

Multimodal Clustering

Bruno Francisco Martins da Silva

Advisor: Marco Antonio Casanova

Department of Computer Science, PUC-RIO, Rio de Janeiro, Brazil

{bsilva, casanova}@inf.puc-rio.br

Abstract

This paper addresses the *vector stream similarity search problem*, defined as: “Given a high dimensional vector q and a time interval T , find a ranked list of vectors, retrieved from a vector stream, that are similar to q and that were received in the time interval T .” The paper first introduces the project specifications and requirements, then the architecture of the project is detailed. After that, the technologies used in the project are shown, followed by a brief description of a proof-of-concept implementation of a classified ad retrieval tool that uses staged HNSW and a test routine. The paper concludes with the description of the user of this application and the documentation needed for this user to run the application.

Keywords: busca; indexação; similaridade; anúncios; Redis; JINA

1 Especificações do Projeto

1.1 Finalidade

Um anúncio consiste em uma descrição textual de um produto, com poucas imagens do produto, seu preço, condições de venda e os dados obrigatórios do vendedor. Uma plataforma de comércio online, ou simplesmente uma plataforma de comércio, é utilizada neste artigo a fim de especificar uma aplicação da Web onde os vendedores podem postar anúncios e os compradores podem pesquisar produtos e fechar transações. O volume de transações que uma plataforma de negociação deve suportar pode ser significativo. É comum que uma plataforma popular de produtos online normalmente processe milhares de anúncios por dia.

A principal motivação deste artigo se encontra no desafio de criar uma ferramenta de recuperação de anúncios que recebe como entrada um anúncio tem por objetivo obter uma lista de anúncios ranqueada baseada na similaridade entre os mesmos. A similaridade, nestes casos pode ser computada utilizando a descrição textual, um conjunto de imagens, o preço, entre outras propriedades dos anúncios. Este artigo, considera o cenário em que novos anúncios são continuamente incluídos da plataforma, criando assim uma *stream* de anúncios.

A construção dessa ferramenta de recuperação de anúncios, neste cenário, irá enfrentar uma grande dificuldade. Como o conjunto de anúncios é dinâmico, no sentido de que os vendedores continuamente criam novos anúncios, em um ritmo acelerado, e muitos desses anúncios têm um tempo de vida muito curto devido ou ao produto ser efetivamente vendido, ou simplesmente porque o vendedor decidiu remover o anúncio, ou porque o anúncio simplesmente se tornou obsoleto por alguma razão. Posto isso, pode-se modelar este cenário como um processo de recuperação que acontece sobre a *stream*, limitando em algum ponto no passado, como uma janela de tempo.

De um ponto de vista de alto nível, este cenário requer resolver o problema de busca por similaridade em um vetor de *stream*, definido por: “Dado um vetor q (de alta dimensão) e um intervalo de tempo T , encontre uma lista de vetores ranqueados, recuperados da *stream* de vetores, que são similares a q e que foram recuperados no intervalo de tempo T ”.

1.2 Escopo

Posto isso, as principais contribuições deste artigo são: apresentar uma breve descrição de uma implementação prova de conceito de uma ferramenta de recuperação de anúncios baseado no JINA ¹, um *framework* para construir aplicações que aproveitam as *engines* de busca neural, com o objetivo de controlar o processo de recuperação, e na implementação do “HNSW” (Malkov and Yashunin (2020)), na parte de indexação dos vetores.

¹<https://jina.ai/>

1.3 Requisitos

1.3.1 Requisitos Funcionais

Os requisitos funcionais do sistema foram obtidos baseado na finalidade e no escopo do projeto.

- **RF01** - A ferramenta deve permitir que o usuário carregue um arquivo de texto no formato CSV de qualquer diretório do seu computador;
- **RF02** - A ferramenta deve permitir que o usuário acesse o sistema de qualquer navegador do seu computador;
- **RF03** - A ferramenta deve permitir que o usuário possa pesquisar por qualquer conjunto de palavras-chave que ele desejar;
- **RF04** - A ferramenta deve permitir que o usuário possa configurar a quantidade de resultados que ele deseja analisar;
- **RF05** - A ferramenta deve permitir que o usuário possa repetir a quantidade de buscas quantas vezes o usuário desejar;
- **RF06** - A ferramenta deve permitir que o usuário possa alterar a consulta por palavras-chave que ele realizou sem a necessidade de reiniciar a ferramenta;
- **RF07** - A ferramenta deve permitir que o usuário possa alterar o conjunto de dados que ele deseja analisar sem a necessidade de reiniciar a ferramenta;

1.3.2 Requisitos Não Funcionais

Dentro dos requisitos não-funcionais que a ferramenta deve apresentar tem-se:

- **RNF01 - Performance:** Refere-se ao tempo de resposta do uso das funcionalidades do sistema;
- **RNF02 - Usabilidade:** Refere-se a interface ser amigável para o usuário, permitindo com que o mesmo seja encorajado a utilizar o sistema;
- **RNF03 - Compatibilidade:** Refere-se ao sistema ser compatível com os sistemas MacOS ou Linux;
- **RNF04 - Instalação:** Refere-se ao sistema ser instalado de forma amigável no computador do usuário;

2 Projeto

2.1 Visão Geral

Para contruir uma ferramenta capaz de resolver o problema de busca por similaridade em um vetor de *stream*, uma ferramenta *Web* foi desenvolvida em três blocos distintos, consistindo de uma estrutura principal e duas estruturas auxiliares. A estrutura principal é responsável por administrar o fluxo de tarefas entre as estruturas auxiliares, além de possuir uma interface gráfica que permite tanto o carregamento da base de dados de anúncios que o usuário deseja analisar (*indexing*) quanto a exploração dessa base de dados através de um consultas por palavras chave (*seraching*). É importante notar que o retorno destas consultas é configurável, ou seja, o usuário pode predefinir a quantidade de resultados que ele gostaria de analisar. Já as estruturas auxiliares são divididas em: um encodificador de texto e um banco de dados.

