UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

MAX HUMBERTO RECUERO JUNIOR

UMA BASE TEXTUAL EM PORTUGUÊS DESTINADA À PESQUISAS DE PROCESSAMENTO DE LINGUAGEM NATURAL APLICADOS À ENGENHARIA DO SOFTWARE

PATO BRANCO

2023

MAX HUMBERTO RECUERO JUNIOR

UMA BASE TEXTUAL EM PORTUGUÊS DESTINADA À PESQUISAS DE PROCESSAMENTO DE LINGUAGEM NATURAL APLICADOS À ENGENHARIA DO SOFTWARE

A Textual Database on Portuguese Language to Natural Language Processing Aplications on Software Engineering

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Bacharel em Engenharia da Computação do Curso de Bacharelado em Engenharia da Computação da Universidade Tecnológica Federal do Paraná.

Orientador: Profª. Drª. Eliane Maria de Bortoli Fávero

PATO BRANCO

2023

MAX HUMBERTO RECUERO JUNIOR

UMA BASE TEXTUAL EM PORTUGUÊS DESTINADA À PESQUISAS DE PROCESSAMENTO DE LINGUAGEM NATURAL APLICADOS À ENGENHARIA DO SOFTWARE

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Bacharel em Engenharia da Computação do Curso de Bacharelado em Engenharia da Computação da Universidade Tecnológica Federal do Paraná.

Data de aprovação: 21/Junho/2023

Eliane Maria de Bortoli Fávero
Doutora
Universidade Tecnológica Federal do Paraná

Dalcimar Casanova
Doutor
Universidade Tecnológica Federal do Paraná

Rúbia Eliza de Oliveira Schultz Ascari
Doutora
Universidade Tecnológica Federal do Paraná

PATO BRANCO

2023

RESUMO

Aplicações de Processamento de Linguagem Natural (PLN), em sua maioria, requisitam grandes volumes de dados. Isso é necessário para que que seja possível gerar uma aplicação consistente, o que normalmente é obtido por meio de métodos de aprendizado de máquina. Grande parte das pesquisas nessa área apresentam bases de dados específicas e elaboradas pelos próprios autores, o que dificilmente atinge o volume necessário para que aplicações de aprendizagem de máquina obtenham um resultado satisfatório. Visando disponibilizar uma base de dados textuais em Português brasileiro, para a área de Engenharia de Software, este trabalho apresenta o processo de obtenção automática e tratamento de dados textuais, extraídos de perguntas e respostas contidas no portal *Stack Overflow*, bastante popular entre os membros da área. Sendo assim, esse trabalho aborda conceitos de *web crawler* e *web scrapper*, ferramentas utilizadas para extrair dados da Internet, e também métodos de pré-processamento dos textos extraídos, aplicando técnicas de PLN. O pré-processamento se faz importante, pois esse tipo de dado possui diversas características de linguagem HTML e uma grande variabilidade de dados considerados estranhos, porém busca-se manter a originalidade e coêrencia das sentenças, para que futuras pesquisas possam utilizar-se dessa base em diversas tarefas de PLN de forma consistente. Os resultados apresentam a abordagem utilizada, características e dificuldades encontradas. Por fim, é apresentada a base de dados gerada com destaque para suas principais características.

Palavras-chave: bases textuais; prprocessamentos de texto; web crawling; web scraping.

ABSTRACT

Applications of Natural Language Processing (NLP) mostly require large volumes of data. This is necessary in order to develop a robust application, typically based on machine learning algorithms. Much of the research in this field relies on specific and curated databases created by the authors themselves, which rarely reach the volume required by machine learning aplications. With the aim of providing a textual database in Brazilian Portuguese for the Software Engineering field, this work presents the process of automatic data retrieval and processing, extracted from questions and answers on the popular portal *Stack Overflow*, widely used by members of the field. Therefore, this work encompasses concepts of web crawling and web scraping, tools used to extract data from the internet, as well as preprocessing methods for the extracted texts using NLP techniques. Preprocessing is important because this type of data often contains HTML language characteristics and a wide range of irregular data, but the goal is to maintain the originality and coherence of the sentences, allowing future research to consistently utilize this database for various NLP tasks. The results present the approach used, its characteristics, and the challenges encountered. Finally, the generated database is presented, highlighting its main features.

Keywords: textual database; text pre-processing; web crawling; web scraping.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

Siglas

API	Interface de Programação de Aplicação, do Inglês *Application Programming Interface*
CSV	Valores separados por vírgula, do inglês *Comma Separated Values*
DOM	*Document Object Model*
EAQ	*Educational Administration Quarterly*
HTML	Linguagem de marcação de hipertexto, do Inglês *HyperText Markup Language*
HTTP	*Hypertext Transfer Protocol*
PLN	Processamento de Linguagem Natural
Regex	Expressoes Regulares, do inglês *Regular Expression*
SAX	*Simple API for XML*
SQL	Structured Query Language
t-SNE	*t-Distributed Stochastic Neighbor Embedding*
XML	*Extensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO

O Processamento de Linguagem Natural (PLN), é uma subárea da inteligência artificial que se concentra no estudo da interação humana com computadores usando a linguagem natural (ex. voz e texto). O objetivo principal é desenvolver tecnologias capazes de compreender e responder a linguagem humana de maneira eficiente e natural (MANNING; RAGHAVAN; SCHÜTZE, 2014). Algumas das aplicações mais comuns de PLN incluem análise de sentimento, tradução automática, geração de texto, entre outros. Atualmente a pesquisa em PLN está sendo intensificada devido ao crescente interesse no uso de sistemas de linguagem natural para aprimorar a interação humano-computador e para aplicações em áreas como saúde, segurança e negócios (JOU; CHANG; LIN, 2019).

O principal insumo necessário para a realização de pesquisas em PLN é uma base de dados textual, volumosa o suficiente para permitir a generalização de um modelo inteligente, treinado a partir dos conhecimentos contidos nessa base. Porém, validar e avaliar métodos de PLN ainda são tarefas difíceis, principalmente devido à falta de bases de dados estabelecidas para referência, testes de performance e validações. Esse problema se agrava quando há a necessidade de uma base de dados específica de um domínio (ex. Engenharia de Software, Agronomia, entre outros), pois estudos específicos estabelecem suas próprias referências e conduzem testes em bases de dados específicas ao problema (KE; LIU, 2022).

A Internet disponibiliza uma diversidade de informações, em diversos formatos, tais como imagens, áudios e textos, que podem ser encontrados em diversas fontes (ex. *sites* de jornalismo, fóruns, *e-commerces*, repositórios de colaboração em núvem (ex. GitHub), e vários outros. Assim, cada *website* organiza as informações de acordo com o seu objetivo e público-alvo. Um exemplo é a plataforma *Stackoverflow*, que atua como um fórum de perguntas e respostas, contendo o nome de quem perguntou, o título da publicação, a pergunta, as respostas, as categorias ou assuntos da publicação e métricas de avaliação (ex. *"likes"*) para escolher a melhor resposta (HIN, 2020).

Ao se analisar a quantidade de informações disponíveis na web, nas mais diversas áreas de atuação, percebe-se a possibilidade de coletar e estruturar essas informações, a fim de criar bases de dados textuais, objetivando atender às pesquisas em PLN de domínios específicos. Porém, é necessário um método adequado de extração desses dados a partir de sua fonte, seguido da aplicação de métodos de pré-processamento, a fim de disponibilizar apenas as informações relevantes ou pertinentes à área de estudo, sem perda de contexto.

A área de engenharia de software é bastante carente de dados textuais para pesquisas envolvendo PLN, especialmente de dados na língua portuguesa. Considerando esse contexto, esse trabalho propõe o seguinte problema de pesquisa: **Como criar uma base de dados textual na língua portuguesa, destinada às pesquisas de PLN aplicadas à área de engenharia de software?**

Para isso, foi implementar uma ferramenta de busca, extração e tratamento de textos em *websites* específicos, os quais contenham tópicos relacionados à engenharia de software e seus processos. Desta forma, esse trabalho explorou o portal *Stackoverflow* [1], o qual tem sido muito popular entre os desenvolvedores e estudiosos da área, além de facilitar a coleta de dados. Por esse motivo, é considerado uma fonte importante de dados da área, possibilitando até mesmo a rotulação desses dados textuais de acordo com algumas categorias (ex. tecnologia aplicada, tipo do texto, entre outros).

Algumas técnicas para extração desses dados diretamente da web e de forma automatizada já têm sido propostas. Dentre elas, estão os sistemas de *web crawling* e *web scraping*, os quais foram explorados durante o desenvolvimento desse trabalho, a fim de identificar qual a configuração mais indicada. Com isso, espera-se contribuir com pesquisas na área de engenharia de software aplicando métodos de PLN. Um exemplo de pesquisa que seria beneficiada, seria a estimativa de esforço de software com base em analogia, fazendo uso de textos de requisitos de software (FÁVERO; CASANOVA; PIMENTEL, 2022).

1.1 Objetivos

1.1.1 Objetivo Geral

- Criar uma base de dados textual na língua portuguesa, destinada à pesquisas de PLN aplicadas à engenharia de software.

1.1.2 Objetivos Específicos

- Desenvolver um *Web Crawler* e um *Web Scrapper* para extrair dados de perguntas e respostas do StackOverflow;
- Realizar pré-processamento básico nos textos extraídos;
- Gerar a base de dados para futuras pesquisas na área de engenharia de software;
- Descrever os dados extraídos e contidos na base.

1.2 Justificativa

Atividades que envolvem a área de PLN requerem bases de dados muito grandes. Pode-se exemplificar isso nos trabalhos do Escritório de Pesquisas Navais, que conduziu uma pesquisa piloto no assunto de mineração de dados textuais, utilizando-se de dados da década de 90

[1] www.stackoverflow.com

(1991 até 1999), com objetivo de analisar repetições de frases, levantar requisitos e características de materiais, oportunidades, longevidade e manutenção (LOSIEWICZ; OARD; KOSTOFF, 2000). Outro exemplo é feito pela *Educational Administration Quarterly* (EAQ), com artigos publicados de 1965 à 2014, contemplando um total de 1539 artigos, com o objetivo de analisar tópicos recorrentes dos mesmos (WANG; BOWERS; FIKIS, 2017). Mais especificamente na área de Engenharia de Software, a única base de dados disponível está na língua inglesa, e possui 23.313 textos de requisitos de software (CHOETKIERTIKUL *et al.*, 2019).

Os exemplos apresentados necessitaram de dados que foram coletados ao longo de anos para formar uma base estruturada e possível de ser aplicada em um estudo científico. A base de dados disponibilizada por Choetkiertikul *et al.* (2019) é fruto da coleta de requisitos em 16 projetos de código aberto, o que também onerou bastante tempo. Sendo assim, este trabalho objetiva disponibilizar uma base de dados consistente no contexto da engenharia de software, para ser usada em pesquisas da área.

