UNIVERSIDADE FEDERAL DE PERNAMBUCO – UFPE

CENTRO DE INFORMÁTICA – CIN

CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

ARTHUR FRADE DE ARAÚJO

DESENVOLVIMENTO DE SISTEMA DE RECOMENDAÇÃO DE *ANIMES*: UMA APLICAÇÃO DA SIMILARIDADE POR COSSENOS

RECIFE

2024

ARTHUR FRADE DE ARAÚJO

DESENVOLVIMENTO DE SISTEMA DE RECOMENDAÇÃO DE *ANIMES*: UMA APLICAÇÃO DA SIMILARIDADE POR COSSENOS

Trabalho de Graduação apresentado ao Curso de Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do título de bacharel em Ciência da Computação.

Orientador: Sérgio Ricardo de Melo Queiroz

Áreas: Engenharia de *Software* e *Machine Learning*

RECIFE

2024

Ficha de identificação da obra elaborada pelo autor,
através do programa de geração automática do SIB/UFPE

de Araújo, Arthur Frade.
Desenvolvimento de sistema de recomendação de animes: Uma aplicação da similaridade por cossenos / Arthur Frade de Araújo. - Recife, 2024.
26 p. : il.

Orientador(a): Sergio Ricardo de Melo Queiroz
Cooorientador(a): Vinícius Cardoso Garcia
Trabalho de Conclusão de Curso (Graduação) - Universidade Federal de Pernambuco, Centro de Informática, Ciências da Computação - Bacharelado, 2024.
Inclui apêndices.

1. Aprendizagem de Máquina. 2. Filtragem Colaborativa. 3. Similaridade por Cossenos. 4. Engenharia de Software. I. Queiroz, Sergio Ricardo de Melo. (Orientação). II. Garcia, Vinícius Cardoso. (Coorientação). IV. Título.

000 CDD (22.ed.)

ARTHUR FRADE DE ARAÚJO

DESENVOLVIMENTO DE SISTEMA DE RECOMENDAÇÃO DE *ANIMES*: UMA APLICAÇÃO DA SIMILARIDADE POR COSSENOS

Trabalho de Graduação apresentado ao Curso de Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do título de bacharel em Ciência da Computação.

Aprovado em: 31/07/2024

BANCA EXAMINADORA

_____

Prof. Sergio Ricardo de Melo Queiroz (Orientador)
Universidade Federal de Pernambuco

_____

Prof. Vinícius Cardoso Garcia (Examinador Interno)
Universidade Federal de Pernambuco

RESUMO

A sociedade contemporânea vive em uma era de abundância inesgotável de conteúdo. Nesse sentido, as plataformas de mídia buscam recomendar material que atraia o usuário. Este trabalho desenvolveu um sistema de recomendação como forma de imersão nessa disciplina. O nicho escolhido para atuação prática foram os *animes*. Foi implementada uma matriz de semelhança entre *animes* utilizando similaridade por cossenos e foram criados vetores de *animes*, cujas dimensões são avalições dadas por usuários da plataforma *My Anime List* (MAL), utilizados no cálculo de similaridade. Para disponibilizar essa matriz de uma forma acessível a usuários finais, criou-se um produto digital composto por uma API (*Application Programming Interface*) de *back-end* e uma aplicação que se integra com o MAL para coletar insumos para o modelo. Por fim foi implementada uma interface de navegação web.

Palavras-Chave: Aprendizagem de Máquina, Filtragem Colaborativa, Similaridade por Cossenos, Engenharia de *Software*, *MLOps*.

ABSTRACT

Contemporary society lives in an era of inexhaustible content abundance. In this context, media platforms seek to recommend material that attracts users. This work developed a recommendation system as a way of immersing in this discipline. The chosen niche for practical application was anime. A similarity matrix between anime was implemented using cosine similarity, and anime vectors were created, with dimensions being ratings given by users of the My Anime List (MAL) platform, to be used in the similarity calculation. To make this matrix accessible to end-users, a back-end API (Application Programming Interface) was created, along with an application that integrates with MAL to collect inputs for the model. Finally, a web navigation interface was implemented.

Keywords: Machine Learning, Collaborative Filtering, Cosine Similarity, Software Engineering, MLOps.

SUMÁRIO

1. INTRODUÇÃO

O filósofo Ortega y Gasset certa vez afirmou que um indivíduo é composto de seu próprio ser e de sua circunstância, no sentido de que é impossível para o "ser" estar dissociado de seu meio [20]. Em seu pensamento, para que uma pessoa consiga salvar a si mesma, ela precisa redimir a sua circunstância. Essa redenção, por sua vez, só pode acontecer por meio de um ponto de partida: o conhecimento. Nesse sentido, se o homem não conhece a sua própria circunstância, nunca poderá ser verdadeiramente livre ou até mesmo feliz.

A sociedade moderna tem vivido uma era de abundância de conteúdo, pois a informação corre por todos os lados e é impossível para alguém consumir tudo o que está disponível. Em virtude disso, grandes conglomerados tecnológicos e de mídia buscam apresentar informações que atraiam a atenção dos usuários [7]. Para tanto, diversas técnicas de recomendação são utilizadas nas mais diversas aplicações: redes sociais, plataformas de mídia, *streaming* etc. Tais ferramentas moldam os padrões de consumo da sociedade moderna, tornando relevante ao profissional de tecnologia conhecer o funcionamento delas para que possa viver a realidade de forma ativa e crítica. É justamente essa necessidade que justifica a realização deste trabalho.