O encodificador de texto tem como finalidade criar os índices dos arquivos que são fornecidos pelo usuário. O objetivo destes índices é fazer com que os dados dos anúncios sejam codificados para um formato que seja possível aplicar uma métrica de similaridade, comparando os últimos com a consulta que o usuário está executando na aplicação através de um valor. O processo de criação destes índices ocorre quando o usuário insere um arquivo de texto, no formato CSV, na aplicação. Esta, ativa a parte de *indexing*, preenchendo um vetor com todos os registros encontrados na base enviada.

Com os vetores preenchidos, o encodificador de texto faz um tratamento no tamanho dos arquivos, aplicando o modelo `sentence-transformers_paraphrase-multilingual-mpnet-base-v2`², responsável por retornar vetores de 768 dimensões (*embeddings*), e estes vetores que são utilizados para a geração dos índices. Outros fatores que também são considerados além do tamanho dos *embeddings* são: a métrica que determina o cálculo da distância entre os vetores, o número inicial de vetores e o tipo do índice que está sendo gerado. A métrica utilizada para a construção do vetor de similaridades foi a “HNSW”.

O banco de dados é o responsável por armazenar e manipular os índices que foram criados na etapa anterior. Seu intuito é exibir parâmetros específicos sobre cada índice, como o tempo de criação dos mesmos, a porcentagem de conclusão do processo de indexação, o número de documentos que foram indexados, dentre outros.

²<https://huggingface.co/sentence-transformers/paraphrase-multilingual-mpnet-base-v2>

2.2 Arquitetura

A figura 1 descreve com mais detalhes como se comporta a arquitetura do projeto. A ferramenta possui a pasta “app” contendo todas as informações relativas a aplicação e arquivos relacionados a infraestrutura necessária para executá-la, como as configurações dos *containers*, as variáveis de ambiente e as bibliotecas requeridas que devem estar presentes no computador do usuário.

Dentro da pasta “app” observa-se diversos submódulos. Detalhando cada um deles, temos: o submódulo “dl_models”, responsável por armazenar os modelos utilizados para tratamento dos dados de anúncios enviados pelo usuário, convertendo os mesmos para o de vetor de 768 dimensões. Vale notar que no contexto atual apenas um modelo foi utilizado. O submódulo “input” é responsável por armazenar os dados que são inseridos na aplicação pelo usuário. Já o submódulo “output” serviu como base para a execução de um teste que envolveu pré-processar os vetores de *embeddings* utilizando o Google Colabs ³, a fim de investigar a performance desta etapa.

O submódulo “routers” apresenta as rotas que a aplicação pode tomar quando ela já está funcionando, podendo esta acessar a parte de indexação ou busca. Já o submódulo “services” exibe os serviços disponíveis para a aplicação, o arquivo *text* representa o encodificador de texto, já o arquivo *performance_analysis* tem como finalidade avaliar a performance dos processos de indexação e busca. Por fim, temos três arquivos: O *helper* que efetivamente possui o método de criação dos índices. Este, é chamado dentro do encodificador de texto, melhorando assim a legibilidade e diminuindo o acoplamento do código; O arquivo *main* é o responsável por levantar a aplicação *Web*; O *shared_context* é o responsável por fazer a conexão com o banco de dados Redis. Seus métodos são utilizados por toda a extensão da ferramenta, visto que o banco de dados tem o papel fundamental na manutenção da estrutura da mesma.