1.3 Estrutura do Trabalho

Nos próximos capítulos, são apresentados os principais conceitos necessários à compreensão da necessidade estruturar uma base de dados, dentre eles o de PLN e suas aplicações, processo de geração de base de dados, ferramentas de extração de dados de *webpages*, como *Web Crawler* e *Web Scraping*, métodos de pré-processamento de textos e conceitos sobre Linguagem de marcação de hipertexto, do Inglês *HyperText Markup Language* (HTML) e trabalhos relacionados. Na sequência, é apresentada a abordagem para atingir o objetivo proposto. E por fim, são apresentados os resultados obtidos.

2 REFERENCIAL TEÓRICO

2.1 Processamento de Linguagem Natural

O processamento de linguagem natural visa realizar a comunicação entre ser humano e computador na linguagem do próprio ser humano, sendo possível compreender significados, contextos e usos. Dentre as atividades e recursos da PLN, pode-se listar a gramática, recurso fundamental de qualquer texto, pois este descreve o conjunto de regras que o texto deve seguir, analise léxica, que realiza a análise da estrutura do texto, processamento morfo-sintático, que realiza a análise da constituição do texto, e análise semântica, que faz a análise do significado das palavras e contexto da frase (GONZALEZ; LúCIA; LIMA, 2001).

Uma das maiores dificuldades que a PLN enfrenta é a ambiguidade. A ambiguidade ocorre quando é possível extrair mais de um significado de uma determinada expressão, porém, uma pessoa consegue interpretar corretamente, em virtude de suas experiências, cultura, contexto histórico, entre outros motivos (PINTO, 2015), enquanto um computador não possui essas características para conseguir diferenciar o significado correto em uma ambiguidade. Nesses casos, torna-se importante a identificação do contexto de cada palavra.

A seguir, lista-se algumas aplicações conhecidas na abordagem de PLN (PINTO, 2015):

- Classificação de Texto: tarefa condizente à representar um texto em forma vetorial representando suas informações. Usando as características extraídas, utilizar de algoritmos de aprendizagem de máquina e inteligência artificial para gerar um modelo que atenda o sistema ou aplicação em questão.
- Agrupamento de Texto: Através de análises, realizar o agrupamento de textos que possuam semelhanças entre si. Essa semelhança poderia depois ser descrita na forma de uma categoria à qual os textos pertencem.
- Extração de Tópicos: uma aplicação que busca reduzir dimensionalidades do conjunto de dados.
- Extração de Informações: tem o objetivo de representar uma informação não estruturada de forma estruturada, que pode ser utilizada no futuro.

2.2 *Web Crawling* e *Web Scraping*

Web Crawler é um método computacional usado para realizar o *web crawling*. Esse método consiste no processo de navegar por diversas páginas da web de forma automatizada, objetivando realizar extração do conteúdo das mesmas. O processo parte de uma ou mais páginas raízes, de onde são extraídos e analisados todos os links de URLs contidos na página.

Esses links são adicionados à uma lista para repetir o processo de extração e análise de cada um deles, navegando assim por uma grande quantidade de páginas em pouco tempo. A navegação pode ocorrer de forma genérica, atingindo o máximo de páginas possível, ou de forma focada, abordando um assunto específico ou grupo de páginas específicas, ou ainda de forma distribuída, utilizando recursos de processamento paralelo para atingir diversas páginas (KAUSAR; DHAKA; SINGH, 2013).

De acordo com uma pesquisa de Menczer (2002), o *web crawling* é uma tarefa importante na recuperação de informações na web, pois permite indexar páginas da web e mantê-las atualizadas. Além disso, o *web crawling* também é útil para coletar dados em larga escala para análise e extração de conhecimento.

Web Scraping consiste no processo de extrair dados não estruturados de páginas da *Web* e estruturá-los para que possam ser analizados e armazenados em bases de dados (SIRISURIYA, 2015) e, até o surgimento de Interface de Programação de Aplicação, do Inglês *Application Programming Interface* (API), era o único meio para aplicações resgatarem informações da Internet.

O processo de *scraping* começa com a requisição de páginas *web* por meio do protocolo *Hypertext Transfer Protocol* (HTTP), retornando assim documentos HTML. Esses documentos passam por um processo de análise e pré-processamento, eliminando as informações indesejadas e por fim transformando as informações restantes, deixando-as prontas para uso (PERSSON, 2019).

Os métodos analisadores mais conhecidos para esse tipo de aplicação são expressões regulares, analisador HTML, analisador *Document Object Model* (DOM), e mais recentemente, analisador baseado em visão computacional (KHDER, 2021).

Por conta da estruturação de dados e da flexibilidade da extração, o *Web Scraping* pode ser encontrado em aplicações de mineração de dados, *marketing*, ferramentas de pesquisas e combinação de dados (SINGRODIA; MITRA; PAUL, 2019).

Em resumo, o *web crawling* é uma técnica mais geral para coletar informações da web, enquanto o *web scraping* é uma técnica mais específica para extrair informações estruturadas de uma página web. Ambas as técnicas são úteis para coletar dados da web, e podem ser usadas em conjunto para obter informações mais abrangentes sobre a web e seus conteúdos.

2.3 Métodos de Pré-processamento

Executar pré-processamento na base de dados de textos coletados, possui diversos objetivos, tais como: como identificar dados corrompidos, atributos irrelevantes, informações faltantes, ou até alterar a estrutura da base dos dados, tudo isso para que as próximas etapas sejam mais efetivas (BATISTA, 2003). Além disso, por tratar-se de dados textuais, pode ocorrer de existir uma palavra escrita de várias formas diferentes, como "Peixe", "peixe"e "PEIXE", porém as três são equivalentes e, para tratar estas situações, pode-se utilizar a padronização de

palavras, transformando todas as letras maiúsculas em minúsculas, remoção ou substituição de números e até caracteres especiais.

Dentre os métodos e técnicas, tem-se expressões regulares - usadas para identificar padrões de textos e caracteres, a *tokenização* - usado para estruturar textos em formas de listas, o *Stemming* - tratando prefixos e sufixos das palavras, ou fazendo uma normalização de verbos por meio da técnica de lematização (COPATTI, 2022).

2.3.1 Analisadores

Analisadores tem o objetivo de ler um documento e prover ao usuário uma interface para acessar o documento. Existem dois tipos principais de analisadores, são os analisadores DOM e analisadores denominados *Simple API for XML* (SAX), em que o analisador DOM utiliza-se de uma estrutura de árvore completa do documento *Extensible Markup Language* (XML) em memória, enquanto o SAX é orientado à eventos (LI, 2009).

Sendo assim, o analisador DOM lê o documento inteiro e constrói uma representação em forma de árvore, onde cada elemento do documento é representado por um nó. De acordo com o autor Li (2009), devido à sua estrutura, o analisador DOM não é eficiente para uso de memória, devido à necessidade de ler o documento inteiro, porém é conveniente para acessos complexos e aleatórios que buscam informações globais do documento. Além disso, o analisador DOM tem a capacidade de ler e escrever no documento. Já o Analisador SAX não necessita que o documento inteiro seja lido, pois este tem a capacidade de armazenar apenas porções de interesse da aplicação, assim sendo mais eficiente em memória, além de lidar melhor com documentos maiores. Devido à característica de não armazenar o documento inteiro, o analisador SAX tem apenas capacidade de leitura.

2.3.2 Expressões Regulares

Expressoes Regulares, do inglês *Regular Expression* (Regex) é uma forma de descrever textos, palavras, ou conjuntos de caracteres (ROBBINS, 2001). As expressões regulares são escritas em uma linguagem de programação específica e podem incluir caracteres especiais que representam quantificadores, conjuntos de caracteres, grupos e outras estruturas de busca. Elas são uma ferramenta poderosa para trabalhar com strings, pois permitem realizar buscas e substituições complexas com uma sintaxe curta e concisa.

Destaca-se que as expressões regulares são amplamente usadas em aplicações como processamento de texto, validação de formulários, processamento de arquivos, e muito mais. As expressões regulares permitem realizar verificações sofisticadas em strings, localizando padrões específicos e extrair informações relevantes, como endereços de e-mail ou números em uma *string* (Goyvaerts, 2009).

Para aplicações de *web scraping*, dado um conjunto de letras ou números, expressões regulares tem como objetivo encontrar todos os conjuntos, campos e sequências que contenham aquele conjunto. Expressões regulares se tornam úteis para ferramentas de buscas na Internet e conseguem identificar uma grande quantidade de padrões, tais como: datas, horário, endereço de e-mail, senhas de usuários, dados em colunas, dados entre *tags*, RG, CPF, entre outros padrões (JARGAS, 2012). Para *web scraping*, existem pacotes de desenvolvimento em linguagens de programação como Python que estruturam as expressões regulares, facilitando seu uso. Além disso, é possível que os dados extraídos possam conter caracteres especiais devido a linguagem HTML contida nas páginas (KHDER, 2021).

2.3.3 Métodos inteligentes de pré-processamento

Existem casos particulares de ruídos, a partir dos quais não é possível determinar um padrão ou alguma lógica para tratá-los por meio de expressões regulares. Nesses casos, pode-se propor a utilização de técnicas de inteligência artificial, a fim de diferenciar o que é ruído do que não é. O modelo pré-treinado contextualizado *Bidirectional Encoder Representations from Transformers* (BERT) (DEVLIN *et al.*, 2019) pode ser aplicado na identificação de ruídos textuais, já que seu mecanismo permite reconhecer o contexto de uma palavra dentro de um texto.

Foram realizados alguns trabalhos voltados para o pré-processamento e remoção de ruídos em postagens do Twitter ex. (RAMACHANDRAN; PARVATHI, 2019; KENNEDY, 2021; JIANQIANG; XIAOLIN, 2017). A maioria dos trabalhos relacionados à limpeza e pré-processamento de textos não tem aplicado métodos inteligentes a fim de filtrar ruídos ((SINGH; KUMARI, 2016; JIANQIANG; XIAOLIN, 2017)). Até o momento nenhum trabalho foi publicado oficialmente, visando o pré-processamento de textos de requisitos da web para engenharia de software. Apenas a monografia desenvolvida por Copatti (2022), é que apresentou uma abordagem inteligente para a limpeza de textos da área de engenharia de software. Essa abordagem se baseia no uso de *embeddings* BERT para agrupar palavras estranhas do texto, e assim eliminá-las.

2.4 Características de HTML

A linguagem HTML é utilizada para descrever elementos de *webpages*, que são denominados *tags*. Os elementos podem ser texto, parágrafo, imagem, títulos, entre outros (FERREIRA, 2013), podendo-se resumir a elementos textuais ou estruturais. As *tags* devem seguir a regra de serem abertas por «"e »", contendo o nome do elemento, e fechadas da mesma forma porém adicionando "/"após o «", seguindo a estrutura do Quadro 1.

