.2%	31.1%	4.8%

Tabela 6 – Tabela dos dados da Figura 26.

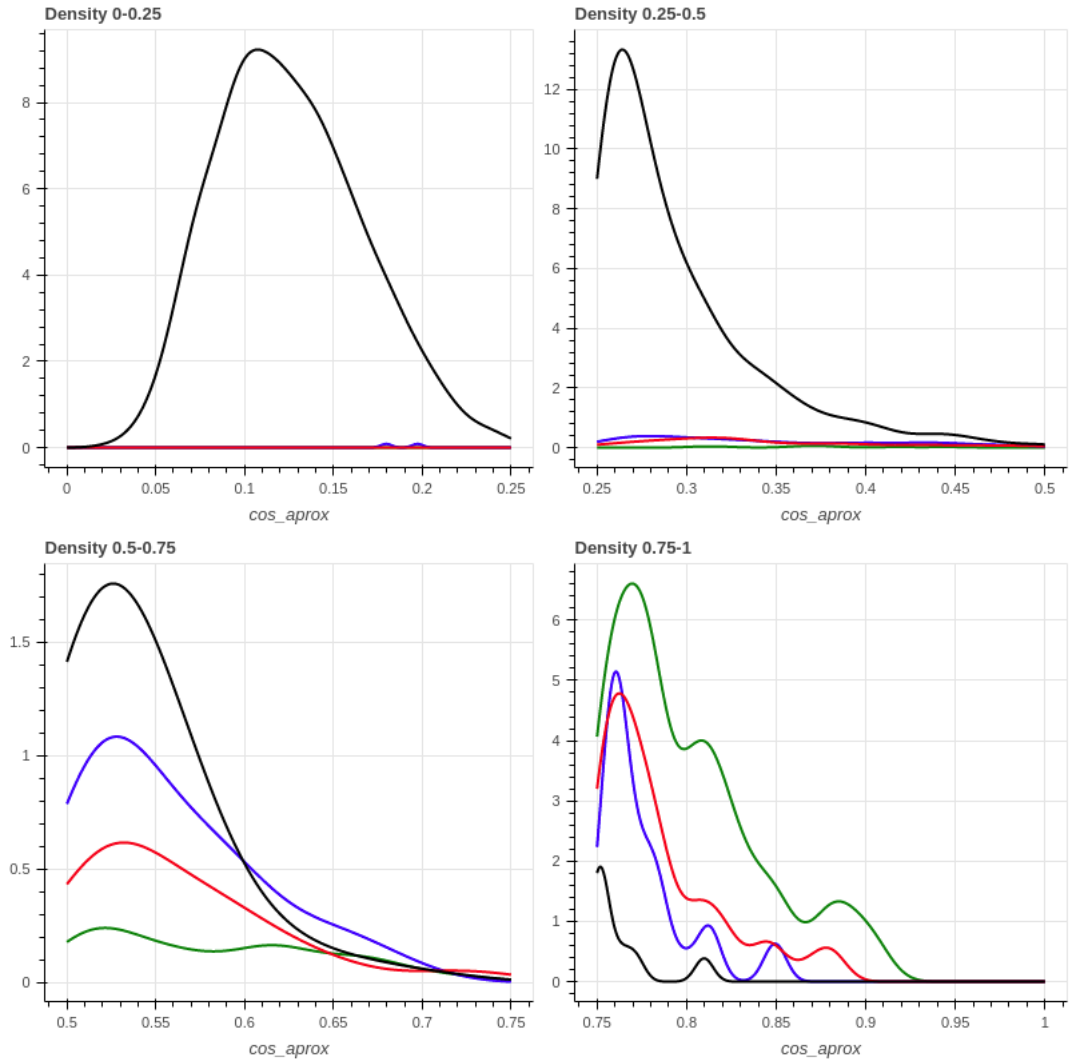

Figura 27 – Gráfico de densidade por intervalo com classificação.

Como é visto na Tabela 6 e na Figura 27 cada intervalo tem uma classificação que se destaca em relação as outras, mas essas porcentagens não são altas o bastante para atribuir um produto a um código EAN. Outra maneira de visualizar isso são produtos que foram classificados com **Classe 2** e **3** com um valor de aproximação de cosseno alta o bastante para ter uma grande influência na Figura 27 no intervalo de $0.75 \leq x < 1.00$, onde deveria ter um influência muito alta por parte da *Classe 1*.

Diante dos experimentos realizados e da dificuldade de estabelecer um limiar preciso para atribuição do código EAN foi decidido apenas sugerir produtos a partir da classificação usando o TF-IDF e similaridade do cosseno visto na seção 5.2.