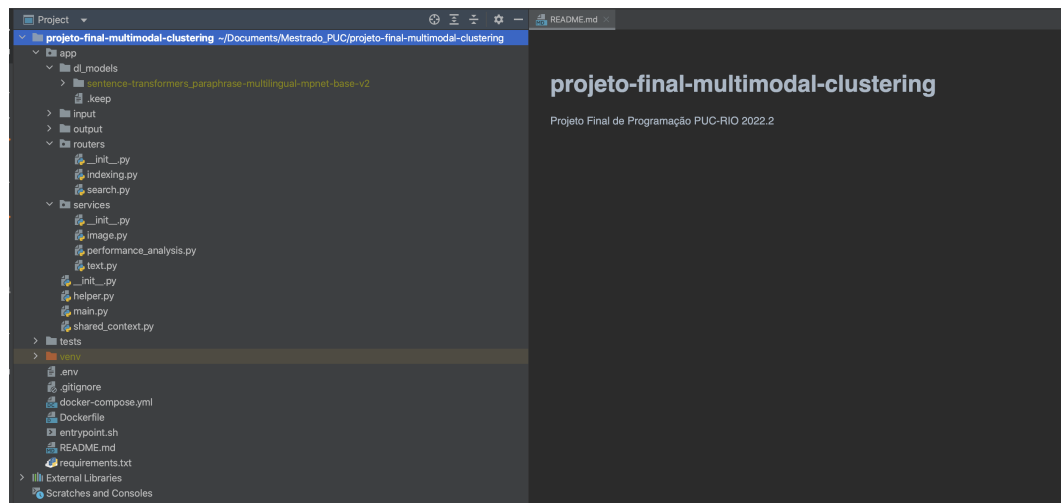


Figure 1: Arquitetura do Projeto

Já a Figura 2 descreve o diagrama UML que representa como esta arquitetura está estruturada, exibindo com detalhes seus componentes e as conexões entre eles. Além de indicar quais componentes estão relacionados com as estruturas principais mencionadas anteriormente, em que o bloco da FastAPI corresponde a estrutura principal, o bloco do *text_encoder* corresponde ao encodificador de texto e o bloco do Redis corresponde ao Banco de Dados.

³<https://colab.research.google.com>

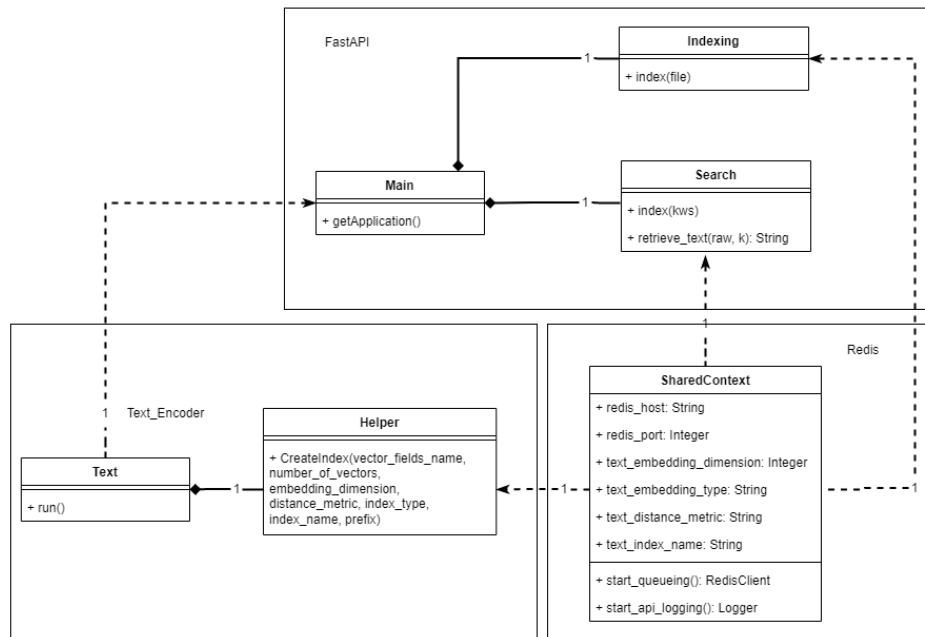


Figure 2: Diagrama UML da Arquitetura

3 Tecnologias

3.1 Python

Python ⁴ é uma linguagem de programação de alto nível, dinâmica, interpretada, modular, multiplataforma e orientada a objetos. Por ser uma linguagem de sintaxe relativamente amigável, ganhou popularidade entre profissionais da indústria tecnológica que não são especificamente programadores, como engenheiros, matemáticos, cientistas de dados, pesquisadores e outros. Um de seus maiores atrativos é possuir um grande número de bibliotecas, nativas e de terceiros, tornando-a muito difundida e útil em uma grande variedade de setores dentro de desenvolvimento web, e também em áreas como análise de dados, machine learning e IA.

3.2 Numpy

A biblioteca Numpy ⁵ fornece um grande conjunto de funções e operações de biblioteca que ajudam os programadores a executar facilmente cálculos numéricos. Esses tipos de cálculos numéricos são amplamente utilizados em tarefas como: Modelos de Machine Learning, Processamento de Imagem e Computação Gráfica e Tarefas matemáticas.

3.3 Pandas

A biblioteca Pandas ⁶ é uma biblioteca Python para análise de dados. Ela possui código aberto e uso gratuito. O Pandas é construído com base em duas bibliotecas mais famosas do Python: matplotlib para visualização de dados e NumPy para operações matemáticas, sendo uma união dessas bibliotecas, e permitindo que sejam acessados muitos dos métodos de matplotlib e NumPy com menos esforço. Esta biblioteca é conhecida por sua alta produtividade e desempenho e sua popularidade deriva do fato da importação e a análise de dados ser muito mais amigável.

3.4 Docker

Docker ⁷ é uma tecnologia de containerização para criação e uso de máquinas Linux (*containers*). Docker é uma ferramenta parecida com uma máquina virtual extremamente leve, mas não se trata de fato de uma

⁴<https://www.python.org>

⁵<https://numpy.org>

⁶<https://pandas.pydata.org>

⁷<https://www.docker.com>

máquina virtual. Ele utiliza (*containers*) que possuem uma arquitetura diferente, permitindo maior portabilidade e eficiência. O container exclui a virtualização e muda o processo para o Docker. Além disso, os *containers* oferecem maior flexibilidade para a criação, implantação, cópia e migração de um *container* de um ambiente para outro.