Outra característica importante das *tags* são os atributos das mesmas. Os atributos são descritos na abertura da *tag* e recebem valores que podem alterar as propriedades e funcionamento do elemento, como por exemplo na Tabela 1, o qual possui o atributo "href", fazendo assim o elemento ter a funcionalidade de redirecionar ao endereço do valor atribuído à "href".

Na Tabela 2, é disposta uma listagem das principais *tags* utilizadas para criação de *webpages*. Essas características se fazem importantes nesse trabalho, pois desde a extração de dados via mecanismos de *web crawling* até o pré-processamento desses dados, será necessário envolver tais características.

Tabela 1 – Código HTML descreve uma tag HTML e suas características de forma simples.

```
<div href="www.google.com">conteúdo </div>
```

Fonte: Autoria Própria.

Tabela 2 – Principais *tags* usadas para construir páginas na linguagem HTML. Nela são listados tantos elementos textuais quanto estruturais.

Tag	**Funcionalidade**
<header>	Elemento estrutural de cabeçalho
<main>	Elemento estrutural de centro da página
<section>	Elemento estrutural de componentes da página
	Elemento estrutural de componentes da página
<footer>	Elemento estrutural de rodapé da página
<h1> à <h6>	Elementos textuais de título
	Elemento textual em linha
<p>	Fonte de dados para extração

Fonte: Autoria Própria.

2.5 Trabalhos Relacionados

Sirisuriya (2015) trata a estruturação dos dados como objetivo geral de um *web scraper*, e como opções de estruturas são tabelas, planilhas, bancos de dados relacionais e também separação de dados por tabuladores ou vírgulas. Em sua pesquisa, Sirisuriya (2015) exemplifica também diferentes aplicativos, como *web scraper* nas extensões de navegadores, os quais realizam extrações de dados, transporta-os para arquivos Valores separados por vírgula, do inglês *Comma Separated Values* (CSV), o *Easy Web Extract* que é um software desenvolvido para extrações rápidas e simples com opções de exportar dados em formato .csv, .txt, HTML, XML e formatos para bancos relacionais Structured Query Language (SQL), e o *Scrapy*, um *framework* desenvolvido em Python para implementar um *Web Scraper*.

Um trabalho que explorou a construção de base de dados é dado por (JIAXIANG, 2020), em que foram obtidos comentários de publicações da plataforma TikTok para análise de sentimento. Os comentários foram escolhidos de forma aleatória sob a proposta de que as palavras

devem refletir apelos sentimentais. Assim, foram extraídos nome de usuário, idade, gênero, número de *likes* e comentário das publicações, e organizados em um arquivo .csv para serem processados e utilizados posteriormente.

Brito et al. (BRITTO; PACIFICO; PACíFICO, 2020) trazem em sua pesquisa o processo de construção de uma base de dados textual e pré-processamento dos textos, realizando as tratativas para letras maiúsculas, remoção de *stopwords*, além de remover outros símbolos como *hashtags*. Os autores selecionaram um total de dez mil comentários na língua portuguesa, relacionados à aplicativos *mobile* da Apple, são escolhidos modelos de inteligência artificial para realizar a tarefa de classificação dos dados coletados.

Como exemplo de configuração de *web crawler*, (MACHUCA; GALLARDO; TOASA, 2021) apresenta uma busca por dados relacionados à COVID19 na rede social Twitter. Sua busca se baseava em datas, tópicos e linguagem, buscando encontrar as cinquenta mil publicações em inglês mais vistas pela rede social ao longo do mês. A pesquisa também apresenta as técnicas de pré-processamento de texto como normalização, *tokenização* e *Stemming*. Através dos dados coletados e da aplicação de algoritmos de classificação, foi possível realizar a análise de sentimento EM relação às publicações ao longo do ano, no que se refere à COVID19.

Não foi identificado nenhum trabalho destinado à criação de uma base de dados textual em português para a área de engenharia de software.

3 MATERIAS E MÉTODO

Para atingir os objetivos propostos para este trabalho, foi definida a metodologia apresentada a seguir, a qual consiste de materiais necessários ao desenvolvimento do trabalho, bem como aos métodos a serem implementados.

3.1 Materiais

Para desenvolvimento do trabalho, foram utilizados os materiais apresentados na 3:

- Linguagem de Desenvolvimento: o desenvolvimento será dado em Python, por sua facilidade em manipulação de dados. Além disso, exemplos de aplicação de expressões regulares são dadas por (FRIEDL, 2002), utilizando a biblioteca "re"do Python. Essa biblioteca é amplamente utilizada atualmente. Outro fator que influencia a escolha da linguagem é a necessidade de exportar grandes quantidades de dados para arquivos *CSV* ou outros formatos baseados em tabelas. Isso é possível com a biblioteca Pandas, por exemplo.
- Fonte de Dados: O *StackOverflow* está entre as maiores plataformas de discussão de desenvolvimento de software e projetos. Cada item de dado apresenta título, perguntas, respostas e algumas métricas da comunidade, como visualização e pontuação do comentário (*likes*) (HIN, 2020). A plataforma também conta com um mecanismo de identificação de categoria que pode ou não ser utilizado, são as chamadas *tags*. Ambas as plataformas demonstram uma abundância de dados relacionados ao assunto de Engenharia de Software e desenvolvimento, e uma grande variedade de informações, justificando assim a escolha como fonte de dados que satisfará a necessidade deste estudo.

Tabela 3 – Lista de materiais mapeados e suas finalidades. Estes serão utilizados durante a etapa de desenvolvimento.

Material	Finalidade
Python	Linguagem de Programação
Pandas	Biblioteca para manipular dados
Requests	Biblioteca para solicitações HTTP
Re	Biblioteca para expressões regulares
Matplotlib	Biblioteca para gerações de gráficos 2D e 3D
Sklearn	Biblioteca para métodos de aprendizado de máquina
Sentence Transformer	Biblioteca para criação de *embeddings* de sentenças
Stack Overflow	Fonte de dados para extração

Fonte: Autoria Própria.

3.2 Método

Considerando a necessidade de navegar por uma grande quantidade de páginas em busca de informações, definiu-se pela aplicação de um *web crawler* para realizar esse trabalho, o que se justifica com base em sua definição apresentada no Capítulo 2. A geração de uma base de dados textual usando *web crawler*, é o processo de coletar dados de diferentes sites na Internet e armazená-los em uma base de dados para futura análise e uso. De acordo com alguns autores, o processo de geração de uma base de dados textual usando *web crawler* envolve as seguintes etapas:

1. **Definição do objetivo**: É necessário definir claramente o objetivo da coleta de dados e os tipos de informações que se deseja armazenar na base de dados (KUSHAL; DEB; MUKHERJEE, 2018). Com base nas perguntas e respostas do *Stack Overflow*, foram escolhidos como dados alvo a serem extraídos: o título do tópico, o conteúdo de pergunta descrito pelo usuário, as categorias a que o tópico pertence, e os comentários de respostas à pergunta. Na Figura 1, é apresentado um exemplo de postagem no *Stack Overflow*, em que estão presentes: o elemento título como sendo *"Vulnerabilidades ao criar projeto no React Native"*, logo abaixo a pergunta realizada e as categorias a que essa pertence, neste caso, sendo unicamente a categoria referente à linguagem "R". Além desses itens, outros elementos do próprio site estão presentes, como o contador de *likes*, data em que foi feita a pergunta e quantas visualizações possui.

2. **Configuração do mecanismo de extração de dados**: O *web crawler* deve ser configurado para navegar apenas pelos sites que são alvos para o objetivo da coleta de dados. Para isso, é importante estabelecer regras para evitar a navegação por sites que não sejam relevantes, ou que contenham informações desnecessárias (CHOI; KIM, 2018). O tempo necessário para a coleta de dados depende do tamanho e/ou do volume de dados, além da complexidade dos sites que serão analisados (ALHARBI; ALGHAMDI, 2019). Para encontrar as próximas páginas a partir da origem, foi necessário identificar padrões na estrutura das páginas do portal. Sendo assim, a fim de localizar as páginas seguintes, foram utilizados valores e palavras-chaves encontrados no documento HTML desta página. No *Stack Overflow* existe uma lista paginada, contendo todas as perguntas já abertas por usuários, e para prosseguir, de página em página, existe um botão com a descrição "Next". Assim, é possível utilizar esse padrão usando expressões regulares, a fim de extrair o endereço da próxima página.A Figura 2 apresenta o processo completo.

3. **Obtenção das páginas**: Na etapa de navegação, os endereços das páginas finais serão armazenados em uma estrutura de dados e, após a etapa de navegação, será iniciado o processo de leitura do documento HTML de cada página armazenada ante-

Figura 1 – Conteúdo de um tópico aberto no *Stack Overflow*, contendo o título, conteúdo da pergunta e categorias.

riormente. Em seguida, serão coletados os dados previamente definidos, por meio de expressões regulares. Ilustra-se o fluxo para esta atividade na Figura 4.

4. **Coleta de dados**: Com a parte de navegação desenvolvida, as páginas de perguntas precisam ser acessadas e a informação contida nela deve ser extraída. Desta forma, nesta etapa são coletados os dados previamente definidos, por meio de expressões regulares. Ilustra-se o fluxo para esta atividade na Figura 4.

   Conforme é possível observar na Figura 4, após concluída a navegação a partir da página de origem e gerada a lista de endereços dessas páginas, cada uma das páginas será acessada, utilizando o método de *Web Scraping*. O objetivo do método é encontrar dados específicos (título, categorias, assunto e comentários). O número de páginas encontradas, depende da fonte de dados (*Stack Overflow*).

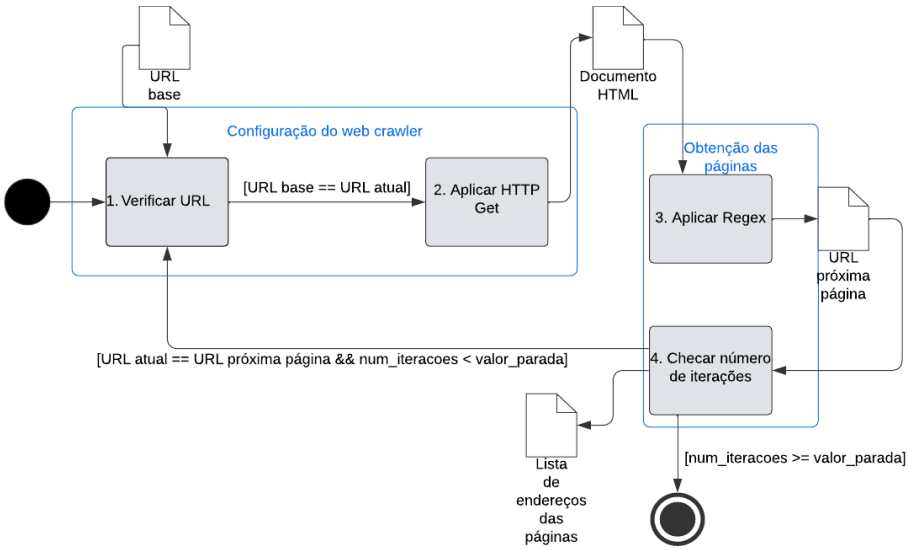


Figura 2 – Configuração e aplicação do *web crawler* e obtenção das páginas adequadas para a coleta de dados.

