Sistema de Recomendação Colaborativo com API REST

Nesta tarefa, você mergulhará no mundo dos sistemas de recomendação construindo um sistema de filtragem colaborativo e expondo sua funcionalidade por meio de uma API REST. A filtragem colaborativa é uma técnica poderosa usada por muitas plataformas populares como Amazon, Netflix e Spotify para fornecer recomendações personalizadas aos usuários. Você ganhará experiência prática com manipulação de dados, treinamento de modelo e desenvolvimento de API, criando, em última análise, um sistema que pode sugerir itens com base nas preferências e comportamento do usuário.

Objetivos

- Implemente um algoritmo de filtragem colaborativa (baseado em usuário ou item) para gerar recomendações.
- Projete e desenvolva uma API REST seguindo as especificações descritas no documento "Common Recommender REST API".
- Utilize conjuntos de dados disponíveis publicamente para treinar e testar seu sistema de recomendação, indicados abaixo.
- Documente claramente seu código e endpoints de API.

Requisitos

Requerimentos técnicos:

- Linguagens de programação: Python é recomendado devido ao seu rico ecossistema de bibliotecas para ciência de dados e desenvolvimento de API; R - uma das APIs será feita em R.
- Bibliotecas:
    - Manipulação e análise de dados: pandas, NumPy
    - Filtragem colaborativa: scikit-learn
    - Desenvolvimento de API REST: Flask ou FastAPI - e R com pumbleR
- Conjuntos de dados: escolha pelo menos um dos seguintes conjuntos de dados para trabalhar:
    - Conjunto de dados do sistema de recomendação de artigos (Kaggle)
    - Conjunto de dados de recomendação de livros (Kaggle)
- Especificações da API: siga as diretrizes fornecidas no documento "Common Recommender REST API", garantindo que sua API cubra as seguintes funcionalidades:
    - Adicionando usuários e itens
    - Itens de classificação
    - Recuperando recomendações para um usuário
    - Recuperando itens semelhantes a um determinado item

Requisitos de implementação:

1. Pré-processamento de dados:
    - Carregue o conjunto de dados escolhido em seu ambiente Python usando pandas.

- Explore os dados para compreender sua estrutura, recursos e possíveis problemas (valores ausentes, etc.).
- Execute todas as etapas necessárias de limpeza e pré-processamento de dados.
- Prepare os dados para treinamento do modelo (por exemplo, criando matrizes de itens do usuário).

Um exemplo de trecho de código:

```
import pandas as pd

# Load the dataset using pandas
data = pd.read_csv("path/to/dataset.csv")

# Explore the data
print(data.head())
print(data.info())

# Handle missing values (e.g., imputing with mean/median)
data.fillna(data.mean(), inplace=True)

# Create user-item matrix (for user-based CF)
user_item_matrix = data.pivot_table(index="user_id", columns="item_id", values="rating")
```

2. Modelo de filtragem colaborativa:
    - Escolha e implemente os algoritmos de filtragem colaborativa baseados em usuário e em item.
        - Uma API será feita com Python e a outra será feita com R.
    - Treine os modelos escolhidos nos dados pré-processados.
    - Avalie o desempenho do seu modelo usando métricas apropriadas (por exemplo, precisão, recall).

    Um exemplo de trecho de código:

```
from sklearn.neighbors import NearestNeighbors

# User-based CF with k-nearest neighbors
model = NearestNeighbors(n_neighbors=5, algorithm="brute")
model.fit(user_item_matrix)

# Get recommendations for a user
user_id = 123
distances, neighbors = model.kneighbors(user_item_matrix.loc[user_id])

# ... (process neighbors to get recommendations) ...
```

3. Desenvolvimento de API REST:
    - Use Flask ou FastAPI (Python) e o PumbleR (R) para criar uma API REST que exponha as funcionalidades do seu sistema de recomendação.
    - Defina os endpoints das APIs de acordo com o documento "Common Recommender REST API".
    - Implemente funções para lidar com solicitações para cada endpoint, incluindo:
        - Adicionando novos usuários e itens ao sistema.
        - Processando classificações de usuários para itens.
        - Gerar e retornar recomendações para um determinado usuário.
        - Encontrar e devolver itens semelhantes a um item especificado.
    - As APIs em Python e R deverão ser utilizadas integralmente no seu programa, ou seja, do seu programa principal em Python com Flask você deverá consumir o

webservice feito em R com o PumbleR para obter a recomendação respectiva. Você até pode expor tudo a partir do Flask, como abaixo, mas internamente, a função deverá consumir o webservice feito no PumbleR.

