Sistemas de recomendação de música usando a plataforma JINA.AI

Music Recommendation System using the Jina.AI platform

Matheus Alves Sousa e Silva[1] e Rafael Lima de Carvalho[1]

[1] *Curso de Ciência da Computação, Universidade Federal do Tocantins, Tocantins, Brasil*

Data de recebimento do manuscrito: 23/08/2023
Data de aceitação do manuscrito: 30/09/2023
Data de publicação: 16/10/2023

Resumo— Nos últimos anos, o consumo de música passou por mudanças significativas, com o *streaming* online se tornando uma opção popular. Grandes plataformas como *Spotify* e *Apple Music* oferecem vastos catálogos, impulsionando sistemas de recomendação para atender aos usuários. Tais sistemas são complexos e precisam ser escaláveis para lidar com volumes de dados. O Jina.AI, uma plataforma líder em inteligência artificial multimodal, destaca-se entre as opções para criar soluções escaláveis. Assim, o presente trabalho tem por objetivo mostrar a criação de um sistema de recomendação de músicas, baseado em conteúdo e usando-se o conceito de *embeddings* para indexação e busca neural usando a plataforma Jina.AI. Os resultados do experimento apresentado demonstra a viabilidade, versatilidade e facilidade para criação de um sistema de recomendação baseado em conteúdo com Jina.AI.

Palavras-chave—Sistemas de Recomendação, Recomendação Musical, Jina.AI

Abstract— *In recent years, music consumption has undergone significant changes, with online streaming becoming a popular option. Major platforms like Spotify and Apple Music offer vast catalogs, powering recommendation systems to cater to users. Such systems are complex and need to be scalable to handle data volumes. Jina.AI, a leading multimodal artificial intelligence platform, stands out among the options for creating scalable solutions. Thus, this study aims to demonstrate the creation of a content-based music recommendation system using the concept of embeddings for indexing and neural search using the Jina.AI platform. The experiment's results presented show the viability, versatility, and ease of creating a content-based recommendation system with Jina.AI.*

Keywords—*Recommender Systems, Music Recommendation, Jina.ai.*

I. Introdução

A música transcende as fronteiras do cotidiano e se estabelece como um elemento intrínseco à vida humana, participando ativamente de momentos triviais e significativos, enquanto influencia a evolução da espécie e da sociedade [1]. Sua presença histórica se manifestou por meio de diferentes meios, desde apresentações ao vivo até a era dos suportes físicos como discos de vinil, CDs e fitas K7. Enfitretanto, a era digital, impulsionada por avanços na velocidade de transmissão de dados e na infraestrutura da internet, trouxe uma revolução no consumo musical, onde o acesso via internet representa atualmente uma parcela significativa, atingindo 64% de todos os formatos de consumo [2].

Plataformas de *streaming*, como Spotify e Apple Music, emergiram como catalisadoras dessa transformação, oferecendo catálogos vastos, contendo centenas de milhões de faixas [3]. No entanto, essa profusão de escolhas também deu origem a um dilema: a vastidão do repertório disponível torna inviável para os usuários explorarem integralmente as opções. Nesse contexto, os sistemas de recomendação desempenham um papel fundamental [4].

Os sistemas de recomendação, embasados em algoritmos sofisticados, têm a função de filtrar e personalizar o vasto conjunto de itens musicais, entregando aos usuários uma seleção alinhada com suas preferências históricas, padrões de usuários semelhantes e características intrínsecas das próprias músicas [5]. O Spotify, por exemplo, opera em escala global, servindo inúmeras pessoas simultaneamente, o que demanda robustez e escalabilidade para lidar com o constante influxo de novos usuários e músicas.

Diante desse cenário, surge o Jina.AI, uma plataforma de *Machine Learning Operations* (MLOps) que se destaca como uma solução avançada para a construção e implementação de sistemas de recomendação [6]. Ao oferecer suporte a diversos algoritmos de busca vetorial e uma estrutura flexível para a integração de diversos modelos de aprendizado de

Dados de contato: Matheus Alves Sousa e Silva, matheus.alves1@uft.edu.br

máquina, a plataforma Jina.AI enfrenta o desafio de criar sistemas de recomendação personalizados e eficientes. Além disso, sua capacidade de expor serviços implementados à internet é facilitada pela inclusão de serviços nativos de computação em nuvem.