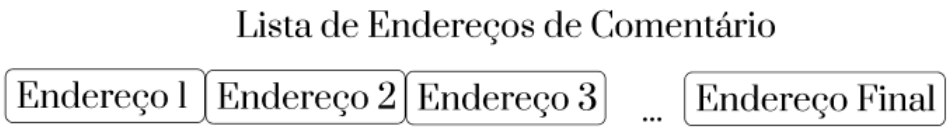


Figura 3 – Resultado esperado da etapa de obtenção das páginas, uma lista de endereços do provedor de dados contendo todas as páginas que serão usadas para coleta de dados.

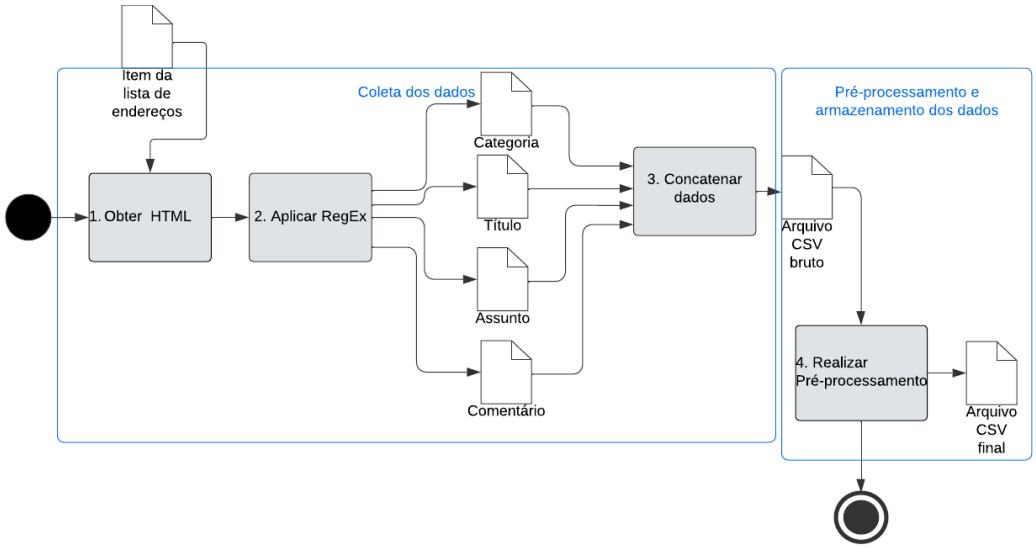


Figura 4 – Aplicação do *Web Scraping* para coleta de dados específicos, seguida de pré-processamento e armazenamento dos dados.

5. **Pré-processamento dos textos**: Os dados textuais coletados foram então pré-processados, buscando eliminar ruídos desnecessários ao reconhecimento do seu contexto. Para isso, foram aplicados métodos de limpeza e substituição de ruídos, os quais se basearam em Regex. Após a limpeza, os dados são escritos em um arquivo *CSV* que representa a base de dados final.

6. **Armazenamento de dados**: Os dados coletados precisam ser armazenados de forma que permita uma futura análise e utilização em pesquisa de PLN. Sendo assim, é importante escolher um formato de armazenamento apropriado (ELBASSUONI; AL-ANSARI, 2019). Como tratam-se de dados não-estruturados, não se aplica um banco de dados relacional. Também não se trata de um banco de dados NoSQL, pois não requer que sejam escaláveis e distribuídos, entre outras características desse tipo de banco (HAN; KAMBER; PEI, 2011). Desta forma, o formato *Comma-Separated Values* (*CSV*) é usado para armazenar e aplicar dados em formato de texto simples, onde os valores são separados por vírgulas ou outros caracteres delimitadores. Embora alguns bancos de dados NoSQL possam suportar a importação de dados em formato *CSV*, isso não significa que o *CSV* em si seja um formato NoSQL.

   A geração de uma base de dados textual usando *web crawler* é útil em muitos campos, como mineração de dados, análise de sentimentos, análise de tendências, entre outros (GOYAL; BHATIA, 2016). É importante lembrar que a geração de uma base de dados usando *web crawler* pode ser ilegal, dependendo das leis e regulamentos locais. Sendo assim, é importante verificar se a coleta de dados está de acordo com as leis e regulamentos aplicáveis antes de iniciar o processo. Além disso, é importante garantir que a privacidade dos dados coletados seja preservada, especialmente quando se trata de informações pessoais (DOMINGOS; RICHARDSON, 2000). Isso pode ser alcançado através de práticas éticas de coleta de dados, como a obtenção de consentimento explícito dos indivíduos e a criptografia dos dados coletados.

   A geração de uma base de dados textual usando *web crawler* também pode enfrentar desafios técnicos, como a detecção e utilização de sistemas de bloqueio de *web crawlers*, a identificação de informações relevantes e a avaliação da qualidade dos dados coletados (METWALLY; AGRAWAL; ABBADI, 2005). Portanto, é importante considerar esses desafios ao planejar e executar o processo de geração de base de dados usando *web crawler*.

4 RESULTADOS E DISCUSSÕES

Nesta seção são demonstrados os resultados da abordagem proposta, descrevendo dificuldades de cada etapa e apresentando os dados extraídos das mesmas. Também é apresentada a base de dados e levantado um demonstrativo dos dados contidos na mesma, validando a metodologia apresentada. Por fim é elaborada uma comparação entre os dados extraídos originalmente, sua transformação durante as etapas, até o dado na sua forma final.

```
1  class Crawler():
2      def __init__(self, domain, startUrl, next):
3          self.domain = domain
4          self.url = startUrl
5          self.nextRegEx = next
6          self.stack = Stack(domain)
7
8      def Navigate(self):
9          for i in range(0,150):
10             response = requests.get(self.domain + self.url,
11             headers=headers, verify=False)
12             content = str(response.content)
13             self.StashLinks(content)
14             #NAVEGAR
15             nexts = re.findall(self.nextRegEx, content)
16             url = re.findall(r'href="[^"]*"', nexts[0])
```

Figura 5 – Código implementado para realizar a etapa de *Web Crawler*. A classe recebe o endereço no qual deve navegar e a expressão regular que busca pela próxima página.

A Figura 6 demonstra a estrutura de uma página de lista de perguntas do *Stack Overflow*. O processo de navegação incluiu 150 (cento e cinquenta) páginas do *Stack Overflow*, totalizando um conjunto de 2250 (duas mil duzentas e cinquenta) perguntas. Os endereços das perguntas separadas foi organizado em um arquivo para ser consumido pelo emphWeb Scrapper. Na Figura 5 está descrito o trecho de código utilizado para a navegação. Nesse trecho, o *Web Crawler* parte do endereço base, este sendo a primeira página de perguntas do Stack Overflow, e identifica através das expressões regulares o botão de prosseguir para próxima página e o endereço da próxima página, e este processo é repetido até uma condição de parada que seria, exceto por não achar a próxima página, um valor de iterações realizadas, definindo uma parada após 150 (cento e cinquenta) iterações. Este valor foi escolhido devido, após algumas etapas de teste, notar-se que o provedor de dados bloqueia ao ocorrerem muitos acessos em um curto período de tempo, sendo necessário aguardar para que sejam liberadas novas requisições.

A utilização da técnica de *Web Scrapper* resultou na extração de 2182 (duas mil cento oitenta e duas perguntas). Isso porque, apesar do *crawler* ter extraído um maior número de endereços, alguns desses endereços encontravam-se indisponíveis ao acesso, gerando um erro de "Página não encontrada". Ainda nessa etapa, devido à grande quantidade de *tags* do do-