3.5 FastAPI

FastAPI ⁸ é um *framework Web* moderno e de alta performance para construir API's com Python. Suas principais características são: A velocidade, API's desenvolvidas com o FastAPI possuem uma alta performance, ao ponto de serem comparadas com API's desenvolvidas com tecnologias mais consolidadas; A intuitividade, o código fonte do *framework* foi inteiramente desenvolvido utilizando o recurso de *type hints* do Python, isso possibilita que se gaste menos tempo debugando o código; A leveza, pois, ele foi inteiramente pensado para ser amigável aos usuários, fazendo assim com que se gaste bem menos tempo lendo a documentação; Por fim, tem-se a robustez, em que o código desenvolvido já está pronto para produção, assim não é necessário fazer nenhuma alteração para então colocar as aplicações desenvolvidas no ar, além de que o FastAPI pode gerar a documentação de forma automática.

3.6 Redis

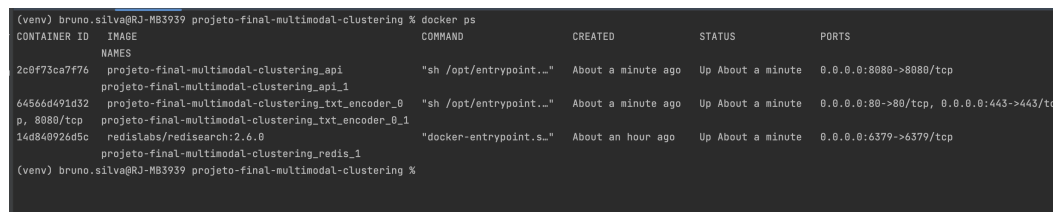
Redis ⁹ é um banco de dados relacional focado em alto desempenho. Sua principal característica é a agilidade com que acessa e armazena informações, muito por conta de sua estrutura de funcionamento. Ele oferece um conjunto de estruturas versáteis de dados na memória que permite a fácil criação de várias aplicações personalizadas. Os principais casos de uso do Redis incluem cache, gerenciamento de sessões, PUB/SUB e classificações.

3.7 Código Fonte

A estrutura desta aplicação foi desenvolvida em Python, utilizando diversos recursos de bibliotecas Numpy e Pandas para efetuar as operações matemáticas e a leitura de dados, respectivamente. Estas operações matemáticas estão relacionadas com a métrica e o modelo matemático que são utilizado para calcular as distâncias entre os registros nos vetores, durante o processo de criação dos índices. Já a leitura de dados ocorre durante o carregamento da base de dados de anúncios feita pelo usuário na interface gráfica. Nesta etapa, enquanto nenhum dado é carregado a aplicação mantém uma fila vazia que fica esperando algum dado ser inserido. Quando um arquivo em CSV - formato de texto, é inserido pelo usuário, a fila começa a ser preenchida e o encodificador de texto começa a remover os arquivos da fila e indexa-los.

FastAPI e Docker foram utilizados para servir de corpo para a aplicação. Os *containers* do Docker foram os responsáveis por estruturar tanto a estrutura principal, que contém a FastAPI, quanto as estruturas auxiliares, contendo o banco de dados do Redis e o encodificador de texto. Percebe-se então que três *containers* são criados para sustentar toda a aplicação. O *container* que executa o processo da FastAPI é responsável por manter a interface gráfica *Web* funcionando, garantindo assim a interação do usuário com o sistema. Já o *container* que executa o processo do Redis é responsável por manter o banco de dados ativo, recebendo tanto as requisições de PUB/SUB quanto a escrita dos índices, garantindo todas as propriedades de um banco relacional. É importante notar que desligar estes *containers* não faz com que os dados sejam perdidos, isto ocorre apenas se eles forem destruídos. A figura 3 representa a aplicação em funcionamento, exibindo os *containers* ativos.

Logo, através das tecnologias mencionadas anteriormente foi desenvolvida a ferramenta apresentada neste artigo. Seu código fonte se encontra disponível no GitHub no *link*: <https://github.com/BrunoFMSilva/projeto-final-multimodal-clustering>, apresentando comentários nas principais funções e módulos do mesmo.



CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
2c0f73ca7f76	projeto-final-multimodal-clustering_api	"sh /opt/entrypoint..."	About a minute ago	Up About a minute	0.0.0.0:8080->8080/tcp
64566d491d32	projeto-final-multimodal-clustering_txt_encoder_0	"sh /opt/entrypoint..."	About a minute ago	Up About a minute	0.0.0.0:80->80/tcp, 0.0.0.0:443->443/tcp
p, 8880/tcp	projeto-final-multimodal-clustering_txt_encoder_0_1	"docker-entrypoint.s..."	About an hour ago	Up About a minute	0.0.0.0:6379->6379/tcp
14d840926d5c	redislabs/redisearch:2.6.0				
	projeto-final-multimodal-clustering_redis_1				

Figure 3: *Containers* do Docker em execução

⁸<https://fastapi.tiangolo.com>

⁹<https://redis.io>

4 Roteiro de Testes

Foi desenvolvida uma rotina de testes, utilizando a biblioteca *unittest*¹⁰, para avaliar o comportamento das funções principais presentes nos arquivos *shared_context.py* e *helper.py*. No primeiro arquivo, os testes tiveram o intuito de verificar se as conexões com o Redis conseguiram ser efetuadas e obtidas com sucesso. Além de verificar se o modelo utilizado pelo encodificador de texto para fazer o tratamento dos registros estava sendo gerado no formato correto. Já no segundo arquivo foi verificado se a criação dos índices estava respeitando o formato esperado.