Assim, o presente artigo descreve experimentos computacionais que exploram a implementação de técnicas de sistemas de recomendação por meio da plataforma Jina.AI. Este artigo está estruturado da seguinte forma: na Seção II, são abordadas as bases de dados, algoritmos e ferramentas empregados, detalhando sua aplicação. A Seção III exibe os resultados obtidos com os experimentos realizados. A Seção IV encerra o artigo com as considerações finais.

II. MATERIAIS E MÉTODOS

Nesta seção, apresenta-se a base de dados de letras de músicas utilizada para ilustrar a prova de conceito aqui proposta (subseção IIa). Em seguida, a subseção IIb apresenta uma rápida descrição da ferramenta Jina.AI. A subseção IIc descreve um experimento baseado em conteúdo (*content-based*) tendo por objetivo mostrar as etapas necessárias para montar um recomendador de músicas usando Jina.AI.

a. Base de dados de letras de músicas

A disponibilidade de conjuntos de dados amplos é crucial para efetivamente conduzir as fases de treinamento e avaliação de sistemas de recomendação. Bases de dados específicas para sistemas de recomendação musical são elaboradas para facilitar a pesquisa, permitindo que os cientistas se concentrem nas técnicas e algoritmos subjacentes. Ao serem disponibilizadas publicamente, essas bases de dados também resolvem o desafio do licenciamento de músicas. Nesta seção, as principais fontes públicas de dados para sistemas de recomendação musical são apresentadas e detalhadas.

A base aqui considera foi a "Song lyrics from 79 musical genres", disponível no *Kaggle*[1], foi criada por Anderson Neise. Ela reúne letras de músicas de 79 gêneros musicais distintos, coletadas do site Vagalume. Os dados abrangem 379.893 letras de 4.239 artistas, acompanhadas de metadados como nome do artista, gênero, linguagem e uma métrica de popularidade. No estudo, somente letras em inglês foram utilizadas para garantir a compatibilidade com o modelo de extração de *embeddings*. Após filtragem, a base continha 191.385 letras, e devido a limitações de processamento, foi reduzida em 50%, totalizando 95.693 registros.

b. A plataforma Jina.AI

Jina.AI trata-se de uma tecnologia de código aberto que permite criar aplicações de inteligência artificial multimodais com tecnologias nativas para a nuvem. Ela funciona através de quatro pilares a saber [6]: *Document*, *DocumentArray*, *Executors* e *Flow*. *Document* é uma estrutura de dados para representar a informação a ser tratada. No caso do presente trabalho, trata-se da estrutura que armazenará as letras das músicas. O *DocumentArray* é uma estrutura de dados capaz de armazenar dados prontos para serem transmitidos via rede. O *DocumentArray* já possui embutida medidas de similaridade tais como cosseno e distância Euclidiana. Um *Executor* é capaz de realizar um trecho de código, sempre tomando como entrada um *DocumentArray* e emitindo como saída um *DocumentArray* (possivelmente modificado). Por fim, o *Flow* é responsável pela orquestração dos executores. Os executores podem ser classes Python feitas pelo próprio desenvolvedor como também códigos disponíveis no Jina Hub. Este é um hub de Executores de código aberto já prontos para serem usados. A Figura 1 exibe o fluxo usado no experimento deste trabalho e sua configuração como Flow do Jina.AI na Figura 3.

c. Recomendações utilizando letras de músicas usando Jina.AI

A execução do experimento acontece em duas etapas: o *flow* descrito na Figura 1 representa a etapa de indexação dos dados; enquanto o descrito na Figura 2 descreve a etapa de busca dos resultados e geração das recomendações. Todos os executores Jina.AI utilizados para esse experimento podem ser encontrados no Jina Ai Executor Hub [7].

Etapa de Indexação

O processo começa com letras de músicas a serem indexadas armazenadas em um *Document* do DocArray (Figura 1a). O executor *Sentencizer* divide o texto em pequenas partes chamadas *chunks*, armazenando-os no atributo `.chunks` do *Document* de entrada (Figura 1b). O executor *Tags Copier* replica os metadados da música para os *chunks*, permitindo a identificação durante a busca (Figura 1c).

Os *embeddings* dos *chunks* são extraídos usando o *TransformerTorchEncoder* (Figura 1d). Isso cria representações vetoriais densas que capturam similaridades semânticas entre palavras e frases. O modelo usado é o *all-MiniLM-L6-v2*, um *sentence-transformer* de 384 dimensões. Cinco instâncias do *TransformerTorchEncoder* são usadas para processar vários documentos simultaneamente.

