Methodologies Movie Recommendation System

Technologies stack

Front-end:

* Typescript
* React
* Babel + webpack (packaging)

Backend:

* Python + Javascript/TypeScript
* Microservice architecture
* Kubernetes + istio (service mesh)

Container orchestrator for easily managing and spinning containers on demand. Istio service mesh to ease inter service communication eg: request retry, service discovery.

* MongoDB (cloud hosted Atlas instance) + mongoose ORM
* Docker
* Cookies for user recognition instead of or until user decides to register.
* Elastic search for powerful search queries on movie data and metadata.

Deployment:

* Circleci

Continuous integration & continuous delivery pipeline

Recommendation algorithms:

1. Matrix factorisation using SVD (single value decomposition) or ALS (alternating least squares). Currently the most widely used recommendation method. It involves decomposing the user-movie interaction matrix into the product of two smaller matrices. One being the user with latent features and similarly the other for movies.
2. Neural network model with user, item, metadata, implicit interactions, embeddings

It’s possible to represent user/item data through one hot encoding method resulting in continuous vectors that can be further fed into a deep learning model.

eg: [user1, user2, user3] => user1 = [1,0,0]

A major disadvantage of this method is that every item will be totally dissimilar with another.

Embeddings are another way of representing categorical variables as continuous vectors. They also have the added benefit that similar items are positioned close to each other in the resulting latent space. For this reason they will be particularly useful for the problem of recommending movie.

For the purpose of this project, the embeddings creation will involve training an autoencoder on the ratings from the movie lens dataset. The result as stated above will be user and movie embeddings comparable in their respective latent space.

Evaluation:

1. Evaluate algorithm performance on the movie lens datasets 70-30 train-test split. Depending on the complexity and resource constraint full dataset will be used or a fraction of it.
2. Get actual users to interact with the system. Especially measure performance off implicit interactions (eg. User clicks on movie and spends x amount of time reading about it or follows link to movie trailer). Also evaluate overall system usability.

Architecture diagram



User flow diagram



Papers / articles:

1. <https://storage.googleapis.com/pub-tools-public-publication-data/pdf/45530.pdf>
2. <https://medium.com/@iliazaitsev/how-to-implement-a-recommendation-system-with-deep-learning-and-pytorch-2d40476590f9>
3. <https://towardsdatascience.com/prototyping-a-recommender-system-step-by-step-part-2-alternating-least-square-als-matrix-4a76c58714a1>
4. <https://papers.nips.cc/paper/1861-algorithms-for-non-negative-matrix-factorization.pdf>
5. <https://arxiv.org/vc/arxiv/papers/1603/1603.04259v2.pdf>