Posto isso, o objetivo desse trabalho é construir um sistema de recomendação de animes que seja utilizável por um usuário final, ou seja, um produto pronto para ser utilizado por consumidor real e que não necessariamente, entenda como se dá a recomendação de forma específica. Para isso, foi necessário escolher um nicho de atuação como forma de implementar uma prova de conceito, e as animações produzidas no Japão, os *animes*, foram selecionadas para essa tarefa. Os *animes* são uma forma de entretenimento muito apreciada atualmente e que tem ganhado cada vez mais espaço na mente das pessoas que apreciam um bom entretenimento.

O objetivo será alcançado por meio de um aprofundamento nos fundamentos teóricos da disciplina e da criação de um sistema: um motor de recomendação, que utilize a técnica de similaridade por cossenos, uma *Application Programming Interface* (API) de acesso ao modelo de recomendação e uma interface gráfica. A partir disso será possível conviver com essa realidade complexa apresentada pela tecnologia de uma forma consciente e livre.

2. SISTEMAS DE RECOMENDAÇÃO

2.1 Introdução aos sistemas de recomendação

Os sistemas de recomendação têm como objetivo aumentar o engajamento dos consumidores ao fornecerem conteúdo adequado às suas preferências pessoais ou à opinião geral do público de um determinado nicho. Podemos afirmar que esses sistemas são os grandes responsáveis por moldar os padrões de consumo online da sociedade moderna [7], já que hoje, praticamente, qualquer grande aplicativo que possui um *feed* de conteúdo, utiliza técnicas de *machine learning* para apresentar informações que sejam de interesse do usuário. Eles são implementados utilizando técnicas de aprendizagem de máquina, que analisam o comportamento dos usuários para identificar padrões de preferência. As principais abordagens para implementação de um sistema de recomendação, na atualidade, são [6]:

2.2 Filtragem baseada em conteúdo

A filtragem baseada em conteúdo é uma abordagem que tem como foco sugerir itens com base nas características de outros itens que o indivíduo já avaliou ou interagiu no passado. Nessa abordagem, cada item, no caso em questão, *anime*, é representado por um conjunto de características também chamadas de *features*. Por exemplo, um *anime* pode ser descrito por atributos como gênero (ação, comédia, drama), diretor, estúdio de produção, ano de lançamento, entre outros. Esses atributos são transformados em vetores de características que descrevem cada item de forma numérica.

O perfil do usuário é então construído com base nas características dos itens que ele já consumiu e avaliou positivamente. Este perfil pode ser representado por um vetor médio ou ponderado dos vetores dos itens que o usuário gostou, então, se um usuário assistiu e gostou de vários *animes* de ação, comédia e aventura, o perfil dele refletirá essas preferências.

A similaridade entre os itens e o perfil do usuário é calculada usando métricas de similaridade. Os itens que possuem alta similaridade com o perfil do usuário são então recomendados. Algumas estratégias para calcular a similaridade entre dois vetores serão discutidas nas próximas sessões do presente trabalho.

Vantagens da filtragem baseada em conteúdo:

- Independência de outros usuários: A recomendação é personalizada para o usuário individual, sem depender de dados de outros usuários.

- Transparência: É mais fácil explicar por que um determinado item foi recomendado, já que a recomendação se baseia em características conhecidas do item.
- Eficácia em dados escassos: Funciona bem mesmo quando há poucos dados de interações de usuários.

Desvantagens da filtragem baseada em conteúdo:

- Necessidade de metadados: Requer uma representação rica e detalhada dos itens, o que pode não estar sempre disponível.
- Falta de diversidade: Pode tender a recomendar itens muito semelhantes aos que o usuário já conhece, limitando a descoberta de novos tipos de conteúdo.
- Complexidade de implementação: A extração e a representação das características dos itens podem ser complexas, especialmente em domínios onde os atributos não são facilmente quantificáveis, como descrição e sinopse, no caso de *animes*.

2.3 Filtragem colaborativa

A filtragem colaborativa tenta prever as preferências de um usuário, analisando as preferências de muitos outros usuários. A ideia central é que, se dois usuários concordaram no passado, é provável que eles concordem no futuro. Essa técnica não depende das características explícitas dos itens, mas sim das interações históricas (avaliações, cliques, visualizações etc.) entre usuários e itens. Existem dois tipos principais de filtragem colaborativa: baseada em usuários e baseada em itens.

Filtragem colaborativa baseada em usuários: Neste método, recomendações são feitas com base na similaridade entre usuários. Se um usuário A tem um histórico de avaliações semelhantes ao de um usuário B, o sistema recomendará ao usuário A os itens que o usuário B avaliou positivamente e que o usuário A ainda não avaliou.

Filtragem colaborativa baseada em itens: Neste método, recomendações são feitas com base na similaridade entre itens. Se um item A é semelhante a um item B e o usuário avaliou positivamente o item A, o sistema recomendará o item B ao usuário.

Vantagens da filtragem colaborativa:

- Independência de metadados: Não requer informações explícitas sobre as características dos itens, apenas as interações entre usuários e itens;
- Capacidade de capturar padrões complexos: Pode identificar padrões complexos de preferências que não são aparentes apenas com base nas características dos itens;

- Aprimoramento contínuo: À medida que mais interações de usuários são registradas, a precisão das recomendações melhora.