O *AnnLiteIndexer* recebe como entrada um vetor de documentos contendo as propriedades *chunks* e *embeddings*, armazena-os na memória ou na nuvem usando o banco de dados RocksDB [8]. No fluxo especificado, foram previstos cinco *shards* separados para otimizar a indexação. Parâmetros específicos são definidos para o *AnnLiteIndexer* para influenciar a precisão da indexação e qualidade das buscas. Detalhes dos parâmetros estão na Tabela 1.

TABELA 1: PARÂMETROS UTILIZADOS NO EXECUTOR ANNLITEINDEXER

Parâmetro	Valor
n_dim	384
ef_search	64
ef_construction	128
max_connection	32
index_acess_paths	@c

[1]https://www.kaggle.com/datasets/neisse/scrapped-lyrics-from-6-genres

ISSN: 2675-3588

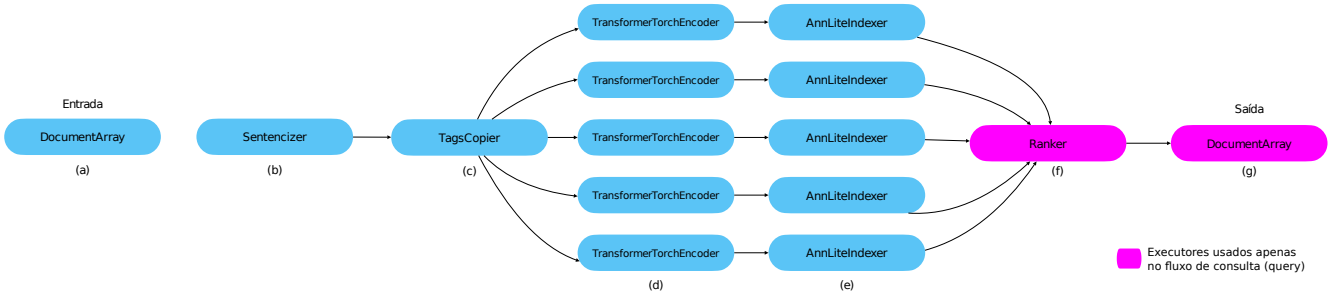

Figura 1: O flow definido para a indexação e busca das letras de músicas. Os executores apresentados em (f) e (g) fazem parte apenas da etapa de busca (ou geração da recomendação).

Etapa de Busca

Conforme o que se apresenta na Figura 1, o fluxo inicia recebendo um *DocumentArray* contendo um único Document (consulta) onde será separado em chunks pelo *Sentencizer chunks* cujas *embeddings* serão calculados pelo *TransformerTorchEncoder*. Assim, ele executa buscas para comparar os *embeddings* dos *chunks* com os do banco de dados gerados na etapa de indexação, usando cinco *shards*.

O resultado da busca é armazenado no atributo `.matches` dos *chunks* no *Document* de busca. Dentro de `.matches` ficam armazenados os $N = 10$ *Documents* mais semelhantes aos *chunks* do *Document* de busca, considerando-se a métrica de similaridade do cosseno.

A última etapa ocorre no *Ranker* (Figura 1f). Ele recebe *Documents* do executor anterior, contendo `.matches` com os *chunks* mais semelhantes. O *Ranker* coleta os *matches* para cada *chunk*, ordenando-os por similaridade de cosseno. Esses *matches* são armazenados na raiz do *Document* de busca, em `.matches`.

O resultado do *flow* de busca é o próprio *Document* de busca, contendo os *Documents* mais semelhantes encontrados no banco de dados indexado. Ambos os fluxos retornam recomendações em formato de *Document*, contendo *ids* e *tags* de artistas recomendados.