Estes arquivos foram escolhidos como base central na rotina de testes devido a sua importância geral no ambiente da aplicação, pois eles servem de base para todos outros componentes. A figura 4 exibe a localização destes testes. Cada um deles possui seu devido comentário e para executá-los basta baixar o projeto no GitHub, disponibilizado anteriormente, e roda-los no terminal, em uma máquina que possua sistema operacional Linux ou MacOS, que possua a linguagem de programação python instalada.

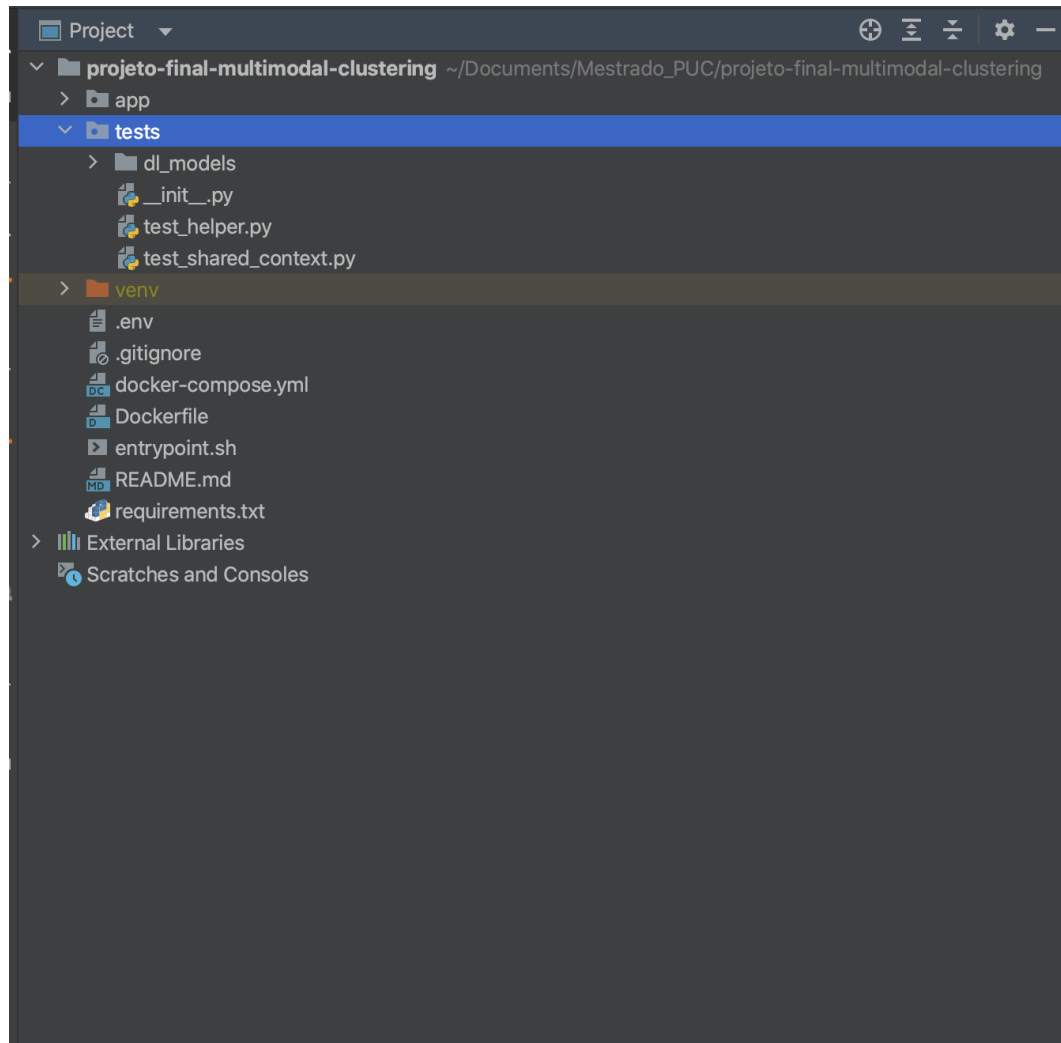


Figure 4: Arquivos de Teste

5 Documentação

O usuário desta ferramenta é o analista de dados de companhias que lidam diretamente com um grande fluxo de anúncios diariamente. O analista, por precisar entender a natureza e o comportamento dos dados que estão em trânsito neste fluxo contínuo, executa diversos processos a fim de identificar as propriedades destes anúncios. Entretanto, devido ao enorme volume de anúncios presente neste fluxo, torna-se extremamente exaustivo efetuar estas análises em tempo real. Logo, uma ferramenta capaz de capturar uma grande quantidade de anúncios e permitir com que os mesmos sejam observados em um instante de tempo fixo torna-se extremamente útil.

¹⁰<https://docs.python.org/3/library/unittest.html>

Posto isso, a ferramenta apresentada neste artigo tem como objetivo facilitar o processo de análise de dados que os analistas precisam efetuar no seu dia a dia. Este processo de facilitação começa na busca por anúncios ou conjuntos de anúncios específicos. Na ferramenta, a busca ocorre por palavras-chaves relacionadas ao escopo do próprio anúncio, sendo assim mais amigável. Além disso, como é possível configurar a quantidade de resultados que a ferramenta retorna ao usuário, ele pode buscar por anúncios específicos, verificando a existência do mesmo na base de dados original ou fazer comparações com grandes volumes de dados que possuem maior similaridade com o anúncio desejado. Por exemplo, pode-se verificar se algum modelo de celular está sendo anunciado na plataforma original. A ferramenta retornaria os anúncios mais similares ao solicitado, incluindo sua descrição. Com estas descrições, torna-se possível verificar se aquele conjunto de dados está coerente ou apresenta alguma irregularidade. O que, em casos extremos, pode até significar um indicativo de fraude.

