Amazon Electronics Products Recommender System

Team: Jingxuan Bao, Yaoqi Deng, Guanlin Huang

Project Mentor TA:

1) Introduction

In summary for our works until now, we aim to create a sophisticated machine learning model to build an Amazon electronics product recommender system using collaborative filtering techniques. The dataset used for training and testing the model is an extensive collection of Amazon electronics product reviews, with 7,824,482 entries, including user IDs, product IDs, ratings, and timestamps. Follow the Kaggle dataset link at https://www.kaggle.com/datasets/pritech/ratings-electronics.

To ensure high-quality recommendations, the dataset is filtered to only include users with at least 5 ratings and products with at least 20 ratings. Exploratory Data Analysis was also implemented to the filtered data, for us to view the distribution of product/user average ratings, and the time series features of the dataset were also visualized.

We implemented a collaborative filtering recommender system for Amazon electronics products using three algorithms, KNNBasic, KNNBaseline, and Singular Value Decomposition (SVD). The KNNBasic and KNNBaseline were constructed using k-nearest neighbors with k=10 and cosine similarity, focusing on finding similar items based on user preferences. SVD was factorized based on user factors and item factors to obtain two lower-dimensional matrices and use the dot product between the user's factor vector and the item's factor vector to predict the ranking of items for recommendation. Due to the computation cost, until now we were only able to use default parameters for SVD training.

We assessed the performance of all three models using three accuracy measures: Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE). These metrics were employed to evaluate the differences between the predicted ratings and actual ratings in the test set. By comparing the error metrics, we were able to determine the effectiveness of each model in generating accurate recommendations.

2) How We Have Addressed Feedback From the Proposal Evaluations

After we observed the original rating data distribution, we decided to limit our data to users with at least 5 ratings and products with at least 20 ratings. After filtering, the dataset contains 1,828,344 entries, which allows for a more accurate representation of user preferences and product popularity. The exploratory data analysis of filtered data reveals 253,613 unique users and 57,586 unique products.

To make our project a workable schedule, we first tried to train some baseline models like KNN and SVD. These models could still provide acceptable performance, but we still want to use more powerful and complex models then. We already discussed the workability of Neural Networks [1], and also an LSTM based model to capture the timestamp features that we ignored before [2]. We believed these models could make better performance than the results we already achieved, and also provide a more advanced idea in this area over those already existing research in Kaggle.

3) Prior Work We are Closely Building From

A. Kaggle submission: "Amazon Electronics Products Recommender System". URL: https://www.kaggle.com/saurav9786/recommender-system-using-amazon-reviews. This Kaggle kernel provides an example of a recommender system using the Amazon electronics dataset and implements user-based collaborative filtering with the KNNWithMeans algorithm. The implementation can serve as a reference for our project. Code link: https://www.kaggle.com/saurav9786/recommender-system-using-amazon-reviews/code.

B. Kaggle submission: “Product Recommendation Systems”. URL:https://www.kaggle.com/code/pritech/product-recommendation-systems. This Kaggle submission provides a more detailed implementation for SVD, KNN, and KNNWithMeans. The author provides an item popularity based recommendation, but the author also points out that simply ignoring the user behavior is not a good choice. Code link: https://www.kaggle.com/code/pritech/product-recommendation-systems/code.

4) Contributions

4.1 Implementation Contributions

We already discussed KNN, KNNBaseline, SVD, neural network, and LSTM models. We plan to experiment with different models to identify the best-performing one. KNN and SVD models have been already trained using the user-item matrix, while the neural network will be implemented using TensorFlow.

Employing cross-validation and Grid Search to tune hyperparameters: We aim to fine-tune the hyperparameters of each model to optimize their performance. We plan to employ cross-validation to evaluate each model's performance and use Grid Search to identify the best hyperparameters.

Comparing the performance of the models: After fine-tuning the hyperparameters, we will compare the performance of each model. We will evaluate the accuracy, precision, recall, and F1 score of each model to identify the best-performing one.

4.2 Evaluation Contribution

Comparing the KNN and KNNBaseline we already implemented, The KNNBasic model achieves an MAE of 0.8987, MSE of 1.6284, and RMSE of 1.2761. The KNNBaseline model, which incorporates user and item biases, shows improved performance with an MAE of 0.8565, MSE of 1.4343, and RMSE of 1.1976. The lower error metrics for the KNNBaseline model suggest that accounting for user and item biases in the collaborative filtering model enhances the overall recommendation quality.

Our preliminary study used the default parameters for SVD model training. Based on our training with three cross-fold validation, we obtained a result with an RMSE score of 1.123 and the MAE score of 0.840.

For the KNN and KNNBaseline model fit implementation, the RAM cost was over 60 GB, based on this memory requirement, we upgraded our SageMaker based machine to 128 GB RAM. Additionally, We have tried to use grid search to obtain an optimal parameter setting for all these models. However, the search method we implemented was too much computation consumption for us, this is a challenge we are working on to overcome recently.

We started implementing a neural network using TensorFlow. The main model we plan to use is a multi-layer perceptron, with ReLU serving as the activation function. For the output layer, we will use a NeuCF layer that takes in the concatenated output of both the MLP and MF models and feeds this input to a Dense neural network layer. By combining these models in this way, we hope to achieve the desired characteristics of each model, ultimately generating predictions for user-item pairs and determining which items to recommend.

However, we have encountered some difficulties while implementing it. Specifically, we have had an ambiguous bug that may be related to memory limitations. We are currently investigating this issue and considering alternative strategies to optimize the memory usage of the mode.

5) Risk Mitigation Plan

If our approach does not work, we will need to consider alternatives to turn in a useful project report. One option could be to evaluate the reasons for failure and identify specific examples where our approach works better, even if it does not perform well on the full dataset. Another option could be to try our algorithm on different, simpler data such as a "toy" synthetic dataset we generated.

One downside of our current SVD model is ignoring the temporal factor. There are a few ways that we are planning to do to take review time into consideration. First, we can use time-based weighting, which assigns more weight to recent views compared to older ones. We can also perform trend analysis on product popularity or user behavior to identify if there is any trend for a certain product at a certain time of the year that might have a higher review. This might help to make predictions based on factors like seasonal products.

When putting time into consideration, Long Short-Term Memory (LSTM) networks are an interesting approach for our recommendation system design. It is a type of RNN that is well-suited for processing sequential data, which is theoretically a better model compared to the SVD which lacks temporal factors. For the next steps, we will implement and evaluate the performance of LSTM, which can also help us understand better the importance of timestamps in evaluating product reviews.

Other Prior Work / References:

[1] Neural Collaborative Filtering for Deep Learning Based Recommendation Systems | Architecture Breakdown & Business Use Case | Width.ai. (n.d.). Retrieved April 4, 2023, from https://www.width.ai/post/neural-collaborative-filtering

[2] Zhao, C., You, J., Wen, X., & Li, X. (2020). Deep Bi-LSTM networks for sequential recommendation. *Entropy*, *22*(8). https://doi.org/10.3390/E22080870