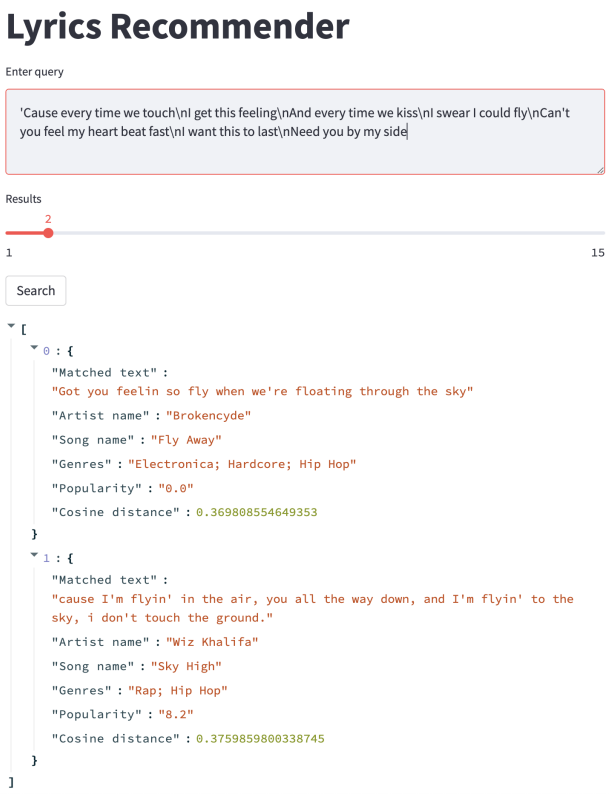

Figura 2: Aplicação de prova de conceito mostrando a resposta do indexador construído com Jina.AI.

III. Resultados

Ao iniciar o fluxo definido na seção c, a plataforma Jina.AI permite que um serviço em formato *Representational State Transfer* (REST) fique disponível na porta e endereço configurado pelo desenvolvedor. A Figura 2 exibe uma aplicação *front-end* criada usando a biblioteca *Streamlit* onde é possível estipular o documento ou parte dele em formato de consulta e obter até 15 (quinze) documentos recomendados de acordo com a proximidade de cosseno.

IV. Considerações Finais

Este artigo mostrou a utilização da ferramenta de *MLOps* Jina.AI para a implementação de sistemas de recomendação de música. Para tanto, foi proposto um fluxo onde considera as letras de músicas em uma representação de *embeddings* gerada pelo modelo pré-treinado para palavras da língua inglesa *sentence-transformers*. Apresentou-se a versatilidade e simplicidade da ferramenta Jina.AI para

```
jtype: Flow
version: '1'
gateway:
  protocol: [grpc, http, websocket]
  port: [54324, 54325, 54326]
workspace: './workspace'

executors:
- name: sentencizer
  uses: jinaai://jina-ai/Sentencizer:latest

- name: transformer_encoder
  uses: jinaai://jina-ai/TransformerTorchEncoder:latest
  replicas: 5
  uses_with:
    pretrained_model_name_or_path:
        sentence-transformers/all-MiniLM-L12-v2
    traversal_paths: "@c"

- name:  ann_indexer
  uses: ann_exec/config.yml
  install_requirements: true
  shards: 5
  polling: {'/index': 'ANY',
            '/search': 'ALL',
            '/update': 'ALL',
            '/delete': 'ALL',
            '/status': 'ALL'
            }
  uses_with:
      n_dim: 384
      ef_search: 64
      ef_construction: 128
      max_connection: 32
  workspace: ./workspace
  timeout-ready: -1

- name: ranker
  uses: jinaai://jina-ai/SimpleRanker:latest
  uses_with:
    metric: cosine
  install_requirements: true
```

Figura 3: Código Yaml definindo o fluxo de dados na plataforma Jina.AI.

ISSN: 2675-3588

a rápida prototipação de um sistema de recomendação de músicas baseado em conteúdo. Para a aplicação em consideração, utilizou-se a base de dados “Song lyrics from 79 musical genres”.

REFERÊNCIAS

[1] J. Schulkin and G. B. Raglan, “The evolution of music and human social capability,” *Frontiers in Neuroscience*, vol. 8, 2014. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fnins.2014.00292

[2] A. Alexander, “A visual breakdown of global music consumption,” Feb 2023. [Online]. Available: https://www.visualcapitalist.com/cp/a-visual-breakdown-of-global-music-consumption/

[3] W. Apple, “Celebrating 100 million songs,” May 2023. [Online]. Available: https://www.apple.com/newsroom/2022/10/celebrating-100-million-songs/

[4] C. C. Aggarwal, *Recommender Systems - The Textbook*. Springer, 2016.

[5] B. Rocca, “Introduction to recommender systems,” Jun 2019. [Online]. Available: https://towardsdatascience.com/introduction-to-recommender-systems-6c66cf15ada

[6] B. Wang, C. Mitroi, F. Wang, S. Saboo, and S. Guzman, *Neural Search-From Prototype to Production with Jina: Build deep learning–powered search systems that you can deploy and manage with ease*. Packt Publishing Ltd, 2022.

[7] J. ExecutorHub, “Executor hub,” Jun 2023. [Online]. Available: https://docs.jina.ai/concepts/serving/executor/hub/index.html

[8] M. RockDB, “A persistent key-value store,” 2023. [Online]. Available: https://rocksdb.org/

ISSN: 2675-3588