Outro fator interessante do uso da ferramenta apresentada como mecanismo de análise é o fato da identificação de anúncios duplicados na base de dados original, mas esta duplicação pode ocorrer tanto de inconsistências com a própria base de dados ou pode-se identificar vendedores que estão criando múltiplos anúncios que possuem o mesmo item e a mesma descrição, a fim de otimizar o tempo de venda daquele produto.

Por fim, esta ferramenta também pode ser usada como filtro de anúncios, pois ao se buscar por palavras-chave específicas pode-se identificar anúncios que estão sendo colocados na base com nomes impróprios ou com descrições impróprias, o que permite a exclusão destes anúncios e até o banimento desses usuários que estão apresentando este comportamento inadequado, através de outros processos fora da ferramenta.

Para utilizar esta ferramenta é necessário baixá-la do GitHub, através do link mencionado na seção anterior. Com o projeto adquirido, o primeiro passo é baixar as bibliotecas do arquivo *requirements.txt*. Com essas bibliotecas instaladas deve-se abrir a pasta raiz do projeto e rodar os seguintes comandos:

1. `docker-compose build`
2. `docker-compose up`

Estes, serão os responsáveis por levantar os *containers* do Docker, que, após alguns segundos disponibilizaram a aplicação Web no link: <http://localhost:8080/docs>, como pode ser observado na Figura 5.

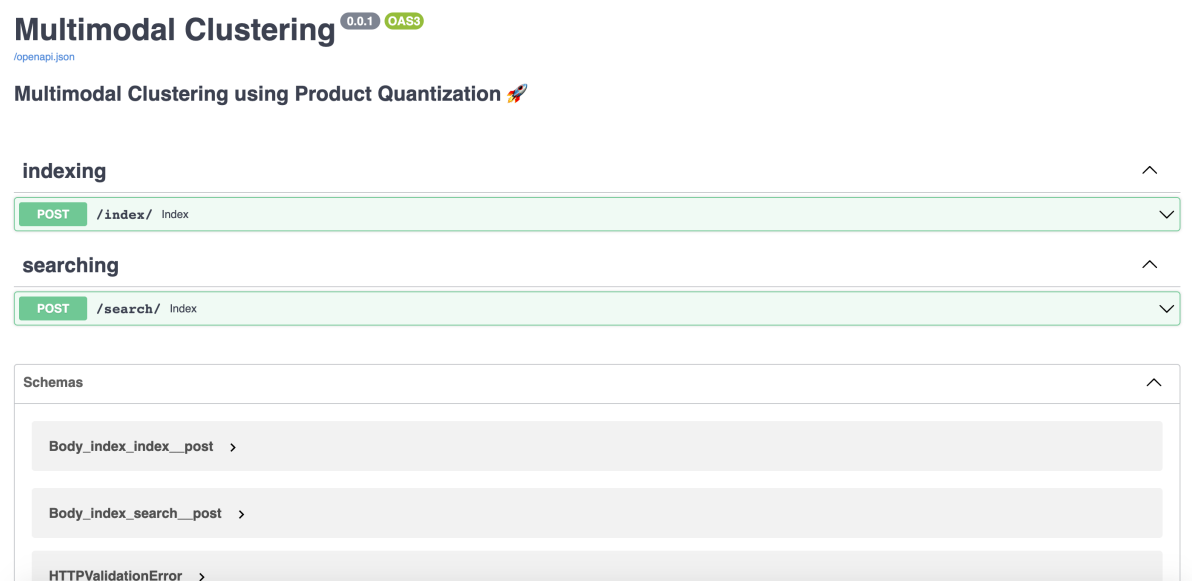


Figure 5: Interface Gráfica

Com a interface gráfica funcionando, o usuário pode interagir com as seções de *indexing* e *searching*. Na seção de indexing, o usuário deve clicar na opção *try it out* (Figura 6), depois inserir o arquivo CSV contendo os dados que serão indexados (Figura 7) e por fim clicar no botão de execute para iniciar o processo (Figura 8). Contudo, esses arquivos devem possuir uma formatação específica, que é demonstrado na Figura 9.

indexing

POST /index/ Index

Parameters

Name	Description
skip integer (query)	Default value : 0

Request body required

multipart/form-data

file * required
string(\$binary)

Try it out

Figure 6: Exemplo de Indexação

indexing

POST /index/ Index

Parameters

Name	Description
skip integer (query)	0

Request body required

multipart/form-data

file * required
string(\$binary)

Escolher arquivo Nenhum arquivo escolhido

Execute

Figure 7: Exemplo de Indexação

indexing

POST /index/ Index

Parameters

Name	Description
skip integer (query)	0

Request body required

multipart/form-data

file * required
string(\$binary)