Desvantagens da filtragem colaborativa:

- Problema do "*cold start*": Dificuldade em recomendar itens para novos usuários ou recomendar novos itens que ainda não foram avaliados;
- Escalabilidade: Pode ser computacionalmente intensivo para grandes conjuntos de dados, especialmente ao calcular similaridades em tempo real;
- Esparsidade dos dados: Muitos sistemas de recomendação lidam com matrizes de interações esparsas, onde a maioria das entradas é vazia, dificultando a identificação de similaridades.

2.4 Sistemas híbridos

Combina abordagens colaborativas e baseadas em conteúdo [6]. Entretanto, para o escopo deste trabalho de graduação, não serão abordadas técnicas híbridas, por apresentarem um grau de complexidade que não permitiria a criação de um produto digital em tempo hábil.

2.5 Similaridade por cossenos

Uma das abordagens mais simples e eficazes para medir a similaridade entre itens com base nas avaliações (*ratings)* atribuídas a ele. Este método tradicional [8] é amplamente utilizado em filtragens devido a sua simplicidade e eficiência.

A similaridade por cossenos mede a semelhança entre dois vetores de atributos calculando o cosseno do ângulo entre eles. No âmbito deste trabalho, os vetores representarão os *animes* e as suas dimensões, serão as avaliações dos usuários. O valor da similaridade por cossenos varia entre -1 e 1, onde 1 indica que os vetores são idênticos (ou seja, o ângulo entre eles é 0 graus), 0 indica que os vetores são ortogonais (sem similaridade), e -1 indica que os vetores são opostos.

A fórmula para calcular a similaridade por cossenos entre dois vetores A e B é:

$$similaridade(A, B) = \frac{A \cdot B}{||A||||B||}$$

Onde:

$A \cdot B$ é o produto escalar dos vetores A e B.

$||A||$ é a norma do vetor A.

$||B||$ é a norma do vetor A.

A norma [9] de um vetor A é dada por:

$$||A|| = \sqrt{\sum_i x_i^2}$$

Em um sistema de recomendação de *animes*, a similaridade por cossenos pode ser aplicada de duas maneiras principais:

- Similaridade entre itens (Filtragem baseada em conteúdo): Para recomendar *animes* semelhantes a um que o usuário já assistiu e gostou, podemos representar cada *anime* como um vetor de características (como gênero, estúdio, diretor etc.) e calcular a similaridade por cossenos entre esses vetores. Os *animes* com maior similaridade serão recomendados ao usuário.
- Similaridade entre usuários (Filtragem colaborativa): Para recomendar *animes* com base em preferências de usuários semelhantes, podemos representar cada usuário como um vetor de avaliações (ratings) dos *animes* que assistiu. Calculando a similaridade por cossenos entre esses vetores, podemos identificar usuários com gostos semelhantes e recomendar *animes* que eles gostaram, mas que o usuário-alvo ainda não assistiu.

No trabalho em questão, será adotada uma estratégia de recomendação que utilizará filtragem colaborativa baseada em itens, onde a matriz é configurada com itens nas linhas e cada coluna representa a avaliação de um usuário.

Vantagens:

- Simples e rápida: A fórmula é fácil de calcular e pode ser computada de maneira eficiente, mesmo para grandes conjuntos de dados;
- Escalável: Funciona bem com grandes matrizes esparsas comuns em sistemas de recomendação, onde muitos usuários não avaliaram muitos itens;
- Independência de magnitude: Foca na orientação dos vetores, ignorando a magnitude, o que é útil quando a magnitude das avaliações não é relevante.

Desvantagens:

- Dependência de dados explicativos: Requer uma representação significativa dos itens ou usuários em termos de suas características ou avaliações.
- Limitações em dados escassos: Pode ser menos eficaz quando há pouca sobreposição nas avaliações dos usuários ou nas características dos itens.

3. METODOLOGIA

Esta pesquisa aplicada utilizou uma abordagem qualitativa. Para alcançar o objetivo proposto, foram aplicadas as seguintes etapas: Primeiro, foi formulada a hipótese de que é possível utilizar a abordagem de similaridade de cossenos para implementar um sistema de recomendação de animes. Em seguida, foi realizada uma revisão da literatura para fundamentação teórica no tópico de sistemas de recomendação. Na etapa de coleta de dados, a plataforma *Kaggle* foi utilizada para coletar as informações necessárias: tabela de avaliação de animes por usuários da plataforma *MyAnimeList*. Após isso, passamos para a análise e tratamento dos dados, onde criamos uma matriz de recomendação utilizando a técnica de similaridade por cossenos e construímos uma aplicação acessível a um usuário final. É importante destacar que este trabalho não se propõe a avaliar a eficácia do modelo de recomendação.

4. RESULTADOS E DISCUSSÃO

O desenvolvimento do sistema de recomendação de *animes* foi composto por várias etapas bem definidas, cada uma desempenhando um papel crucial para garantir a eficácia do sistema. A seguir, estão detalhadas cada uma dessas etapas:

4.1 Revisão da literatura sobre sistemas de recomendação

Consistiu em uma revisão abrangente da literatura existente sobre sistemas de recomendação, bem como literatura sugerida pelo professor orientador [6]. Este passo foi fundamental para estabelecer uma base teórica sólida e identificar as técnicas e abordagens mais eficazes e inovadoras no campo dos sistemas de recomendação. A revisão incluiu:

- Estudo de diferentes tipos de sistemas de recomendação, como filtragem colaborativa, filtragem baseada em conteúdo e abordagens híbridas.
- Análise de métricas de similaridade e algoritmos de *machine learning* aplicáveis.
- Identificação de estudos de caso e aplicações práticas em diversos domínios, com foco específico em recomendações de mídia, como filmes e séries.

