

Project Report :

Milestone 1:
Recommendation Model &
First Deployment

Github Link :

<https://github.com/cmu-seai/group-project-s23-The-hangover-Part-ML>

Ameya Morbale
Ankit Kumar
Prakhar Pradeep
Praveen Ramesh
Kedi Xu

Learnings

With the advent of technology, movie recommendation systems have become increasingly popular. Currently, several methods are used to develop movie recommendation systems, such as collaborative filtering, and content-based filtering. Collaborative filtering utilizes data from multiple users to make recommendations, while content-based filtering focuses on analyzing the properties of movies to generate suggestions. In our project, we explore two collaborative filtering based models and deploy it in a virtual environment, available to serve recommendations via API query. We used collaborative filtering because it can handle the cold start problem and provides flexibility on the user's perspective and their preferences.

Data

The data used for building the recommender system is collected from the Apache Kafka stream. After connecting with the Kafka broker server, the data is read from the broker. The stream has the user's watch history and movie rating data. However, to build the model, only the rating data is filtered out by string match and dumped in a "kafka_log_sample" CSV file. The data contains 1014609 entries of ratings, in a CSV file is roughly 52 MB. Finally, the data is split into train and test sets (75% for training and 25% for testing). Model evaluations are carried out on the test set

ML Techniques

1. Singular Value Decomposition ++

SVD++ is an extension of the classic Singular value decomposition - matrix factorization method for collaborative filtering. Apart from learning the normal user-factor and movie-factor matrices, SVD++ also considers the implicit ratings, i.e., for every user and movie, SVD++ learns two bias terms. These biases account for both a user's tendency to rate movies higher or lower than the average and the characteristics of a given movie that affect its tendency to receive high ratings. This is the reason for choosing this algorithm. The model is implemented through the [Surprise Scikit](#) package. As can be seen here <https://github.com/cmu-seai/group-project-s23-The-hangover-Part-ML/blob/main/MovieRecommendationFinal.ipynb>, the model's object is created and fitted with the training data for learning

2. Co-Clustering

Co-clustering can be seen as a method of co-grouping two types of entities simultaneously, based on similarity of their pairwise interactions. The reason for choosing this algorithm is that it exploits the inherent duality between users & movies making it possible to enhance the clustering along both dimensions, by using the information contained in column clusters during row assignments and vice versa. The model is implemented through the [Surprise Scikit](#) package. As can be seen here <https://github.com/cmu-seai/group-project-s23-The-hangover-Part-ML/blob/main/MovieRecommendationFinal.ipynb>

Model Comparison

Metric name	Model 1 - SVD ++	Model 2 - Co-Clustering
Prediction Accuracy(RMSE)	0.683	0.798
Training Cost (seconds)	7.74 s	39.3 s
Inference Cost (millisecond)	40.7 ms	42 ms
Disk Space (MB)	119.5 MB	39.7 MB

Model Implementation : Follow the Headers

<https://github.com/cmu-seai/group-project-s23-The-hangover-Part-ML/blob/main/MovieRecommendationFinal.ipynb>

1. Prediction Accuracy (Cell 6 & 9 under respective algorithm headings)

Metric : Root Mean Square Error

Data : Given data is split into test and train data. The accuracy of the model is evaluated
On held-out test data

Operationalization : The model with least error and highest accuracy is chosen for
Deployment

2. Training Cost (Cell 6 & 9 under respective algorithm headings)

Metric : Time required for model to train its parameters on train dataset

Data : Given data is split into test and train data. Cost is calculated w.r.t. train data

Operationalization : Model training code reports training time of potential models on full
training dataset

3. Inference Cost (Cell 8 & 10 under respective algorithm headings)

Metric : Prediction Time

Data : Calculate the time to run the prediction function from initialization to results

Operationalization : Inference cost can be averaged over N predictions to give average
prediction time

4. Disk Space (Memory on PC)

Metric : Storage space required by trained model

Data : Document the space on disk required to store all model parameters and its
attributes

Operationalization : Convert model to pickle format and monitor its disk space

(Comparing both models, we use SVD++ because it generates more accurate prediction)

Prediction service

Backend

To deploy the service we used Flask. Flask is a simple but robust backend framework, abstracting away a lot of low level logic for deployment. The server is hosted on our team's Linux machine. To deploy the server we used only python for now - due to the ease and simplicity. We run the flask app on "0.0.0.0". We used 'nohup' which lets the process run in the background. We realize that this is not the best way to deploy the server and we will change it in the future iteration.