5.2 Experimentos para sugestão de produtos

Como visto na seção 5.1, não foi possível estabelecer um limiar para a atribuição do código EAN. Com isso o método escolhido para essa nova etapa de experimentos foi sugerir os produtos. Foi mantido o TF-IDF como técnica de vetorização das descrições e a similaridade do cosseno para classificação das descrições.

A base de dados comprada deixou de ser usada e foi priorizada usar a base de notas fiscais do LudiiPrice para montar o vocabulário usando o TF-IDF e relacionar com a base de dados extraída do *crawler*. Nessa etapa foi adicionado mais um *e-commerce*, o Mercantil Rodrigues[1].

Figura 28 – Modelo do produto no Mercantil Rodrigues.

```
▼<div _ngcontent-dqa-c25 class="info-price">
    " R$ 1,75 "
    <span _ngcontent-dqa-c25 class="info-price"> </span>
  </div>
 ▶<div _ngcontent-dqa-c25 class="arrow-right">…</div>
 </div>
 <!---->
 <!---->
</app-tag-preco>
▶<a _ngcontent-dqa-c18 class="ghost-link clearfix" href="/produtos/detalhe/7716/banana-dagua-500g-aprox-6-unid">…</a>
▶<div _ngcontent-dqa-c18 class="clearfix">…</div>
▼<div _ngcontent-dqa-c18 class="caption">
 ▼<p _ngcontent-dqa-c18 class="text-success description center-block text-center hidden-xs">
    <a _ngcontent-dqa-c18 title="Banana D`água 500g (aprox. 6 Unid)" href="/produtos/detalhe/7716/banana-dagua-500g-aprox-6-unid">Banana D`água 500g (aprox. 6 Unid)</a>
```

Figura 29 – Modelo do HTML de um produto no Mercantil Rodrigues.

A extração dos produtos em cada *e-commerce* foi feita a partir das categorias disponibilizadas.

[1] <https://delivery.mercantilrodrigues.com/>

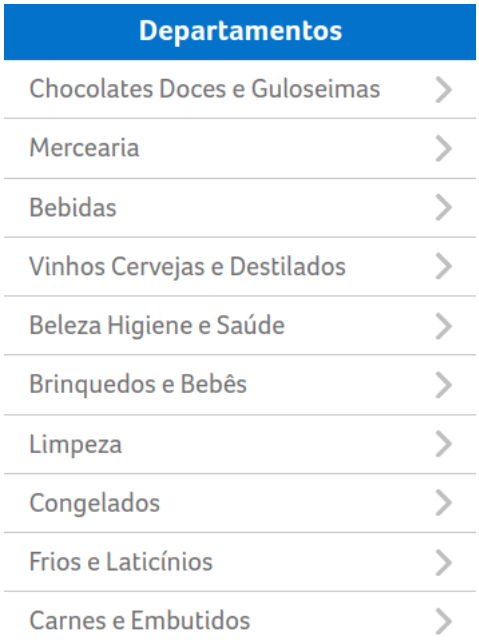

Figura 30 – Exemplo de categorias do Bompreço.

Com esses dados foi feita a API mencionada na seção 4.3 que contém dois tipos de requisições, sendo um deles para realização do crawler, com os dados para especificar o *e-commerce* e o CEP, tendo como retorno o agendamento do *crawler*.

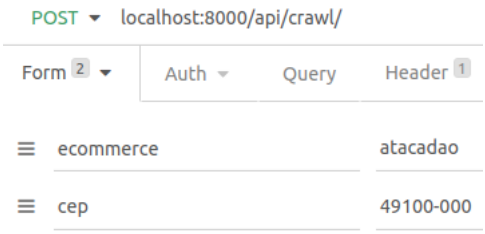

Figura 31 – Modelo de requisição para execução do *crawler* no Atacadão.

A outra requisição é justamente para sugestão de produtos, onde na requisição é inserido a descrição do produto e o retorno possui a descrição, preço e *e-commerce* com maior valor da aproximação do cosseno. Nas Figuras 32 e 33 tanto a requisição quanto o retorno são feitos no formato JSON.