4.2 Criação da matriz de recomendação de *animes*

Para o desenvolvimento deste trabalho, utilizamos uma base de dados de avaliações de *animes* disponível no *Kaggle*, uma plataforma conhecida por hospedar competições de ciência de dados e conjuntos de dados públicos para aprendizado de máquina. A base de dados contém informações detalhadas sobre as avaliações dos usuários para uma ampla gama de *animes*, fornecendo um rico conjunto de informações para treinar e avaliar nosso sistema de recomendação. Ela foi construída a partir do site *MyAnimeList.net* (MAL).

O MAL é uma das maiores e mais populares plataformas online dedicadas aos fãs de *animes* e mangás. Fundada em 2004, MAL oferece uma extensa base de dados onde os usuários podem catalogar, avaliar e comentar sobre seus *animes* e mangás favoritos. Além disso, a plataforma possui uma comunidade ativa de usuários que participam de fóruns de discussão, recomendam séries uns aos outros e compartilham suas listas pessoais de *animes* assistidos. Com uma vasta quantidade de informações e avaliações fornecidas pelos próprios usuários, MAL se tornou uma referência essencial para os fãs de *animes* ao redor do mundo.

Com base nos *insights* obtidos na revisão da literatura, a próxima etapa envolveu a criação de uma matriz de recomendação de *animes* utilizando a estratégia de filtragem colaborativa baseada em itens. O processo incluiu:

- Construção de uma matriz de interações, onde as linhas representam itens (*animes*) e as colunas representam as avaliações dos usuários.
- Cálculo da similaridade entre itens (*animes*) utilizando a similaridade por cossenos. Essa métrica mede a similaridade entre dois vetores (*animes*) considerando o ângulo entre eles, resultando em valores entre 0 e 1, já que os vetores que serão criados nesse projeto, não possuirão dimensões negativas, pois as avaliações irão variar entre 0 (não avaliado), 1 (muito ruim) e 10 (muito bom).

4.3 Criação de um API para consultar a matriz

Para tornar o sistema de recomendação acessível e utilizável, desenvolvemos uma API que permite a consulta da matriz de recomendação. Esta API serve como a camada de comunicação entre o a matriz de similaridade e a *interface* do usuário. A atividade incluiu o desenvolvimento de uma rota para consultas de recomendações com base no nome de um *anime*. Em vez de utilizar um banco de dados tradicional, optamos por armazenar a matriz em um arquivo no formato *parquet*, no próprio *container* da API. Esse formato é otimizado para leitura rápida e manipulação de grandes volumes de dados, sendo adequado para nosso propósito de recomendação [12].

Utilizamos uma abordagem simplificada de armazenamento, já que o objetivo do projeto é entender de forma geral como funciona a operação de um modelo de dados. Para fins de servir como um modelo em ambiente produtivo, existem técnicas específicas e robustas para armazenamento da matriz de similaridade: como a utilização de bancos de dados não relacionais (*NoSQL*) escaláveis [11].

4.4 Criação de uma *interface* gráfica

A *interface* gráfica foi criada para permitir que os usuários finais pudessem interagir facilmente com o sistema de recomendação. Ela permite aos usuários coletar recomendações

com base em um determinado *anime* como ponto de partida. Ou seja, o usuário insere o nome de um *anime* base, e o sistema retorna as recomendações equivalentes.

4.5 Pilha de desenvolvimento

Python [13] foi a linguagem de programação definida para ser nossa ferramenta de desenvolvimento. Esta decisão se baseou nas vantagens que a linguagem oferece, especialmente para problemas *de machine learning*. A seguir, são descritas as principais razões que justificam essa escolha.

É uma linguagem conhecida por sua sintaxe clara e legível, que se assemelha à linguagem natural. Em problemas de *machine learning*, onde muitas vezes é necessário experimentar e iterar rapidamente sobre diferentes modelos e abordagens, sua facilidade de uso é uma grande vantagem.

Possui uma vasta coleção de bibliotecas e *frameworks* específicos para *machine learning* e análise de dados. Algumas das mais importantes, e que foram utilizadas para construção da aplicação prática, incluem:

- *Pandas*: Biblioteca poderosa para manipulação e análise de dados, permitindo a criação e manipulação de *dataframes* de forma eficiente. Esses últimos são como tabelas otimizadas para leitura e processamento de grandes volumes de dados [14].
- *Scikit-learn*: Um dos principais *frameworks* para machine learning, oferecendo uma ampla gama de algoritmos prontos para uso, ferramentas de pré-processamento de dados e avaliação de modelos [15].

Além disso a linguagem também dá suporte a desenvolvimento de aplicações *web*, o que permite ter uma única ferramenta para o desenvolvimento de todo o programa. Duas delas, que também foram utilizadas na criação do produto foram:

- *Flask*: *Framework web* para Python, projetado para ser simples e fácil de usar. Ele fornece as ferramentas essenciais para construir aplicações *web* rapidamente, incluindo um servidor *web* de desenvolvimento e suporte a roteamento de URLs [16].
- *Django: Framework web* de alto nível para Python, projetado para facilitar o desenvolvimento rápido e eficiente de aplicações *web* robustas e escaláveis. Ele vem com uma série de funcionalidades integradas, incluindo um processador de arquivos estáticos, para a criação de *interfaces web* dinâmicas [17].