Escolher arquivo electronics_20220615_sample_1k.csv

Execute

Figure 8: Exemplo de Indexação


```

1 id,image,caption
2 1045201614,https://img.olx.com.br/images/58/584201779242977.jpg,"iPhone 11 128 gigas * Caixa com acessórios *128 gigas * Nota Fiscal
comprado no Brasil * Cabo Original. * Garantia Apple até Junho de 2022 * Saúde da bateria 87% * Perfeito estado, sem arranhões ou
marcas de uso * Aparelho muito pouco usado * Tudo funciona perfeitamente ? Não aceito OLX Pay Troco apenas pelo iPhone 12 mini de
128 gigas e nas mesmas condições que o meu aparelho "
3 1045201774,https://img.olx.com.br/images/58/585201772852955.jpg,"Top snapdragon 865 Samsung S20 FE 5G 128GB 6GB_Ram Versão especial
**** VIX SMARTPHONES OS MELHORES PREÇOS E PRODUTOS DO MERCADO ! **** Top snapdragon 865 Samsung S20 FE 128GB 6GB_Ram Versão
especial com INTERNET 5G E PROCESSADOR SNAPDRAGON 865!!! Samsung S20 FE 128GB CAMERA COM ZOOM ÓTICO DE 3X CAMERA FRONTAL DE 32MP,
memória interna de 128GB e 6GB ram. Produto na caixa LACRADA, com NOTA FISCAL e GARANTIA do fabricante! VALOR: 1999 NO DINHEIRO OU
10X DE 220 NO CARTÃO SOMENTE VENDAS!!! NÃO VENDO NO CARNE!!! SOMENTE VENDAS!!! NÃO VENDO NO CARNE!!! ***** TAMBÉM TEMOS:
-> Nokia 5.4 128GB Android 10 por R$ 999 ou 10x de 110,99 -> FONE sem FIOS Samsung BUDS 2 por R$ 399 ou 10x de 49,99 ->
Samsung A13 128GB por R$ 1299 ou 10x de 145,99 -> Samsung A52S 128GB 6GB por R$1799 ou 10x de 199,99 -> Samsung S20 FE 5G 128GB
6GB por R$1999 ou 10x de 210,99 -> Motorola G22 4GB 128GB R$ 1299 ou 10x de 149,99 -> Motorola G50 4GB 128GB R$ 1499 ou 10x de
169,99 -> Motorola G71 6GB 128GB R$ 1899 ou 10x de 209,99 -> AMAZFIT STRATO5 com GPS R$ 699 ou 10x de 79 smart watch ->
Xiami MI BAND 6 ORIGINAL R$ 299 ou 10x de 31 RELÓGIO INTELIGENTE -> Xiami MI BOX S R$ 499 ou 10x de 59,90 -> Smart TV 55 QLED B65
modelo 2022 R$ 3499 ou 10x de 378 *****"
4 1045198615,https://img.olx.com.br/images/58/585209771855833.jpg,Celular Vendo moto e7 Power sem trinco sem nada biometria tudo
funcionando normal com caixa e cabo de carregar com película 3D
5 1045200625,https://img.olx.com.br/images/58/589238416721301.jpg,iPhone XR 128gb iPhone XR preto 128gb Aceito cartão Acompanha caixa
Carregador Nota fiscal
6 1045198866,https://img.olx.com.br/images/58/588204779666296.jpg,Motorola usado em bom estado Moto e6s 32gb funcionando tudo
Carregador incluso
7 1045202769,https://img.olx.com.br/images/58/580277296930478.jpg,"Relógio Galaxy Active Vendo Watch em perfeito estado. Usado
pouquíssimas vezes? sem marca alguma de uso e funcionando perfeitamente. Bateria também em excelente estado. **Acompanha pulseira
preta, cinza, branca e rosa."
8 1045206668,https://img.olx.com.br/images/58/583217294726628.jpg,"Galaxy tab s6 lite Usado por apenas 4 meses, A bateria tem 7.040 mAh
com duração de aproximadamente 4/5 dias, tela de 10,4 polegadas, câmeras mediana, 64GB de armazenamento e processador bom para a
proposta dele."
9 1045221789,https://img.olx.com.br/images/58/589265534726015.jpg,A30S 400 apenas sem choro A30S 64 Gb
10 1045219130,https://img.olx.com.br/images/58/582240771294075.jpg,Vendo esse a52 novo com todos acessórios Vendo esse a52 novo com
todos acessórios mais informações no meu tem pouco dias saiu da loja somente interessados sem queima vela agradeço
11 1045221215,https://img.olx.com.br/images/58/584201410081248.jpg,"iPhone 7 128gb Vende se iPhone 7 128gb Únicos detalhes são, áudio e
microfone (funcionam com fone de ouvidos) chip não funciona. Demais tudo ok, câmeras, touch, icloud, apps, tudo funciona normal 600$
pra sair hoje Pode ser usado pra retirar peças."
12 1045345960,https://img.olx.com.br/images/58/589239295085830.jpg,"Tablet Samsung Galaxy Tab S7 Tela 11 256 Gb 8gb Ram Apenas venda, o
interessado pode ver pessoalmente comigo em Jacareí, não vendo por OLX PAY, Mercado livre, apenas presencialmente em minha
residência, não troco por nada, aparelho foi comprado novo possui nota fiscal em meu nome porem utilizei no máximo umas 10 vezes,
novíssimo sem detalhe possui Capa Book Cover Original Samsung e película de vidro. Interessados podem chamar no whatsapp
12991227431 Rodrigo, posso passar cartão se preferir, golpistas e desocupados nem percam tempo! Tablet Samsung Galaxy Tab S7, Leve,
fino, 256GB, 8GB RAM, Tela Imersiva de 11", Tablet S Pen, S Pen, Tablet Dual Cam, Câmera Traseira 13MP+5MP, Câmera frontal de 8MP