TERMINAL COMMAND: curl -Is http://128.2.205.109:8082/recommend/12345

On initialization, the server loads the trained recommendation model as well as a list of movies from the disk – both are saved as a pickle file. When users send a http call to the server on "/recommend/<userid>", the server parses the URL for the user id which is passed on to a helper function that performs model inference and ranking and outputs movie recommendations. In addition to the recommended movies, a status code of 200 is returned.

CODE: <https://github.com/cmu-seai/group-project-s23-The-hangover-Part-ML/blob/main/app.py>

Recommendation

The recommendations function takes in two arguments (userid and list of movies) and outputs a list of top 20 recommendations as a comma separated string of movie ids. The function performs model inference on each movie id for the user. The model inference returns a float indicating the likelihood of the user recommending the movie to the user (rating). A tuple of movie id and predicted rating is stored in a list (recommendation list).

CODE: <https://github.com/cmu-seai/group-project-s23-The-hangover-Part-ML/blob/main/app.py#L28-L43>

Ranking

For ranking, a rather simple strategy was used for now and we aim to further improve this in future. The recommendation list is sorted on the rating in the descending order. We take the first 20 items in the list. These 20 items are converted into a comma separated list of movie ids.

CODE: <https://github.com/cmu-seai/group-project-s23-The-hangover-Part-ML/blob/main/app.py#L28-L43>

Team process and meeting notes

Setting up communication channels

After the first introduction in class, team decided upon two information channels

1. WhatsApp for casual communications like setting meeting times and small progress updates
2. Slack for official communication like sharing github ids as well as sharing other official team related information.

Meetings

After setting up the communication channels, the team decided on an introductory meeting. The agenda of the meeting was as followed:

1. Working style
2. Expectations from project
3. Background and proposed project roles

This helped in setting expectations for the project from each team member's perspective. This meeting also helped define roles and responsibilities for each team member. Below are the key roles and responsibilities for each member of the team:

Prakhar – Integration and backend

Ankit – Backend and Deployment


Kedi – Deployment and DevOps

Ameya – Project management and machine learning engineer

Praveen – Data scientist

Kedi started and managed the shared git repo. Praveen and Ameya worked on developing the recommendation system and exporting it into a pickle file. Prakhar and Ankit developed the backend infrastructure. Prakhar integrated the machine learning model in the infrastructure. Ankit and Kedi deployed the backend service on the production server.

Example of meeting notes:

**Ameya Morbale** 8:03 PM

Minutes of the Meeting - 13th Feb,2023

- Discussed code to get data stream and user/movie data; Simple user, movies and ratings could be used to build matrix based recommendations. Additional user and movie details available too (eg. age, gender, imdb etc) **@Prakhar Pradeep** could you suggest how to incorporate these additional details into a recommendation systems? I have only worked on 2D ratings matrix methods
- Discussed timeline - get one quick an dirty model out ASAP and get the entire pipeline running and working; Weekend should be spent finalizing & finishing the milestone and writing the final report

Action Items

- **@Praveen Ramesh @Ameya Morbale**: Process data and build simple model first. Send a .pkl file to dev ops/backend team to work with
- **@Ankit Kumar @Kedi Xu**: Explore how a pre-trained model(.pkl file) can be deployed; read the milestone document on github; they have suggested a few things in 'model inference' section. I guess the expectattion is that when they generate an api request (with a movie) our service had to serve that request with 20 or more recommended movies
- **@Prakhar Pradeep** Can you look into the more complex models? Which can incorporate additional user and movie features? And help us figure out what other models could we build apart from the matrix factor methods. You can also parallely help **@Kedi Xu & @Ankit Kumar** for deployment
-

(edited)

Work delegation

Work delegation was performed during meetings. We broke down the assignment into smaller checkpoints and delegated tasks for that checkpoint. For example our first checkpoint was to set up the git repository, production server and development server, as well as decide the stack. Each member of the team was responsible for two tasks. This redundancy allowed us to enhance collaboration as well as increase workflow efficiency – members could help each other. Breaking down the assignment into checkpoints helped us keep track of progress better and redirect man-power to certain parts of the project more than others.