Movie Recommender

Technology Review

Priyanka Bijlani, Sharmeelee Bijlani,
Laura Thriftwood, Lakshmi Venkatasubramanian

Project Background

When considering which movie to watch, users have access to an overwhelming number of options. Users want custom recommendations to ensure optimal use of their time watching content. Business models benefit from strong recommender systems by increasing user engagement and addiction to streaming platforms.

With this project, we can create our own movie recommendation system that takes user input of one movie and utilizes a rich dataset of movie titles, ratings and user information to output a recommended movie.

Datasets

Streaming Data

1. MovieLens | GroupLens
    a. Over 100k ratings
    b. 1700+ movie titles
    c. 1000+ users

2. Kaggle
    a. https://www.kaggle.com/ruchi798/movies-on-netflix-prime-video-hulu-and-disney
    b. Scraped from IMDB
    c. 16000+ movie titles
    d. Rich feature set
    e. IMDB rating
    f. Rotten Tomatoes rating
    g. Language
    h. Streaming platform (Netflix, Prime, Hulu, Disney+)

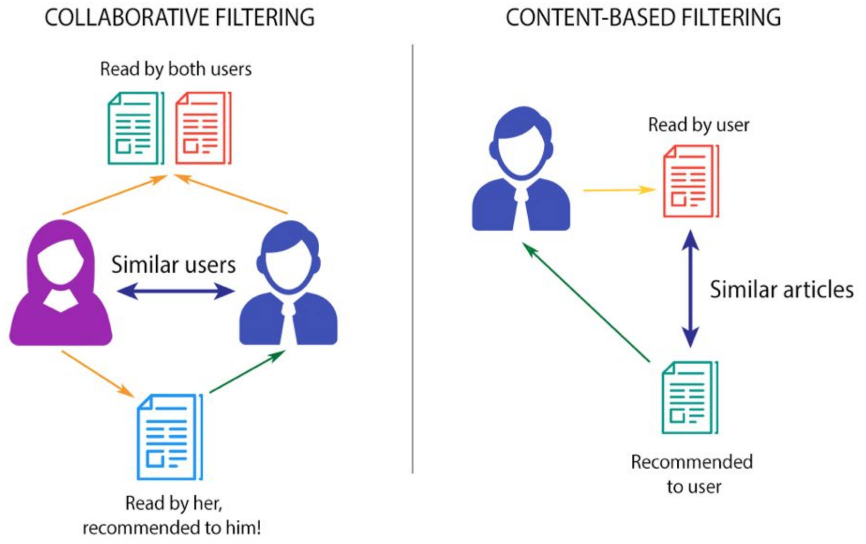
Use Case

1. Give top recommendation
   a. Training input: users, movies, ratings
   b. User input: user ID
   c. Outputs: movie
   d. ML algorithm: Collaborative filtering

2. Find similar movie
   a. Training input: users, movies, ratings
   b. User input: movie
   c. Outputs: movie
   d. ML algorithm: Collaborative filtering (analyzes historical data)

Recommender Systems Overview

- Seeks to predict a user’s reaction to an item (e.g. a movie)
- Requires a dataset that contains a set of items and a set of users who have reacted to some of those items.
- A “reaction” can be explicit (e.g. numerical rating, like/dislike) or implicit (e.g. adding item to a wishlist, spending time viewing an item)
- Recommender systems are broadly classified into two types based on the data being used to make inferences:
    - Content-based filtering, which uses item attributes.
    - **Collaborative filtering,** which uses user behavior (interactions) in addition to item attributes.



1*PfgYi6hg02TDM0GknVa2nQ.png (975×597) (medium.com)

Collaborative Filtering Overview

- Collaborative filtering filters information by using the interactions and data collected by the system from other users. It's based on the idea that people who agreed in their evaluation of certain items are likely to agree again in the future.
- The similarity of items is determined by the similarity of the ratings of those items by the users who have rated both items.
- A purely collaborative filtering approach does not consider any information about the user to make recommendations, similarity is calculated only on the basis of the rating a user gives to an item.
- Collaborative methods are typically worked out using a utility matrix. The task of the recommender model is to learn a function that predicts the utility of fit or similarity to each user.

Packages/Libraries

Recommendation System packages

SciPy/Implict
https://docs.scipy.org/doc/scipy/reference/api.html#guidelines-for-import
ing-functions-from-scipy
https://implicit.readthedocs.io/en/latest/

Surprise
https://surprise.readthedocs.io/en/stable/

Technology Comparison - Surprise

Pros

- **Surprise is a Python scikit for building and analyzing recommender systems that deal with explicit rating data.**
- Provide various ready-to-use prediction algorithms such as baseline algorithms, neighborhood methods, matrix factorization-based (SVD, PMF, SVD++, NMF), and many others.
- Also, various similarity measures (cosine, MSD, pearson…) are built-in.
- Provide tools to evaluate, analyse and compare the algorithms' performance
- Limited documentation/examples
- Accepts dataframe directly as input to the model

Cons

- Supports limited set of algorithms
- Does not work well with sparse data
- Limited documentation/examples
- Declining popularity
- Lower developer activity

Technology Comparison - SciPy / Implicit

Pros

- **General ML package with various algorithm and statistical functions**
- Scipy implicit is a fast Python Collaborative Filtering for Implicit Datasets
- Can predict ratings and compute similarity with the same model
- Computes implicit feature set that can be used with other packages
- Detailed reference documentation
- Lots of usage examples across various forums
- All models have multi-threaded training routines, using Cython and OpenMP to fit the models in parallel

Cons

- Requires managing dependencies
- Expects a sparse matrix as input and requires the dataframe to be transformed before feeding it into the model
- Lower code quality compared to Surprise

References

- scikit-learn vs Surprise | LibHunt
- https://realpython.com/python-scipy-cluster-optimize/
- https://builtin.com/data-science/collaborative-filtering-recommender-system
- https://rubikscode.net/2020/04/27/collaborative-filtering-with-machine-learning-and-python/
- https://realpython.com/build-recommendation-engine-collaborative-filtering/
- https://realpython.com/python-scipy-cluster-optimize/
- https://towardsdatascience.com/various-implementations-of-collaborative-filtering-100385c6dfe0