De acordo com um levantamento feito pelo *Google* [18], Python possui uma das maiores e mais ativas comunidades de desenvolvedores no mundo. Isso faz com que haja grande quantidade de tutoriais, fóruns e documentações sobre a linguagem.

Por fim, a linguagem é amplamente adotada na indústria e academia para pesquisa e desenvolvimento em *machine learning* e *data Science*, o que assegura que as habilidades desenvolvidas são altamente relevantes, transferíveis e confiáveis para o mercado de trabalho.

4.6 Matriz de similaridade com *Jupyter*

A criação da matriz de similaridade foi realizada em um *notebook Jupyter* [10], que é uma ferramenta essencial para o desenvolvimento e experimentação de algoritmos de *machine learning*. Ela oferece um ambiente interativo que facilita a escrita e execução de código Python, além de permitir a visualização instantânea de resultados. Esse ambiente é especialmente útil para a exploração de dados, testes de diferentes abordagens e documentação do processo de desenvolvimento.

Foram utilizados dois *datasets* principais: uma lista de *animes* e uma lista de avaliações dos *animes* por parte dos usuários da plataforma *MAL* supracitada. A lista de *animes* contém informações detalhadas sobre cada *anime*, como título, gênero e descrição, enquanto a lista de avaliações registra as avaliações, de 1 a 10, dos usuários para diferentes *animes*. Como anteriormente mencionado na metodologia, utilizamos dados disponíveis [19] na plataforma *Kaggle*.

Uma etapa crítica no processo foi a transposição da tabela de avaliações. Originalmente, o *dataframe* de avaliações tinha os usuários como linhas e os *animes* como colunas, com as células representando as avaliações dadas pelos usuários aos *animes*. Para calcular a similaridade entre os *animes*, foi necessário transpor esta tabela de forma que as linhas representassem os *animes* e as colunas as avaliações dos usuários. Esta transposição (Apêndice A) permitiu a aplicação de técnicas de similaridade.

Para calcular a similaridade entre os *animes*, foi utilizada a função *cosine_similarity* do componente *Pairwise* que pertence à biblioteca *Scikit-learn*, uma ferramenta poderosa vastamente utilizada na comunidade de dados e aprendizagem de máquina. Após o cálculo da similaridade de todos os vetores previamente criados e demonstrados no Apêndice A, o

algoritmo gerou uma matriz de similaridade que mapeou, um para um, todos os 6668 *animes*. Isso resultou numa matriz 6668 x 6668, onde cada célula representa a similaridade entre o *anime* da linha, e o *anime* da coluna correspondente (Apêndice B). Os códigos fontes completos podem ser encontrados no *Github* [23].

4.7 API de acesso ao modelo com *Flask*

Nessa etapa, foi desenvolvida uma API de *back-end* para servir as recomendações com base na matriz de similaridade. O serviço foi implementando utilizando uma biblioteca para criação de serviços *web*, o *Flask*, que é um *microframework* leve. Foi realizado o carregamento da matriz de recomendação em formato *parquet*, como previamente exposto. Um *dataframe pandas* foi iniciado para manter a matriz de similaridade acessível em memória. Depois foi criado uma rota HTTP para servir o modelo, conforme descrito no *script curl* [21] abaixo, que foi utilizado para testes:

```
curl --request GET \
  --url 'http://host:port/recommendation?anime_name=AnimeName&length=10' \
  --header 'User-Agent: insomnia/2023.5.8'
```

Para fins de testes e posteriormente operação em produção foram desenvolvidos *scripts Docker* e *Docker Compose* [22] para servir a aplicação em *containers*. O código pode ser previamente visualizado no Apêndice C, mas todo o serviço, junto com os arquivos de configuração da aplicação estão disponíveis no *Github* [24].

4.8 Interface gráfica com *Django*

Foi criada a aplicação principal responsável tanto por servir páginas HTML para um browser web, quanto por implementar algumas regras de negócio para coletar o nome do usuário na plataforma *My Anime List* e, a partir disso, buscar a lista de animes já assistidos e mais bem avaliados. Esse Projeto utilizou a API do MAL para acessar essas informações e a interface para essa funcionalidade pode ser visualizada no apêndice D.

Tendo a lista de animes já assistidos pelo usuário, a aplicação faz a escolha dos *top* três mais bem avaliados para construir as recomendações de próximos conteúdos em cima deles.

Para cada um deles, é invocada a API, previamente discutida que, a partir do nome do anime, retorna uma lista de outros similares.

Por fim ocorre uma etapa de refinamento e filtragem dos resultados a fim de remover os animes que possuem similaridade igual a 1 na lista. Finalmente o resultado da lista final é renderizado na tela demonstrada no apêndice E. O os códigos fonte podem ser verificados no *Github* [25].