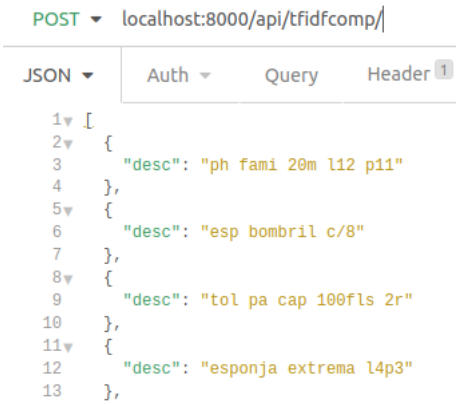

Figura 32 – Modelo dos dados da requisição para sugestão.


```
{
    "desc_esq": "ph fami 20m l12 p11",
    "desc_dir": "Esc Dental Colgate Whitening Macia L2 P1 L2 P1",
    "price": 18.29,
    "cos_aprox": 0.3127576622,
    "market": "delivery.mercantilrodrigues.com.br",
    "cep": "49095-000"
},
{
    "desc_esq": "esp bombril c8",
    "desc_dir": "La De Aco Bombril Com 8 Unidades",
    "price": 1.69,
    "cos_aprox": 0.5029577767,
    "market": "delivery.mercantilrodrigues.com.br",
    "cep": "49095-000"
},
{
    "desc_esq": "tol pa cap 100fls 2r",
    "desc_dir": "Polpa Congelada de Caja Pomar Pacote 100g",
    "price": 2.19,
    "cos_aprox": 0.2886356905,
    "market": "bompreco.com.br",
    "cep": "49100-000"
},
{
    "desc_esq": "esponja extrema l4p3",
    "desc_dir": "esponja de limpeza unidade scotch briteextrema",
    "price": 1.79,
    "cos_aprox": 0.4678089614,
    "market": "atacadao.com.br",
    "cep": "49100-000"
}
```


Figura 33 – Modelo do resultado da requisição de sugestão.

Esta seção de resultados foi uma alternativa visto a impossibilidade de atribuição do EAN citado na seção 4.1.

A validação desses resultados foram realizados com o preenchimento de uma tabela de aceitação pelos participantes deste trabalho, onde a partir de notas fiscais foi realizada sugestões para os produtos contidos nas notas. O preenchimento da tabela era dividido entre a sugestão ser coerente com o produto da nota fiscal ou não. Essa validação foi feita levando em conta apenas o produto, onde a marca e unidade não eram relevantes.

Na Tabela 7 é mostrada a acurácia da validação feita.

Quantidade produtos	**Rotulação**	**Porcentagem**
65	Sugerido	80.2%
16	Não sugerido	19.8%

Tabela 7 – Tabela de validação da sugestão dos produtos

As porcentagens indicação uma grande quantidade de produtos com sugestão válida, mesmo tendo uma quantidade pequena de produtos na base, visto que até o momento é composta por três *e-commerce*, onde não supre todos as possíveis categorias de produtos.

6
Conclusão

Neste trabalho foi proposto a criação de uma base de dados que contém informações como o código EAN, descrição e preço de produtos, em extensão ao trabalho ainda em andamento do LudiiPrice. Nos capítulos anteriores foi compreendido o uso de técnicas de comparação de *strings*, que é de suma importância para classificação de cada produto, modelos de extração de dados estruturados usando *web crawler* e a aplicação dos modelos de *web crawler* em *e-commerce* para obtenção de informações. Foi mencionado o uso das ferramentas da Bluesoft Cosmos e da GS1 Brasil como um modelo de organização de dados de produtos e um exemplo de *web crawler* para extração de dados de *e-commerce*. Foram especificados os métodos usados nos experimentos para atribuição do código EAN e sugestão de produtos a partir das notas fiscais inseridas através do QRCode no aplicativo LudiiPrice; criação da API para requisição do *crawler* e sugestão. Por fim, relatamos os resultados dos experimentos e modelo das requisições. Os resultados demonstraram que o método de comparação e sugestão de produtos é eficiente e viável para ser integrado do LudiiPrice, visto que é uma forma de entregar sugestões como recompensa ao cadastrar novos produtos; um ponto negativo encontrado foi a pequena variação da similaridade do cosseno devido as abreviações dos produtos encontrados nas notas fiscais. Entretanto, novos experimentos com uma maior quantidade de notas fiscais faz-se necessário para garantir mais robustez aos resultados e o possível complemento das descrições abreviadas para um melhor aproveitamento dos valores da similaridade do cosseno.

Referências

AIZAWA, A. An information-theoretic perspective of tf–idf measures. *Information Processing & Management*, v. 39, n. 1, p. 45–65, 2003. Citado na página 20.

AVRAAM, I.; ANAGNOSTOPOULOS, I. A comparison over focused web crawling strategies. *2011 15th Panhellenic Conference on Informatics. IEEE*, p. 245–249, 2011. Citado na página 18.