Um exemplo de trecho de código:

```
from flask import Flask, jsonify, request

app = Flask(__name__)

@app.route("/users/<user_id>/recommendations", methods=["GET"])
def get_recommendations(user_id):
    # ... (get recommendations from your model) ...
    return jsonify(recommendations)

@app.route("/items/<item_id>/similar", methods=["GET"])
def get_similar_items(item_id):
    # ... (get similar items from your model) ...
    return jsonify(similar_items)

if __name__ == "__main__":
    app.run(debug=True)
```

4. Teste e documentação:
    - Teste minuciosamente os endpoints da API para garantir que funcionem conforme esperado.
    - Documente seu código de forma clara, explicando suas escolhas de implementação e decisões de design.
    - Forneça documentação da API que descreva os endpoints disponíveis, formatos de solicitação/resposta e exemplos de uso.

Recursos

- Documento “Common Recommender REST API”: https://drive.google.com/file/d/1ztGhAVlg_jcyGQGZMkMXhGKBixfO6uTI/view?usp=sharing
- Conjuntos de dados:
    Sugestão
    - https://www.kaggle.com/datasets/jainilcoder/article-recommendation-system
    - https://www.kaggle.com/datasets/arashnic/book-recommendation-dataset
    Outros
    - https://github.com/caserec/Datasets-for-Recommender-Systems
    - https://cseweb.ucsd.edu/~jmcauley/datasets.html
- Tutorial de filtragem colaborativa: https://realpython.com/build-recommendation-engine-collaborative-filtering/
- Tutorial API REST com PumbleR:
    - https://solutions.posit.co/gallery/plumber-ml/
    - https://juanitorduz.github.io/intro_plumber/
    - https://rviews.rstudio.com/2018/07/23/rest-apis-and-plumber/
    - https://rpubs.com/nomarpicasso/1150563
- Bibliotecas Python:
    - pandas: https://pandas.pydata.org/
    - NumPy: https://numpy.org/
    - scikit-learn: https://scikit-learn.org/stable/

- Flask: https://flask.palletsprojects.com/
- FastAPI: https://fastAPI.com/
- PumbleR: https://www.rplumber.io/

Avaliação

Sua tarefa será avaliada com base nos seguintes critérios:

- Funcionalidade: A API implementa corretamente as funcionalidades especificadas?
- Desempenho: O sistema de recomendação gera recomendações precisas e relevantes?
- Qualidade do código: o código está bem estruturado, legível e documentado?
- Design da API: a API segue os princípios RESTful e as especificações fornecidas?
- Documentação: o código e a funcionalidade da API estão claramente documentados?
- A participação de cada membro da equipe será avaliada individualmente.

Esta atividade poderá ser feita em grupos de até 3 (três) componentes.

Bônus

- Implemente filtragem colaborativa baseada em usuário e em item e permita que os usuários escolham entre elas por meio da API.
- Explore e implemente técnicas mais avançadas, como fatoração de matrizes ou métodos baseados em aprendizagem profunda para filtragem colaborativa.
- Desenvolva uma interface de usuário (por exemplo, um aplicativo da web) que interaja com a API do seu sistema de recomendação.

Esta tarefa desafiará você a aplicar seu conhecimento de computação distribuída, análise de dados e desenvolvimento de API para construir um sistema prático e bem interessante.

Entregas:

1. Documentação do sistema desenvolvido
2. Todo o código desenvolvido – toda entrega será pelo Classroom do GitHub.
3. Relatório de desenvolvimento e evidências da execução, contendo também o detalhamento das contribuições de cada membro do grupo, **em PDF**.