4.9 Limitações e recomendação para trabalhos futuros

A filtragem colaborativa baseada em similaridade de cossenos pode enfrentar problemas com a esparsidade dos dados. Em sistemas de recomendação, a quantidade de usuários e itens tende a ser numerosa. Se muitos itens não foram avaliados por muitos usuários, a matriz de avaliações se torna esparsa [27], o que dificulta a identificação de vizinhos próximos. Isso pode afetar a qualidade da recomendação final. Essa técnica também é limitada pelo "problema do início a frio" [28], que ocorre quando um novo usuário ou item entra no sistema e não há informações suficientes para calcular a similaridade com outros usuários ou itens. Pelo fato de a abordagem por cossenos necessitar de dados históricos para funcionar bem, isso pode dificultar a consistência do modelo no tempo. Se mal operado o modelo pode facilmente se tornar obsoleto. Por fim, a abordagem apresentada neste trabalho utiliza a filtragem colaborativa baseada em itens, cujo único parâmetro de similaridade são as avaliações. Trabalhos futuros podem acrescentar mais métricas de interação dos usuários com as obras, para compor o cálculo de similaridade dos itens, como por exemplo, o número ou natureza dos comentários nas páginas oficiais do *anime* no MAL. Uma abordagem híbrida, que considere propriedades intrínsecas ao item (filtragem baseada em conteúdo), para o cálculo de similaridade, também pode tornar o modelo de recomendação mais robusto.

5. Conclusão

Este trabalho abordou a disciplina de recomendação sobre o viés teórico e empírico. No processo foi revisada a literatura fundamental e foram aplicadas técnicas de aprendizagem de máquina para construir uma aplicação. Foi escolhido *Python* como a principal linguagem de programação, decisão que foi fundamentada nas, descritas, múltiplas vantagens da tecnologia.

Foi aplicada uma metodologia que, além do estudo teórico, que permitiu implementar, de forma prática, uma matriz de similaridades utilizando a estratégia de similaridade por cossenos. Para a construção dessa matriz, foi utilizado o ambiente interativo dos notebooks *Jupyter*, abordagem que facilitou a experimentação e a documentação simultânea. Os *datasets* de animes e avaliações foram manipulados e transpostos adequadamente para permitir uma construção correta da estratégia de filtragem colaborativa baseada em itens.

Além disso foi implementada uma API e uma interface gráfica para servir o modelo de recomendação ao público por meio da internet. O sistema foi submetido a testes e validação manual, para garantir que as recomendações fornecidas eram minimamente relevantes e úteis.

É razoável considerar que esse trabalho traz um ganho para a comunidade que aprecia *animes* como forma de entretenimento, já que, em meio ao bombardeio de conteúdo que hoje impera na internet, um sistema de recomendação pode auxiliar um indivíduo a poupar o tempo que gastaria procurando algo, para consumir uma boa obra sugerida com base nas avaliações de centenas de milhares de usuários.

Em resumo, este trabalho demonstrou eficácia em combinar diferentes tecnologias para criar um sistema simples de recomendação de *animes*. As escolhas feitas ao longo do desenvolvimento, desde a seleção das ferramentas até a metodologia aplicada, resultaram em um produto digital completo. Isso permitiu obter uma visão mais apurada sobre os sistemas de recomendação, que são ferramentas fortemente empregadas nos dias de hoje.

Referências

[1] Wikipedia. Anime. Disponível em: https://pt.wikipedia.org/wiki/Anime. Acesso em 04/05/2024.

[2] Wikipedia. Sistema de recomendação. Disponível em: <https://pt.wikipedia.org/wiki/Sistema_de_recomenda%C3%A7%C3%A3o> Acesso em 04/05/2024;

[**3]** Amazon Web Services (AWS). O que é uma API? Disponível em: <https://aws.amazon.com/pt/what-is/api/> Acesso em 04/05/2024;

[4] Centro de Informática UFPE. Teste de Usabilidade. Disponível em: <https://www.cin.ufpe.br/~gta/rup-vc/core.base_rup/guidances/concepts/usability_testing_EA14CC80.html> Acesso em 04/05/2024;

[5] INNOQ. *MLOps Principles.* Disponível em:<https://ml-ops.org/content/mlops-principles> Acesso em 07/07/2024;

[6] RICCI, Francesco; ROKACH, Lior; SHAPIRA, Bracha. Recommender Systems Handbook. 2. ed. New York: Springer, 2015;

[7] THE SOCIAL DILEMMA. Direção: Jeff Orlowski. Roteiro: Jeff Orlowski, Davis Coombe. Elenco: Skyler Gisondo, Kara Hayward, Vincent Kartheiser. Netflix, 9 de setembro de 2020. Documentário (1h 29min);

[8] *Science Direct. Cosine Similarity.* Disponível em: <https://www.sciencedirect.com/topics/computer-science/cosine-similarity> Acesso em 16/07/2024;

[9] UNICAMP. Norma e Distância. Disponível em < https://www.ime.unicamp.br/~marcia/AlgebraLinear/norma.html> Acesso em 16/07/2024;

[10] *Jupyter. Project Jupyter Documentation.* Disponível em <https://docs.jupyter.org/en/latest/> Acesso em 17/07/2024;

[11] Oracle. *What is NoSQL.* Disponível em <https://www.oracle.com/br/database/nosql/what-is-nosql/> Acesso em 17/07/2024;