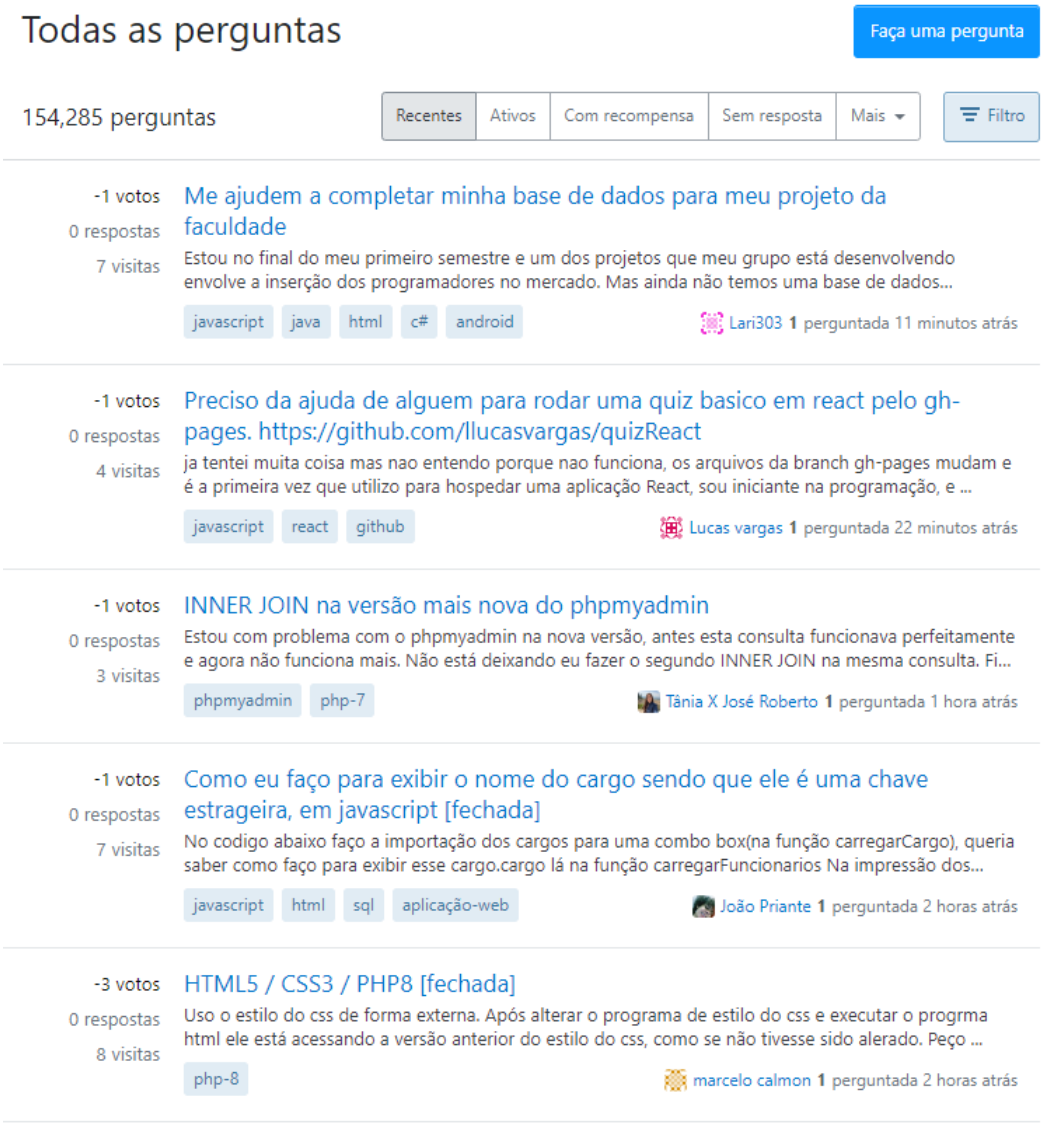


Figura 6 – Exemplo de página que contém a lista de perguntas do *Stack Overflow*. Cada página contém 15 (quinze) perguntas, além do botão de navegação "próximo"que foi utilizado para navegação do *Web Scrapper*

cumento HTML, foi necessário identificar a estrutura da página, pois apenas aplicando expressões regulares eram retornados conteúdos indesejáveis, como "Categorias Recomendadas", "Perguntas Relacionadas", cabeçalho e rodapé da página, informações essas que estão fora do escopo do projeto.

Com o padrão estrutural das páginas identificado, o processo de extração dos dados foi aplicado, utilizando as expressões regulares para primeiramente identificar *tags* para separar o texto futuramente, sendo elas as *tags* que identificavam começo de pergunta, começos de respostas, começo de dados de usuários, conforme mostra a Figura 7. Assim, a partir da estrutura das perguntas e respostas, sabe-se que o conteúdo das mesmas estarão entre um começo de pergunta ou resposta e o começo das informações de usuários. Na Figura 8, estão descritas todas as expressões que foram utilizadas na etapa de extração dos dados.

Tabela 4 – Resultado do primeiro processo (Figura 2), uma lista de endereços utilizados pelo *Web Scraper* para acessar as páginas de perguntas e extrair as informações da mesma.

Linha	Endereço
1	https://pt.stackoverflow.com/questions/583705/dividir-os-resultados-de-uma-consulta-utilizando-mysqli-fetch-assoc
2	https://pt.stackoverflow.com/questions/583701/laravel-me-retorna-esse-erro-no-campo-select-attempt-to-read-property-id-on-b
3	https://pt.stackoverflow.com/questions/583700/redirecionamento-de-uma-servlet-para-jsp
...	...
2249	https://pt.stackoverflow.com/questions/570423/pegar-a-quantia-de-casas-decimais-de-um-double-java
2250	https://pt.stackoverflow.com/questions/570419/converter-xlsx-em-xls-usando-python-e-libreoffice-calc

Fonte: Autoria Própria.

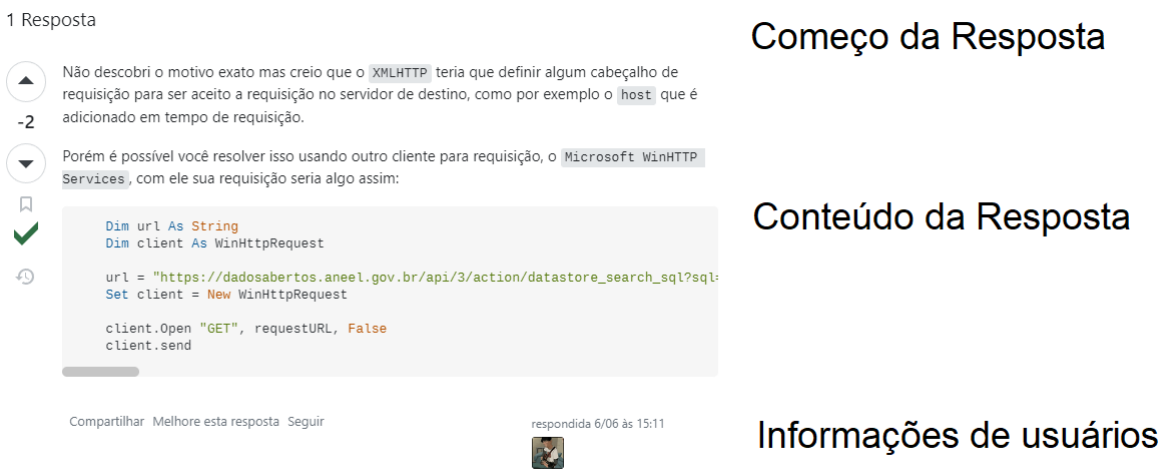


Figura 7 – Exemplo de estruturação da página em que o *Web Scrapper* extraiu os dados.

Após identificado o padrão das perguntas e respostas, foi possível extrair: título, pergunta, respostas, número de respostas, categorias e número de categorias. Os textos extraídos apresentaram diversos caracteres e trechos em padrões Unicode, UTF-8, HTML e hexadecimal (para caracteres especiais). Através do padrão identificado, foi possível substituir os trechos por um valor equivalente no texto. Caracteres como vírgula, ponto e vírgula, dois pontos e ponto, foram removidos do texto. Por fim, para trechos em que existem imagens ou códigos, esses trechos foram substituídos por um identificador indicando que havia um elemento daquele tipo naquele local do texto. Nem todos os trechos de códigos foram substituídos, isto porque em alguns casos, usuários que escreveram a pergunta não formataram o trecho de código, deixando este trecho como um trecho de texto normal, o que inviabiliza o tratamento do código nesses casos pois seria necessário identificar e tratar cada sintaxe de cada linguagem de forma diferente. A Tabela 5 indica as substituições feitas e seus respectivos substitutos.

Com os tratamentos concluídos, os dados foram organizados em um arquivo *CSV*, onde as colunas estão de acordo com a Tabela 6. Para as categorias, foi decidido manter cinco

```
class Scrapper():
    def __init__(self, url):
        self.url = url

    def getHtml(self):
        response = requests.get(self.url, headers=headers, verify=False)
        self.content = str(response.content)
        self.status = response.status_code

    def getProps(self, charTable, htmlTable):
        ## Pegar title
        titleRegEx = r'<h1[^>]*><a[^<>]*class="question-hyperlink"[^<>]*>
        [^<>]*</a></h1>'
        titles = re.findall(titleRegEx, self.content)
        aux = r'>[^<]*</a>'
        titleString = re.findall(aux, titles[0])
        title = titleString[0]
        title = title[1:len(title)-4]
        ## Pegar Pergunta
        questionRegEx = r'<div class="question js-question" data-
        questionid="[^"]*" data-position-on-page="[^"]*" data-score="
        [^"]*"  id="question">.*<div id="answers"[^>]*>'
        questionContent = re.findall(questionRegEx, self.content)
        question = questionContent[0]
        ## Pegar Respostas
        ansRegEx = r'<div[^>]*id="answers"[^>]*>.*<div[^>]*id="sidebar"
        [^>]*>'
        answers = re.findall(ansRegEx, self.content)
        answers = answers[0]
        ## Pegar Tags
        tagRegEx = r'<a[^>]*rel="tag"[^>]*>'
        tags = re.findall(tagRegEx, question)
```

Figura 8 – Código implementado para realizar a etapa de *Web Scrapper* - a classe recebe o endereço da pergunta e utiliza-se de expressões regulares para extrair dados da página.

Tabela 5 – Lista de trechos substituídos nos textos de perguntas e respostas

Conteúdo	Texto Final
Trechos de Código	(CODE)
Imagens	(IMAGE)
Começo de Respotas	(RESPOSTA TAG)
Informações de Usuários	(USER-DATA)
Indicativo de Pergunta Fechada	(FECHADA)

Fonte: Autoria Própria.

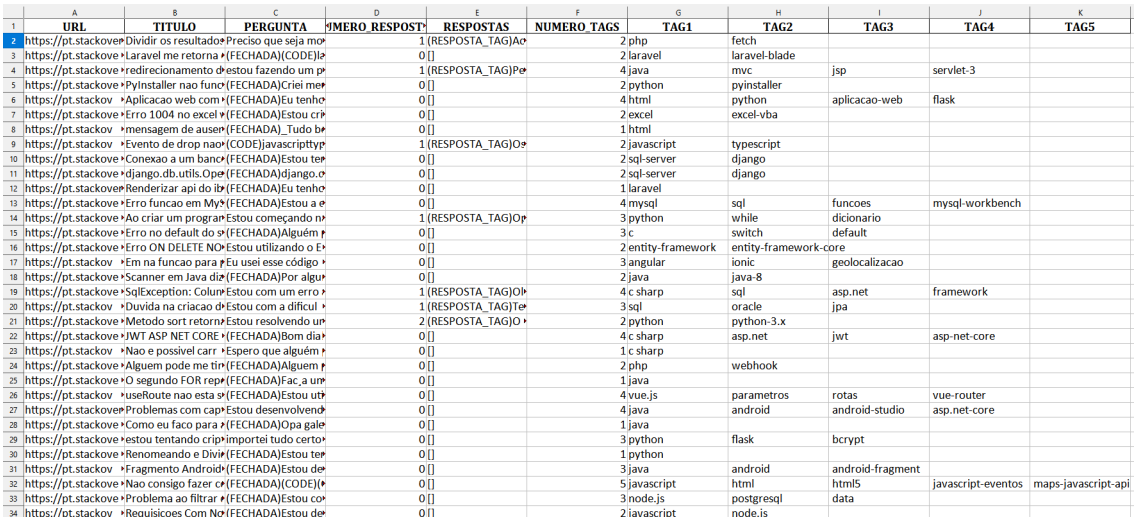
colunas, esta quantidade devido a não ter sido observada nenhuma pergunta com mais de cinco categorias. Em casos em que há menos categorias, as colunas de maior índice não são preenchidas.