```

Figure 9: Formatação do Arquivo CSV

Após isso, o usuário deve esperar pelo fim do processo de indexação. Com este concluído pode-se iniciar a etapa de busca, logo deve-se acessar a seção de *searching*. Repetindo o processo da seção anterior deve-se clicar na opção *try it out* (Figura 10), após isso, na caixa “kws” deve-se escrever a consulta por palavra-chave que se deseja efetuar, podendo configurar a quantidade de resultados obtidos na caixa “k” (Figura 11). Por fim, deve-se clicar na opção *execute* e a aplicação retornará os resultados esperados, incluindo a chave de identificação do anúncio, sua descrição e seu *score*, que é a nota baseada em quão similar o resultado é com a consulta original.

The screenshot shows a web interface titled "searching". At the top, there's a "POST /search/ Index" header. Below it, a "Parameters" section contains two input fields: "kws" (string, query) with the value "kws" and "k" (integer, query) with the value "5". A red arrow points to a "Try it out" button. Below the parameters, there's a "Request body" section with a dropdown menu set to "multipart/form-data". At the bottom, there's a field for "img" (string, \$binary).

Figure 10: Exemplo de Busca

searching

POST /search/ Index

Parameters

Cancel Reset

Name	Description
kws string (query)	<input type="text" value="kws"/>
k integer (query)	<input type="text" value="5"/>

Request body

multipart/form-data

img
string(\$binary)

Nenhum arquivo escolhido

☒ Send empty value

Execute

Figure 11: Exemplo de Busca

searching

POST /search/ Index

Parameters

Cancel Reset

Name	Description
kws string (query)	<input type="text" value="Samsung A3"/>
k integer (query)	<input type="text" value="5"/>

Request body

multipart/form-data

img
string(\$binary)

Nenhum arquivo escolhido

☒ Send empty value

Execute Clear

Figure 12: Exemplo de Busca

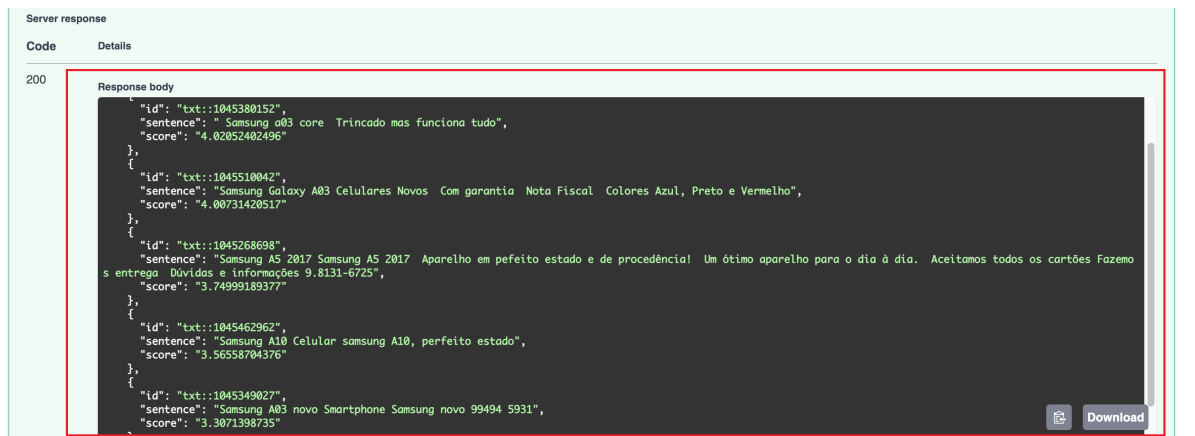


Figure 13: Exemplo de Busca

6 Conclusão

Portanto, o artigo apresenta uma ferramenta que é capaz de obter dados de anúncios que são disponibilizados pelo usuário, no formato CSV, aplicar um processamento sobre esses dados, aplicando um tratamento no tamanho dos registros presentes nos dados e retornando um *json* para o sistema que, a partir deste *json*, calcula os índices desses anúncios utilizando a implementação do “HNSW” para efetuar o cálculo de indexação dos vetores. Com esses índices definidos, a aplicação permite que o usuário faça pesquisas por palavras-chave na base de dados de anúncios utilizando a interface gráfica.

Além disso, a interface gráfica apresenta o parâmetro da quantidade de anúncios que deve ser buscada, permitindo ao usuário certa interatividade e especificidade com a mesma, visto que a ferramenta pode ser utilizada para diferentes propósitos de análise de dados, como: um maneira de conferir se algum anúncio específico está presente na base, filtrar anúncios impróprios, comparar anúncios diversos que estejam relacionados com a mesma categoria, dentre outras. Servindo assim, como uma ferramenta complementar no dia a dia de um analista de dados que está inserido no contexto de anúncios.

References

Malkov, Y. A., & Yashunin, D. A. (2020, apr). Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE Trans. Pattern Anal. Mach. Intell.*, 42(4), 824–836. Retrieved from <https://doi.org/10.1109/TPAMI.2018.2889473> DOI: 10.1109/TPAMI.2018.2889473