[12] *Databricks. What is parquet.* Disponível em

<https://www.databricks.com/br/glossary/what-is-parquet>
Acesso em 17/07/2024;
[13] *Python Software Foundation. Python.* Disponível em <https://www.python.org/about/>
Acesso em 18/07/2024;
[14] *NumFocus. Pandas Documentation.* Disponível em
<https://pandas.pydata.org/docs/index.html>
Acesso em 18/07/2024;
[15] *Scikit-Learn. Machine Learning in Python.* Disponível em <https://scikit-learn.org/stable/user_guide.html>
Acesso em 18/07/2024;
[16] *Flask. Users Guide.* Disponível em <https://flask.palletsprojects.com/en/3.0.x/#user-s-guide>
Acesso em 18/07/2024;
[17] *Django Project. Django Overview.* Disponível em
<https://www.djangoproject.com/start/overview/>
Acesso em 18/07/2024;
[18] Exame. As 5 linguagens mais populares do mundo. Disponível em < https://exame.com/bussola/gosta-de-programacao-confira-as-5-linguagens-mais-populares-do-mundo/>
Acesso em 18/07/2024;
[19] *Kaggle. MyAnimeList Dataset.* Disponível em
<https://www.kaggle.com/datasets/azathoth42/myanimelist?select=anime_filtered.csv>
Acesso em 18/07/2024; https://docs.docker.com/
[20] ORTEGA Y GASSET, José. *Meditações do Quixote*. Tradução de Josué Cândido de Mello. São Paulo: Cortez, 1992.
[21] *Curl. Documentation.* Disponível em < https://curl.se/docs/ >
Acesso em 24/07/2024;
[22] *Docker. Documentation.* Disponível em < https://docs.docker.com/>
Acesso em 24/07/2024;
[23] *Github. Anime Recommendation Algo*. Disponível em
<https://github.com/afa4/anime_recommendation_algo>
Acesso em 24/07/2024;
[24] *Github. Anime Recommendation Server.* Disponível em
<https://github.com/afa4/anime_recommendation_server>

Acesso em 24/07/2024;

[25] *Github. Anime Recommendation Site.* Disponível em <https://github.com/afa4/anime_recommendation_site>
Acesso em 24/07/2024;

[26] SINGH, Ramni Harbir et al. Movie recommendation system using cosine similarity and KNN. International Journal of Engineering and Advanced Technology, v. 9, n. 5, p. 556-559, 2020.

[27] *1 Library.* Filtragem Colaborativa. Disponível em < https://1library.org/article/filtragem-colaborativa-sistemas-de-recomenda%C3%A7%C3%A3o.ynl7x9jq>
Acesso em 02/08/2024.

[28] KUZNETSOV, Stanislav; KORDÍK, Pavel. Overcoming the cold-start problem in recommendation systems with ontologies and knowledge graphs. In: European Conference on Advances in Databases and Information Systems. Cham: Springer Nature Switzerland, 2023. p. 591-603.

Apêndices

APÊNDICE A – TRANSPOSIÇÃO DA TABELA DE AVALIAÇÕES

```
[17]
animes_rated_by_users = pd.pivot_table(ratings_merged, index='title', columns='username', values='my_score').fillna(0)

[18]
animes_rated_by_users.head()
```

username	---- phoebelyn	---L-AND-AME-4EV	-- AnimeBoy-	-- Etsuko-	-- FallenAngel-	-- Kin-	-- N-	-- PHOENIX-	-- RIE-	-- Sc-	...	zzeldaxx7	zzeroparticle	zzganz	zzrorozz
title															
"Bungaku Shoujo" Kyou no Oyatsu: Hatsukoi	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
"Bungaku Shoujo" Memoire	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
"Bungaku Shoujo" Movie	0.0	0.0	6.0	0.0	0.0	6.0	0.0	0.0	6.0	0.0	...	0.0	7.0	7.0	0.0
"Eikou Naki Tensai-tachi" Kara no Monogatari	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
"Eiyuu" Kaitai	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0

5 rows × 108710 columns

APÊNDICE B – MATRIZ DE SIMILARIDADE POR COSSENO

```
[5]
rec_df = pd.DataFrame(rec, columns=animes_rated_by_users.index, index=animes_rated_by_users.index)
rec_df
```

title	"Bungaku Shoujo" Kyou no Oyatsu: Hatsukoi	"Bungaku Shoujo" Memoire	"Bungaku Shoujo" Movie	"Eikou Naki Tensai-tachi" Kara no Monogatari	"Eiyuu" Kaitai	"Parade" de Satie	.hack//G.U. Returner	.hack//Gift	.hack//Intermezzo	.hack//Liminality	...
title											
"Bungaku Shoujo" Kyou no Oyatsu: Hatsukoi	1.000000	0.686575	0.560973	0.000000	0.089963	0.011125	0.104888	0.094758	0.100687	0.102224	...
"Bungaku Shoujo" Memoire	0.686575	1.000000	0.644669	0.030122	0.090460	0.010338	0.105728	0.086793	0.097533	0.097395	...
"Bungaku Shoujo" Movie	0.560973	0.644669	1.000000	0.021984	0.070997	0.007545	0.092292	0.080690	0.097768	0.093442	...
"Eikou Naki Tensai-tachi" Kara no Monogatari	0.000000	0.030122	0.021984	1.000000	0.079741	0.000000	0.019568	0.016859	0.016804	0.030577	...
"Eiyuu" Kaitai	0.089963	0.090460	0.070997	0.079741	1.000000	0.000000	0.041182	0.022168	0.025172	0.030261	...
...	...	...	...	...	...	...	...	...	...	...	...