Tabela 6 – Lista das colunas e seus respectivos conteúdos no arquivo gerado como resultado

Coluna	Conteúdo
URL	Endereço acessado pelo *Scrapper*
Título	Titulo da postagem
Pergunta	Conteúdo da pergunta na postagem
Número respostas	Número de respostas feitas à postagem
Respostas	Todas as respostas da pergunta em forma de texto
Número tags	Número de categorias da pergunta
Tag1	Descrição da primeira categoria
Tag2	Descrição da segunda categoria
Tag3	Descrição da terceira categoria
Tag4	Descrição da quarta categoria
Tag5	Descrição da quinta categoria

Fonte: Autoria Própria.

Após gerar o resultado demonstrado na Figura 9, o arquivo foi utilizado com intuito de realizar uma breve descrição dos dados contidos nele. No resultado, foram observadas características da base, dentre elas, número de perguntas e número de respostas, categorias encontradas, entre outros. Foram então quantificadas as categorias com maior volume de perguntas. O número de respostas foi 1480 (um mil quatrocentos e oitenta), totalizando 702 (setecentas e duas respostas) a menos que perguntas, isso pois nem todas as perguntas possuem respostas e outras possuem mais que uma resposta. Na Figura 10, o item *b* demonstra essa característica, onde tem-se o número de perguntas (ocorrências) que contêm determinada quantidade de respostas.



	A	B	C	D	E	F	G	H	I	J	K
1	URL	TITULO	PERGUNTA	MERO_RESPOST	RESPOSTAS	NUMERO_TAGS	TAG1	TAG2	TAG3	TAG4	TAG5
2	https://pt.stackover	Dividir os resultados	Preciso que seja mo	1	(RESPOSTA_TAG)Ac	2	php	fetch			
3	https://pt.stackove	Laravel me retorna	(FECHADA)(CODE)la	0	[]	2	laravel	laravel-blade			
4	https://pt.stackove	redirecionamento d	estou fazendo um p	1	(RESPOSTA_TAG)Pe	4	java	mvc	jsp	servlet-3	
5	https://pt.stackove	PyInstaller nao func	(FECHADA)Criei me	0	[]	2	python	pyinstaller			
6	https://pt.stackov	Aplicacao web com	(FECHADA)Eu tenho	0	[]	4	html	python	aplicacao-web	flask	
7	https://pt.stackove	Erro 1004 no excel	(FECHADA)Estou cri	0	[]	2	excel	excel-vba			
8	https://pt.stackov	mensagem de ausen	(FECHADA)_Tudo b	0	[]	1	html				
9	https://pt.stackov	Evento de drop nao	(CODE)javascripttyp	1	(RESPOSTA_TAG)O	2	javascript	typescript			
10	https://pt.stackove	Conexao a um banc	(FECHADA)Estou te	0	[]	2	sql-server	django			
11	https://pt.stackove	django.db.utils.Ope	(FECHADA)django.o	0	[]	2	sql-server	django			
12	https://pt.stackover	Renderizar api do ib	(FECHADA)Eu tenho	0	[]	1	laravel				
13	https://pt.stackove	Erro funcao em My	(FECHADA)Estou a e	0	[]	4	mysql	sql	funcoes	mysql-workbench	
14	https://pt.stackove	Ao criar um progra	Estou começando n	1	(RESPOSTA_TAG)Op	3	python	while	dicionario		
15	https://pt.stackove	Erro no default do s	(FECHADA)Alguém p	0	[]	3	c	switch	default		
16	https://pt.stackove	Erro ON DELETE NO	Estou utilizando o E	0	[]	2	entity-framework	entity-framework-core			
17	https://pt.stackov	Em na funcao para p	Eu usei esse código	0	[]	3	angular	ionic	geolocalizacao		
18	https://pt.stackove	Scanner em Java diz	(FECHADA)Por algu	0	[]	2	java	java-8			
19	https://pt.stackove	SqlException: Colun	Estou com um erro	1	(RESPOSTA_TAG)OI	4	c sharp	sql	asp.net	framework	
20	https://pt.stackov	Duvida na criacao d	Estou com a dificul	1	(RESPOSTA_TAG)Te	3	sql	oracle	jpa		
21	https://pt.stackove	Metodo sort retorna	Estou resolvendo un	2	(RESPOSTA_TAG)O	2	python	python-3.x			
22	https://pt.stackove	JWT ASP NET CORE	(FECHADA)Bom dia	0	[]	4	c sharp	asp.net	jwt	asp-net-core	
23	https://pt.stackov	Nao e possivel carr	Espero que alguém	0	[]	1	c sharp				
24	https://pt.stackove	Alguem pode me tir	(FECHADA)Alguem p	0	[]	2	php	webhook			
25	https://pt.stackove	O segundo FOR rep	(FECHADA)Fac,a um	0	[]	1	java				
26	https://pt.stackov	useRoute nao esta s	(FECHADA)Estou ut	0	[]	4	vue.js	parametros	rotas	vue-router	
27	https://pt.stackover	Problemas com cap	Estou desenvolvend	0	[]	4	java	android	android-studio	asp.net-core	
28	https://pt.stackove	Como eu faco para a	(FECHADA)Opa gale	0	[]	1	java				
29	https://pt.stackove	estou tentando crip	importei tudo certo	0	[]	3	python	flask	bcrypt		
30	https://pt.stackove	Renomeando e Divi	(FECHADA)Estou te	0	[]	1	python				
31	https://pt.stackov	Fragmento Android	(FECHADA)Estou de	0	[]	3	java	android	android-fragment		
32	https://pt.stackove	Nao consigo fazer c	(FECHADA)(CODE)(	0	[]	5	javascript	html	html5	javascript-eventos	maps-javascript-api
33	https://pt.stackove	Problema ao filtrar	(FECHADA)Estou co	0	[]	3	node.js	postgresql	data		
34	https://pt.stackov	Requisicoes Com No	(FECHADA)Estou de	0	[]	2	iavascript	node.is			

Figura 9 – Leitura do arquivo *CSV* resultado do processo de pré-processamento de texto. O arquivo foi manipulado no software Excel para organização em colunas.

As categorias encontradas totalizaram 1111 (um mil cento e onze) categorias distintas. O gráfico *a* da Figura 10 demonstra a quantidade de categorias atribuídas às perguntas. Na Figura 11, o gráfico apresenta apenas as cem categorias com mais incidência, devido à grande quantidade de categorias distintas, a fim de exemplificar as características da base formulada.

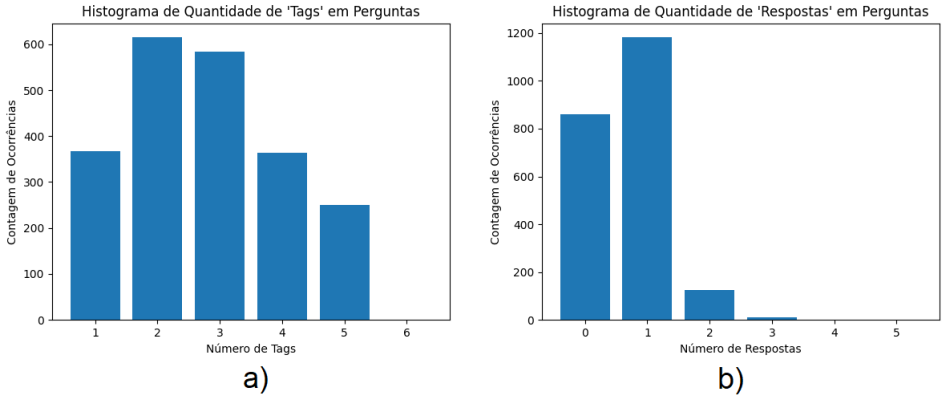


Figura 10 – Histogramas gerados através da biblioteca Matplotlib para ocorrências a) da quantidade de categorias em perguntas e b) da quantidade de respostas em perguntas

Apesar do número de categorias distintas, é possível notar que existem muitas categorias similares (ex: Python, Python3.x). Objetivando diminuir esse número de classes, é indicado como um trabalho futuro, a implementação de um método que permita o agrupamento dessas classes similares, evitando confundir o classificador, o que facilitaria o uso da base de dados em tarefas de classificação. O total de categorias encontradas, considerando as repetidas, foi de 6062 (seis mil sessenta e dois). É possível encontrar outras bases de dados semelhantes ao provedor de dados (*Stack Overflow*), até bases com maior quantidade de dados, como (ANNAMORADNEJAD; HABIBI; FAZLI, 2022), porém não são em Português, além de apresentarem diferentes tipos de dados relacionados ao Stack Overflow, como avaliações de perguntas, .

Por fim, para validar o processo, é comparada uma postagem em sua forma original e a forma apresentada após aplicar o algoritmo desenvolvido. Considera-se que os dados originais são as páginas *web* originais, apresentado na Figura 12. Na Figura 13, exemplifica-se o resultado do algoritmo para a página da Figura 12. A pergunta extraída como exemplo, de título "O que é zero copy", manteve seus textos originais, preservando o sentido original dos mesmos.

Considerando a diversidade de dados presentes na base, e apenas a título de teste da base, foi aplicado um método objetivando verificar, se a partir das categorias atribuídas para cada texto, os dados textuais similares geram agrupamentos. Para isso, primeiramente foi aplicado o modelo de linguagem pré-treinado BERT (DEVLIN *et al.*, 2019), utilizando para isso a biblioteca *SentenceTransformer*, a fim de converter cada texto de pergunta ou resposta para o formato de *embeddings*, o que, de acordo com os autores do BERT, mantém o contexto dos textos.