BRABHAM, C. D. *Crowdsourcing*. [S.l.]: Mit Press, 2013. Citado na página 12.

CAVNAR, B. W. N-gram-based text categorization. *Proceedings of SDAIR-94, 3rd annual symposium on document analysis and information retrieval*, 1994. Citado na página 22.

CHO, J.; GARCIA-MOLINA, H. The evolution of the web and implications for an incremental crawler. *Stanford*, 1999. Citado na página 18.

CHOWDHURY, G. G. Natural language processing. *Annual review of information science and technology*, v. 37, n. 1, p. 51–89, 2003. Citado na página 19.

DEVLIN, J. et al. Bert: Pre-training of deep bidirectional transformers for language understanding. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Association for Computational Linguistics, Minneapolis, Minnesota, p. 4171–4186, 2019. Disponível em: <https://aclanthology.org/N19-1423>. Citado na página 20.

DHENAKARAN, S.; SAMBANTHAN, K. T. Web crawler - an overview. *International Journal of Computer Science and Communication*, v. 2, n. 1, p. 265–267, 2011. Citado na página 17.

HIRAI, J. et al. Webbase: A repository of web pages. *Computer Networks*, v. 33, n. 1-6, p. 277–293, 2011. Citado na página 18.

HUANG, A. Similarity measures for text document clustering. *Proceedings of the sixth new zealand computer science research student conference*, p. 9–56, 2008. Citado na página 22.

INDURKHYA, N.; DAMERAU, J. F. *Handbook of natural language processing*. [S.l.]: CRC Press, 2010. Citado na página 19.

JURAFSKY, D.; MARTIN, H. J. Speech and language processing (draft). 2018. Disponível em: <https://web.stanford.edu/~jurafsky/slp3>. Citado 3 vezes nas páginas 20, 21 e 23.

KAUSAR, M. A.; DHAKA, V. S.; SINGH, S. K. Web crawler: a review. *International Journal of Computer Applications*, v. 63, n. 2, 2013. Citado na página 17.

LEBRET, R.; COLLOBERT, R. Word emdeddings through hellinger pca. arXiv, 2013. Disponível em: <https://arxiv.org/abs/1312.5542>. Citado na página 20.

LEVENE, M.; POULOVASSILIS, A. *Web dynamics: Adapting to change in content, size, topology and use*. [S.l.]: Springer Science & Business Media, 2004. Citado na página 18.

LEVENSHTEIN, I. V. Binary codes capable of correcting deletions, insertions and reversals. *Soviet physics doklady*, p. 707–710, 1966. Citado na página 23.

LI, B.; HAN, L. Distance weighted cosine similarity measure for text classification. *International conference on intelligent data engineering and automated learning*, p. 611–618, 2013. Citado 2 vezes nas páginas 22 e 24.

ONYENWE, I. et al. Developing products update-alert system for e-commerce websites users using html data and web scraping technique. *arXiv preprint arXiv:2109.00656*, 2021. Citado 2 vezes nas páginas 5 e 27.

RAMOS, J. Using tf-idf to determine word relevance in document queries. *Proceedings of the first instructional conference on machine learning*, p. 29–48, 2003. Citado na página 20.

SALTON, G.; BUCKLEY, C. Term-weighting approaches in automatic text retrieval. *Information processing & management*, v. 24, n. 5, p. 513–523, 2000. Citado na página 20.

SHKAPENYUK, V.; SUEL, T. Desing and implmentation of a high-performance distributed web crawler. *Proceedings 18th International Conference on Data Engineering. IEEE*, p. 1–2, 2002. Citado na página 17.

SOUZA, F.; NOGUEIRA, R.; LOTUFO, R. Bertimbau: Pretrained bert models for brazilian portuguese. In: CERRI, R.; PRATI, C. R. (Ed.). *Intelligent Systems*. Cham: Springer International Publishing, 2020. p. 403–417. Citado na página 21.

UDAPURE, T.; KALE, D. R.; DHARMIK, C. R. Study of web crawler and its different types. *IOSR Journal of Computer Engineering*, v. 16, n. 1, p. 01–05, 2014. Citado na página 18.

VIIKMAA, A. *Web Data Extraction for Content Aggregation from E-Commerce Websites*. Tese (Doutorado) — Dissertação (Mestrado)—University of Tartu, 2016. Citado na página 25.

WAGNER, R.; FISCHER, M. The string-to-string correction problem. *Journal of the ACM (JACM)*, v. 21, n. 1, p. 168–173, 1974. Citado na página 23.