Final Report

Tianye Song(tianyes), Udaikaran Singh (udaikars), Luís Gomes (lfgomes), Liangwei Zhang (liangwez), Tushar Vatsa(tvatsa)

May 2022

1 Reflection on Recommendation System

1.1 Machine learning model

1.1.1 Challenges

For Machine Learning models and algorithms, we used the simple K-Nearest Neighbors (KNN) model and Singular value decomposition (SVD) from Surprise library during the project process. Although it ensures a smooth latency for timely updating and retraining, the data capacity of those models are not quite satisfying. Therefore, in practice, we only retrain the model based on rating data within the last two weeks, and we also disregard the contributions from other features, including the type of movies and different attributes of movies such as age, gender and occupations. We believe this issue is likely related to the use of the Surprise dataset, which builds off Scikit-learn. We alleviated this process by down-sampling our training data in order to meet the hardware constraints.

Also, the data versioning is a memory exhausting process. We used Data Version Control(dvc) to store the data on the google drive. dvc generates a pointer file which points to the actual data storage space and its a lightweight file and can be easily stored on GitHub. The data was entirely stored every time the dvc command is run. A better strategy can be used here, such as only storing the changes in the dataset over time to reduce redundancies.

A challenge we faced during our online evaluation was that we did not have a statistical background to make design decisions for our t-tests. To fix this, we looked towards similar implementations of statistical tests on recommender systems. We used these as templates for our AB testing pipeline design. We also made an assumption that a recommendation is considered positive if the user watched the movie within the 5 hour period of the recommendations. This definition of a positive recommendation is limiting and inappropriate. In expanding our AB testing, we would look for more long-term impacts of the recommendations in order to make a more fair comparison of our models.

1.1.2 Improvements

An improvement we can make to our machine learning modeling is using machine learning libraries that can support more data. In our code, we using the Surprise Library, which is using Scikit-learn for recommend systems. This can be improved by switching over to a library like TensorFlow. TensorFlow would allow us to test more computationally complicated models, such as neural networks. TensorFlow would also allow us to utilize hardware like GPU's and train over distributed machines; this would allow for faster and more scale-able retraining of our model.

We could also look to improve our models by incorporating more of the user data into our model for our recommend-er system. In the current models, we primarily used ratings data in our model. We could make our model more expressive by utilizing the user data, such as age, gender, etc. We could also look to integrate outside data, such as movie information in our recommend-er system. With more time and more resources, we may get a more powerful platform and computing capabilities to train a more complicated models which takes in more features and attributes with a acceptable latency and training cost in production.

1.2 Database

1.2.1 Challenges

At Milestone 1, we needed to make the choice about how to store the data. We could use CSV file, SQL database or unstructured database and we eventually chose to use MongoDB. Because MongoDB could store data in a more flexible way.

An initial challenge we had during the first and second milestone was debugging the integration of the Kafka stream and our database. We initially had typos and inconsistencies between how we read in our data and managed the MongoDB database. However, we were able to fix these inconsistencies by the end of the second milestone. And in the long term, this saved us many time dealing with the data collection which is important to our project.

Another challenge we faced in our second milestone errors in our training data that we used in the first milestone. Some issues we addressed in our data cleaning pipeline. For instance, we adjusted the data schema to allow for data values in features that are not included in our read-in data. We did some basic upkeep, like removing duplicate entries in our data-set and removing extra white spaces in our incoming string data. We also performed a basic check on whether the incoming data reflects reasonable values, such as checking if the rating is an integer within the range of 1 to 5, and checking if the data within the incoming stream of data is valid.

1.2.2 Improvements

One thing that we could improve for the database is that we should define the collection we will use and their purpose and data structure at the very beginning. Because we added collections and changed the data structure when there are more tasks added in. For the input data, the might be data issues related to the type of the data and we didn't specify that at first so we need to change the data in the database each time we found some data type issues which is not efficient and proper. If it is possible, we should think more about the data structure and quality issues when we define the structure and data cleaning standards for the database.

There is also some potential instability lying in our data cleaning pipeline. It is difficult for us to consider all possible data drift or error inputs in one time. Therefore, if we plan to deploy the recommendation service at scale in production, additional investment on continuous monitoring data inputs and in process serves as a pivotal role to ensure the safety and performance of our recommendation service.

2 Reflection on teamwork

Teamwork is when a group of people work together toward a common goal or purpose. If each person willingly and intentionally makes the team's interests and objectives their first priority, work reaches heightened levels of success—and the results can make a big impact. A challenge we had in our teamwork was forming meetings given the different schedules of the team members. This is especially true because our team is composed of people from differing programs in CMU, and a mixture of PhD and Masters students. This level of disparity made it difficult at times to coordinate meetings. We looked to alleviate this issue by using the website https://www.when2meet.com in order to more easily pick meeting times for the most of team members. We will not hesitate to say that this is one of the most diverse groups we have ever worked with. We all come from different programs, we all have different experience in software and machine learning, in fact we all come from different parts of the world. So, we tried to spend some time with our teammates in person to establish a good working environment for all of us. And by the end of milestone 1, we were comfortable working with each other and sharing our positives and negatives.

For the rest of the milestones, we established a formation in the team. For example Luís was working as a project manager. Based on the person's strength or weakness, he used to assign task. Tianye made sure that everything we plan has a deadline so that we can work efficiently on it. And Tushar, Liangwei and Udai made sure that we deliver everything on time. We would plan things out much before the deadline so that people can volunteer to work on a certain part based on their work load in other courses. The best thing was that any personal issue was taken care of without other team members having a hint of it and all of us collaborated to make it successful.

The positive aspect to our teamwork was our use of technologies in order to organize our workflow. We primarily communicated and organized our sub-tasks over Slack and Trello; we set up channels for the major sub-problems of our project, such the modeling, Kafka, etc. We also maintained our code over GitHub. Over the course of the project, we improved on our use of git, as we started to utilize features like branches and pull requests in order to organize our code production. Lastly, we all worked on a similar IDE (VS Code) in our code production; this allowed us to more easily debug each other's code during development.

Every team work comes with its own set of challenges and we too had a fair share. For example, everyone was having different work load and it sometimes led to an unequal distribution of work. Also, it was difficult to predict the workload of different parts beforehand. I personally feel that we could have attended more TA sessions to clarify our doubts because we spent quite some time thinking on those lines. Moreover, thorough communications and understandings, as well as flexible arrangements on tasks can also help relieve such issues. For instance, when there is something wrong occurred in the infrastructure or model, we can also help by analyzing possible sources of errors from our own parts.

An improvement that we could implement going into the future is actively performing offline evaluation to determine if the models we are training are fairness.

3 Presentation

Link to final presentation file:
https://github.com/cmu-seai/group-project-s22-dsu/blob/master/MLProd-Final.pdf
Link to final presentation recording:
https://drive.google.com/file/d/1aGeLZS8mzel1O2rYGb5cn1wRM7koDL3h/view?usp=sharing