Em seguida, os *embeddings* referentes aos textos foram submetidos ao método *t-Distributed Stochastic Neighbor Embedding* (t-SNE) (MAATEN; HINTON, 2008), o qual aplica redução de dimensionalidade nos dados para serem observados de forma bidimensional ou tridimensional. Sendo assim, torna-se possível aplicar métodos de agrupamento por distância, agrupamento por densidade, entre outros. Desta forma, a dimensionalidade dos *embeddings* BERT, que eram de 768 (setecentos sessenta e oito), foi reduzida para dois, e então aplicado o

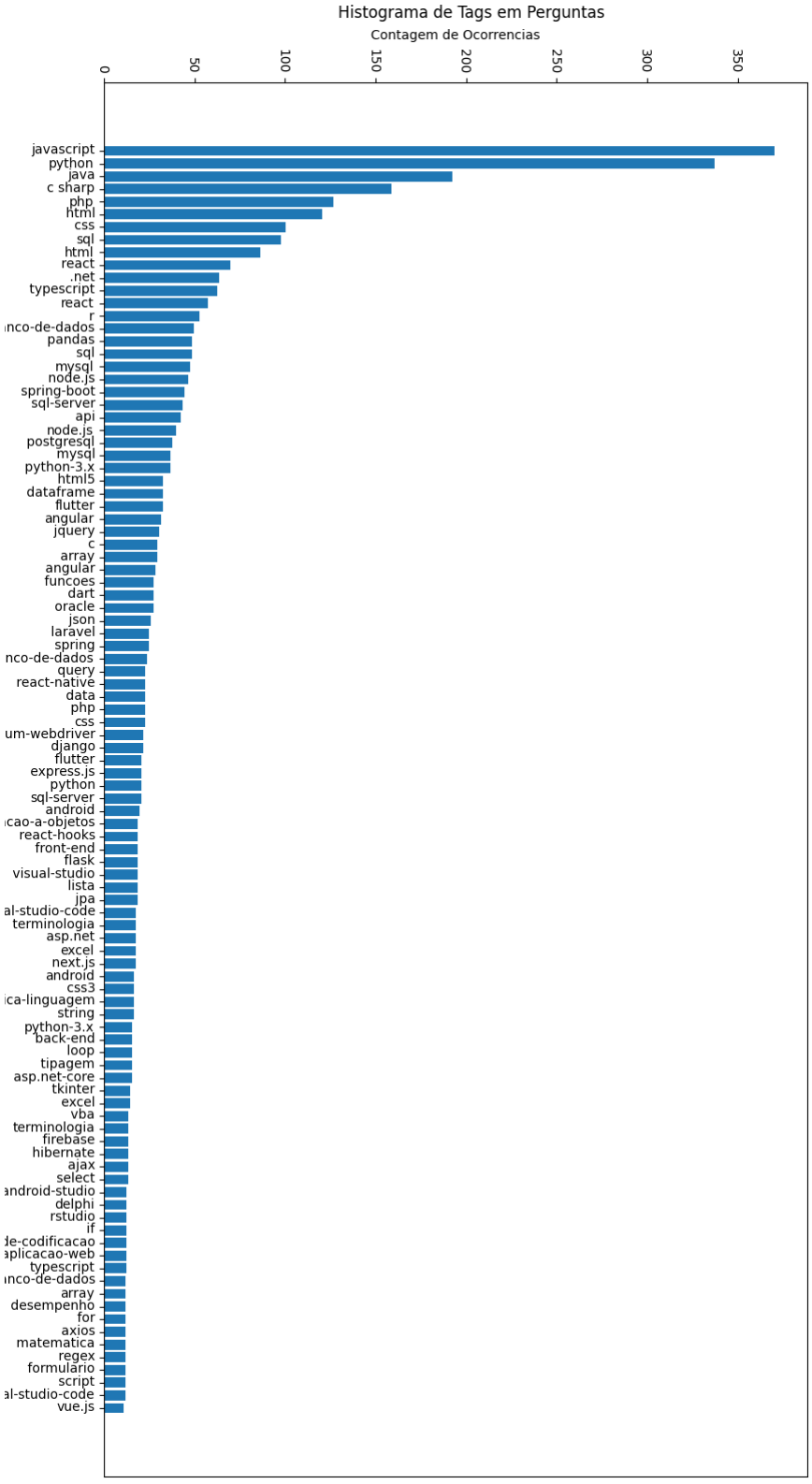


Figura 11 – Gráfico gerado através da biblioteca Matplotlib onde são apresentadas as cem categorias que mais contêm perguntas relacionadas nos dados extraídos.

método de t-SNE considerando quatro agrupamentos, estes agrupamentos considerados através do algoritmo de K-Means para as categorias de perguntas, demonstrado na Figura 14.

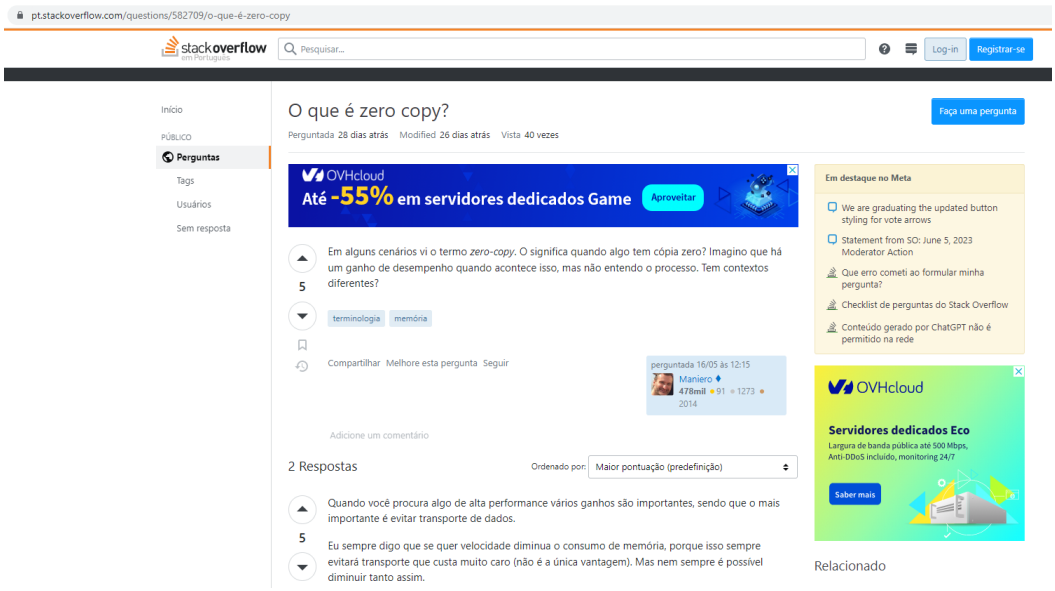


Figura 12 – Tela retirada do *Stack Overflow*, relacionada diretamente à pergunta "O que é Zero Copy", representa um exemplo de dado em sua forma original.

```
1  https://pt.stackoverflow.com/questions/582709/o-que-\%c3\%a9-zero-copy,
2 O que e zero copy?, Em alguns cenários vi o termo zero-copy. O significa
3 quando algo tem cópia zero? Imagino que há um ganho de desempenho quando
4 acontece isso, mas não entendo o processo. Tem contextos diferentes?
5 terminologiamemoria , 2, (RESPOSTA_TAG) Quando você procura algo de alta
6 performance vários ganhos são importantes sendo que o mais importante é
7 evitar transporte de dados Eu sempre digo que se quer velocidade diminua
8 o consumo de memória porque ... contextos compensa fazer isso (USER-DATA)
9 RESPOSTA_TAG)Com a técnica de zero-copy você pode transferir diretamente
10 os dados da tabela original para o sistema de destino sem fazer cópias
11 desnecessárias Em vez de criar uma nova tabela temporária os dados são
12 passados diretamente economizando tempo e recursos(USER-DATA),
13 2,terminologia, memoria,,,
```

Figura 13 – Texto extraído do arquivo (*CSV*) final. Este trecho refere ao conteúdo da Figura 12, porém apresenta o dado em sua forma pré-processada.

Os agrupamentos gerados possuem ainda grande variação entre suas perguntas, por exemplo perguntas de categoria Javascript ficaram no mesmo grupo de perguntas de Java, mas já mostra que a base de dados apresenta uma consistência com relação aos dados que possui. Por tanto, para melhorar os agrupamentos apresentados na Figura 14, seria necessário realizar um tratamento para as categorias, agrupando similares, e definindo um número de agrupadores que esteja de acordo com as categorias através de algoritmos como K-Means. Também deve-se avaliar o conteúdo da pergunta em si, visto que o algoritmo de t-SNE pode realizar agrupamentos com conteúdos interpretados das perguntas, aproximando questões relacionadas a modelos de desenvolvimentos e tópicos não necessariamente relacionados à categoria.

Ao acessar *https://github.com/maxjunior98/crawlerScraper* é possível encontrar os códigos aplicados e desenvolvidos para este projeto.

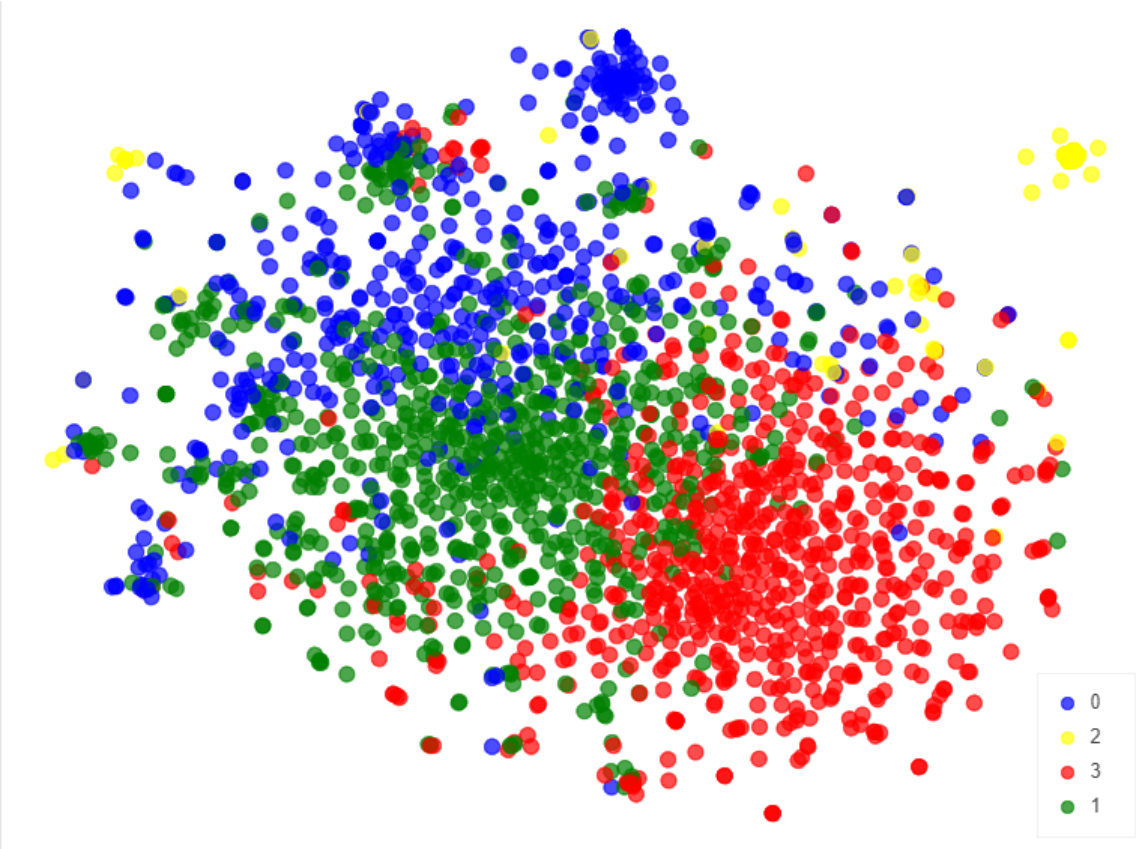


Figura 14 – Resultados da redução de dimensionalidade de textos do *StackOverflow* por meio do método t-SNE.