APÊNDICE C – API PARA SERVIR MATRIZ DE SIMILARIDADE

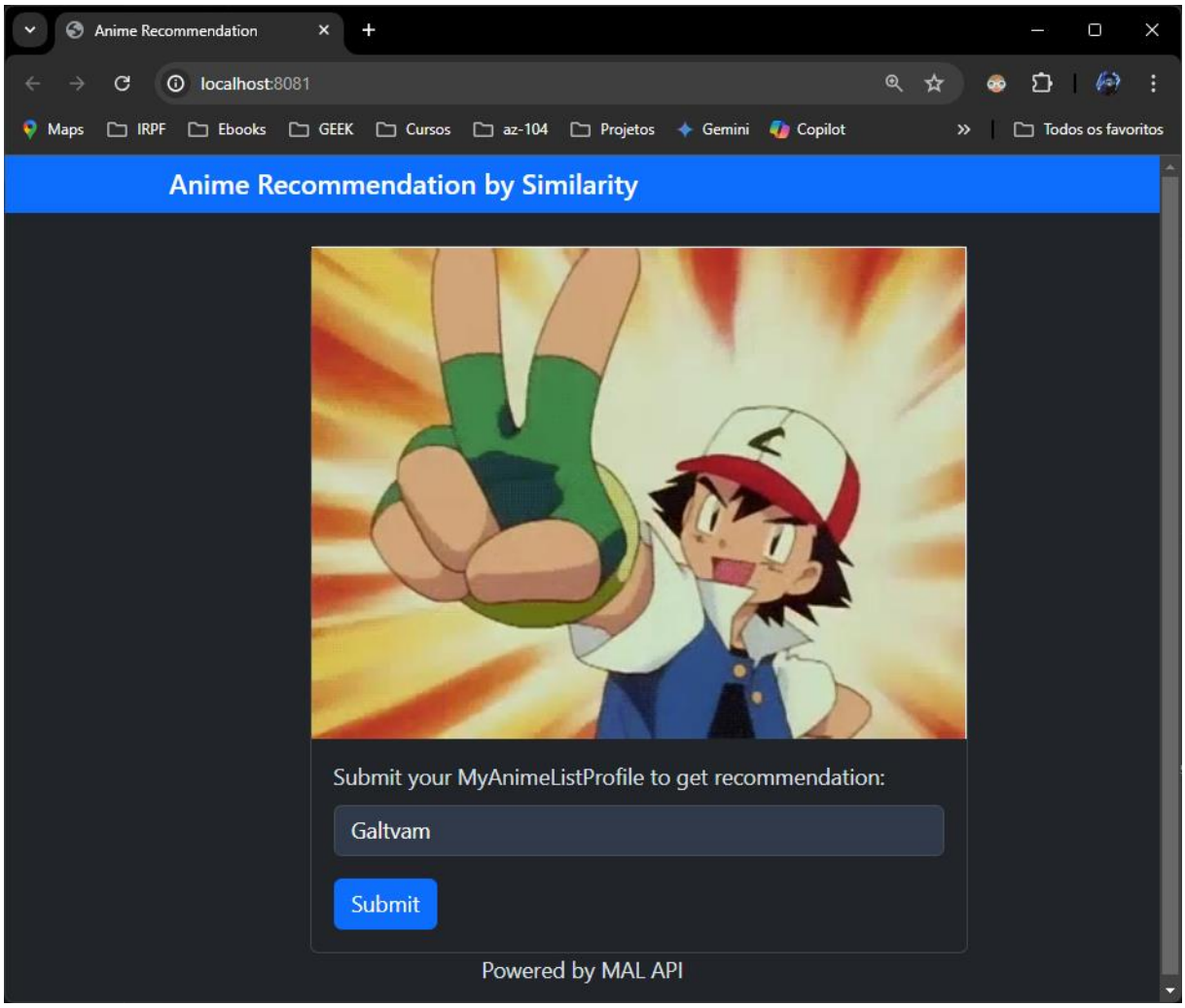
app.py

app.py > Flask

You, last week | 1 author (You)

```python
from flask import Flask, make_response, jsonify, request      You, last week • initial commit
import pandas as pd
import logging
from waitress import serve

logger = logging.getLogger('waitress')
logger.setLevel(logging.INFO)

class ModelServer:
    def __init__(self):
        self.model = pd.read_parquet('recommendation_cosine_matrix.parquet')


    def get_recommendation(self, anime_name, length = 20):
        recommendation = self.model[anime_name].sort_values(ascending=False)
        return recommendation.head(length)


model_server = ModelServer()
app = Flask(__name__)


@app.route('/recommendation')
def recommendation():
    # logger.info('Request Received')
    anime_name = request.args.get('anime_name')
    if(anime_name is None):
        return make_response('Bad Request', 400)
    length = request.args.get('length', default=20, type=int)
    if length > 20:
        length = 20
    recommendation = model_server.get_recommendation(anime_name, length)

    return recommendation.to_json()


@app.route('/')
def index():
    return make_response('Anime Recommendation API :3', 200)


if __name__ == '__main__':
    serve(app, host="0.0.0.0", port=8080)
```

APÊNDICE D – INTERFACE PARA INCLUSÃO DE PERFIL DE USUÁRIO DO MY ANIME LIST



APÊNDICE E – INTERFACE PARA APRESENTAR RESULTADO DA RECOMENDAÇÃO