5 CONCLUSÕES

A aplicação das técnicas de *web crawler* e *web scrapper* em conjunto com expressões regulares mostram-se efetivas como mecanismo de busca de dados em *web pages*, tendo grande abrangência em sua aplicação. Os resultados do processo podem ser direcionados para buscas mais específicas, realizando uma configuração adequada no *web crawler*, a fim de obter os tópicos específicos de pesquisa. Para este estudo foi utilizado apenas um provedor de dados, o *Stack Overflow*, o que torna questionável a eficiência e as limitações encontradas. Isso porque, se o provedor de dados dispõe de uma API (ex. Github) para essa finalidade, o processo de obtenção dos dados se torna mais prático, sem muita necessidade de tratar textos e com quantidades de dados muito maiores.

Os resultados apresentam postagens de um período de seis meses, uma quantidade relativamente pequena considerando que o provedor contém postagens desde o ano de 2013. Em períodos mais antigos, as postagens possuem mais respostas e mais confiabilidade, visto que muitas destas já estão consideradas "resolvidas"e não são mais atualizadas, porém são conteúdos mais antigos. Assim, se o processo proposto fosse aplicado em um período de tempo mais antigo teríamos uma relação de categorias diferente, o que indica que o período escolhido também influencia nos resultados.

A partir da base extraída, percebe-se a necessidade de melhorias para poder utiliza-lá, e são propostos como trabalhos futuros atender novos provedores de dados para aumento do volume de dados, além de utilizar outros métodos para descrever dados da base, visto que há uma grande quantidade de características que podem ser agrupadas. Uma proposta para solucionar a barreira de limite de requisições seria através da utilização de outros provedores, alternando entre eles quando atingido o limite de requisições. Outro trabalho futuro que pode ser desenvolvido é a implementação de um método inteligente para a unificar categorias similares (ex. Python 3.0, Python, python-3.x). Isso tornaria a base mais consistente, e diminuiria os riscos de confundir um método de classificação, por exemplo. Ainda existem determinadas informações no texto que podem ser tratadas e substituídas, objetivando tornar a base mais consistente, como a substituição de números (ex. Celular, Telefone, CPF, CEP), nomes, termos em inglês e trechos de códigos não formatados. Para resolver essas questões, um método de pré-processamento inteligente é sugerido, como por exemplo o desenvolvido por (COPATTI, 2022).

Por fim, a base gerada atende às características necessárias para uma aplicação de PLN, porém a quantidade de dados é pequena para uma aplicação se comparado com outras bases como as utilizadas nos trabalhos de (WANG; BOWERS; FIKIS, 2017), (LOSIEWICZ; OARD; KOSTOFF, 2000) ou (CHOETKIERTIKUL *et al.*, 2019). Para que a base resultada apresente um volume maior, deve-se observar outros provedores de dados similares, além do *Stack Overflow*, como *GitHub*, *w3Schools*, páginas da web também relacionadas à desenvolvimento de software e linguagens de programação.

REFERÊNCIAS

ALHARBI, A.; ALGHAMDI, A. Execution of web crawlers for data collection. **International Journal of Web Data Mining**, v. 4, n. 2, p. 140–150, 2019.

ANNAMORADNEJAD, I.; HABIBI, J.; FAZLI, M. Multi-view approach to suggest moderation actions in community question answering sites. **Information Sciences**, v. 600, p. 144–154, 2022. ISSN 0020-0255. Disponível em: https://www.sciencedirect.com/science/article/pii/S0020025522003127.

BATISTA, G. E. de A. P. A. **Pré-processamento de Dados em Aprendizado de Máquina Supervisionado**. 2003. Tese (Doutorado) — Universidade de São Paulo, 2003. Disponível em: http://www.teses.usp.br/teses/disponiveis/55/55134/tde-06102003-160219.

BRITTO, L.; PACIFICO, L.; PACíFICO, L. Análise de sentimentos para revisões de aplicativos mobile em português brasileiro. *In*: . [*S.l.*: *s.n.*], 2020. p. 1080–1090.

CHOETKIERTIKUL, M. *et al.* A Deep Learning Model for Estimating Story Points. **IEEE Transactions on Software Engineering**, v. 45, n. 7, p. 637–656, jul. 2019. ISSN 1939-3520. Conference Name: IEEE Transactions on Software Engineering.

CHOI, H.; KIM, S. Configuration of web crawlers for data collection. **Journal of Web Data Mining**, v. 3, n. 2, p. 100–109, 2018.

COPATTI, B. S. Pré-processamento de dados de texto utilizando técnicas de aprendizado de máquina. Universidade Tecnológica Federal do Paraná, 2022.

DEVLIN, J. *et al.* Bert: Pre-training of deep bidirectional transformers for language understanding. **North American Association for Computational Linguistics (NAACL)**, 2019.

DOMINGOS, P.; RICHARDSON, M. Privacy preserving data mining. **Proceedings of the 6th ACM SIGKDD international conference on Knowledge discovery and data mining**, p. 71–79, 2000.

ELBASSUONI, A.; AL-ANSARI, N. Storage of data collected by web crawlers. **Journal of Big Data Management**, v. 5, n. 3, p. 220–227, 2019.

FÁVERO, E. M. D. B.; CASANOVA, D.; PIMENTEL, A. R. Se3m: A model for software effort estimation using pre-trained embedding models. **Information and Software Technology**, Elsevier, v. 147, p. 106886, 2022.

FERREIRA, S. **Guia Prático de HTML5**. Universo dos Livros Editora, 2013. ISBN 9788579303944. Disponível em: https://books.google.com.br/books?id=gvV0icqoRsgC.

FRIEDL, J. **Mastering Regular Expressions**. O'Reilly, 2002. (Nutshell Handbooks Series). ISBN 9780596002893. Disponível em: https://books.google.com.br/books?id=CT-MHZyNK2wC.

GONZALEZ, M.; LúCIA, V.; LIMA, V. Strube de. Recuperação de informação e expansão automática de consulta com thesaurus: uma avaliação. 01 2001.

GOYAL, A.; BHATIA, R. Generation of databases using web crawlers. **International Journal of Database Management Systems**, v. 8, n. 4, p. 1–9, 2016.

HAN, J.; KAMBER, M.; PEI, J. **Data Mining: Concepts and Techniques**. [*S.l.*]: Elsevier, 2011.

HIN, D. Stackoverflow vs kaggle: A study of developer discussions about data science. **ArXiv**, abs/2006.08334, 2020.

JARGAS, A. **EXPRESSOES REGULARES: UMA ABORDAGEM DIVERTIDA**. [*S.l.*]: NOVATEC, 2012.

JIANQIANG, Z.; XIAOLIN, G. Comparison research on text pre-processing methods on twitter sentiment analysis. **IEEE access**, IEEE, v. 5, p. 2870–2879, 2017.

JIAXIANG, H. Building domain specific lexicon based on tiktok comment dataset. 12 2020.

JOU, B.-H.; CHANG, C.-C.; LIN, C.-Y. The applications of natural language processing in healthcare. **Journal of medical systems**, v. 43, n. 12, p. 559, 2019.

KAUSAR, M. A.; DHAKA, V.; SINGH, S. Web crawler: A review. **International Journal of Computer Applications**, v. 63, p. 31–36, 02 2013.

KE, Z.; LIU, B. Continual learning of natural language processing tasks: A survey. 11 2022.

KENNEDY, K. **How to Clean Data in Natural Language Processing (NLP)**. [*S.l.*], 2021. Disponível em: https://python.plainenglish.io/nlp-data-cleansing-steps-6b4150cf87cf. Acesso em: 19 de outubro de 2022.

KHDER, M. Web scraping or web crawling: State of art, techniques, approaches and application. **International Journal of Advances in Soft Computing and its Applications**, v. 13, p. 145–168, 12 2021.

KUSHAL, D.; DEB, S.; MUKHERJEE, A. Definition of data collection objective. **International Journal of Data Science and Analytics**, v. 3, n. 4, p. 230–237, 2018.

LI, C. Xml parsing, sax/dom. *In*: ____. **Encyclopedia of Database Systems**. Boston, MA: Springer US, 2009. p. 3598–3601. ISBN 978-0-387-39940-9. Disponível em: https://doi.org/10.1007/978-0-387-39940-9_769.

LOSIEWICZ, P.; OARD, D.; KOSTOFF, R. Textual data mining to support science and technology management. **J. Intell. Inf. Syst.**, v. 15, p. 99–119, 09 2000.

MAATEN, L. van der; HINTON, G. Viualizing data using t-sne. **Journal of Machine Learning Research**, v. 9, p. 2579–2605, 11 2008.

MACHUCA, C. R.; GALLARDO, C.; TOASA, R. M. Twitter sentiment analysis on coronavirus: Machine learning approach. **Journal of Physics: Conference Series**, IOP Publishing, v. 1828, n. 1, p. 012104, feb 2021. Disponível em: https://dx.doi.org/10.1088/1742-6596/1828/1/012104.

MANNING, C. D.; RAGHAVAN, P.; SCHÜTZE, H. **Introduction to Information Retrieval**. [*S.l.*]: Cambridge University Press, 2014.

METWALLY, A.; AGRAWAL, D.; ABBADI, A. Challenges in web data extraction and integration. **ACM SIGMOD Record**, ACM, v. 34, n. 3, p. 13–18, 2005.

PERSSON, E. Evaluating tools and techniques for web scraping. USA, December 2019.

PINTO, S. C. S. Processamento de linguagem natural e extração de conhecimento. Universidade de Coimbra, 2015.

RAMACHANDRAN, D.; PARVATHI, R. Analysis of twitter specific preprocessing technique for tweets. **Procedia Computer Science**, Elsevier, v. 165, p. 245–251, 2019.

ROBBINS, A. **Effective Awk Programming: Text Processing and Pattern Matching**. O'Reilly Media, Incorporated, 2001. (A GNU manual). ISBN 9780596000707. Disponível em: https://books.google.com.br/books?id=blLqsSsuoUcC.

SINGH, T.; KUMARI, M. Role of text pre-processing in twitter sentiment analysis. **Procedia Computer Science**, Elsevier, v. 89, p. 549–554, 2016.

SINGRODIA, V.; MITRA, A.; PAUL, S. A review on web scrapping and its applications. *In*: **2019 International Conference on Computer Communication and Informatics (ICCCI)**. [*S.l.*: *s.n.*], 2019. p. 1–6.

SIRISURIYA, S. de S. A comparative study on web scraping. 2015.

WANG, Y.; BOWERS, A. J.; FIKIS, D. J. Automated text data mining analysis of five decades of educational leadership research literature: Probabilistic topic modeling of eaq articles from 1965 to 2014. **Educational Administration Quarterly**, v. 53, n. 2, p. 289–323, 2017.