Recommendation system and sentiment analysis based on Amazon review and user data

Jinsong Zhen, Jinlin Ye, Junyang Tang, Jianing Li
Georgia Institute of Technology
Atlanta, GA
{jzhen3,jinlinye,cartert,jli3360}@gatech.edu

Abstract

This research focuses on creating an advanced recommendation system and model using the Amazon Fine Food Reviews dataset from Kaggle. By employing techniques such as Singular Value Decomposition++ (SVD++), hybrid collaborative filtering, and neural collaborative filtering, our goal is to generate tailored, personalized recommendations for users. Ultimately, this project aims to improve e-commerce sales and user experience through a deeper understanding of user preferences and enhanced personalized product recommendations to boost sales.

1 Introduction

Our study seeks to create an advanced recommendation system for e-commerce platforms to improve product recommendations based on the Amazon Fine Food Reviews dataset from Kaggle. Traditional methods, such as collaborative filtering and Singular Value Decomposition (SVD), struggle with sparse data, scalability, the cold start problem and the impact of inconsistent data on overall performance and user experience. Sparse data and scalability issues hinder accurate, real-time recommendations, while cold start problems limit personalization for new users or items. To tackle these challenges, we propose using hybrid collaborative filtering, Neural Collaborative Filtering (NCF) and SVD++ to handle the limitations of conventional approaches, by including implicit features, and get rid of the cold start problem to boost personalization and offer relevant product recommendations. Finally, SVD++ achieves the smallest RMSE but NCF is limited by the computational resources which may degrade the performance, so we believe that the NCF will give us a better result if we have a higher computational ability.

This study can have impacts on e-commerce businesses and customers. E-commerce platforms can profit from higher sales and better customer retention while customers will appreciate a personalized shopping experience with tailored recommendations. An effective project will significantly decrease the influence of inconsistent reviews, improving consumer satisfaction and industry expansion.

2 Literature Survey

Collaborative filtering is the most popular technique to analyze user-item interactions to find patterns and provide personalized recommendations. They include memory-based methods, such as user-based and item-based CF, and model-based methods, such as matrix factorization and deep learning models like NCF.

We cover two important publications that are pertinent to our project, on which we developed our methods to create a recommendation system model for the Amazon Fine Food Reviews dataset, in this literature review.

Koren [5] introduces the SVD++ algorithm, a revision of the original matrix factorization approach for collaborative filtering that incorporates the user's previous preference data to take implicit feedback. Koren demonstrates that, compared to conventional SVD and neighborhood-based collaborative filtering techniques, SVD++ significantly improves recommendation accuracy. Although the paper has effectively enhanced performance using Netflix's dataset, Koren acknowledges that further experimentation with more reliable sources of implicit feedback, such as purchase or rental history, is necessary.

He et al. [3] propose a neural collaborative filtering (NCF) approach that combines the strengths of collaborative filtering and deep learning. When compared to conventional matrix factorization methods, the NCF approach exhibits greater expressive capacity. This approach directly contributes to our project's objective of using sophisticated collaborative filtering techniques to generate tailored recommendations. In order to provide a more thorough understanding of customer preferences and improve recommendation quality, our project intends to adapt and combine both methods—SVD++ and NCF—to the Amazon Fine Food dataset.

The limitations of the SVD++ and NCF approaches include the cold start problem, where providing accurate recommendations for new users or items with limited interaction data is challenging. Both methods may also face scalability issues when dealing with large datasets, as they require substantial computational resources. Also, the complexity of implementing and fine-tuning deep learning models like NCF can be daunting.

3 Dataset Description

3.1 Source

We acquired the dataset from Kaggle, where it is hosted under the title "Amazon Fine Food Reviews." The dataset was originally collected and compiled by Julian McAuley and Jure Leskovec [6] from Stanford University for their research paper "From Amateurs to Connoisseurs: Modeling the Evolution of User Expertise through Online Reviews."

3.2 Data Preprocessing

The relatively clean dataset required minimal preprocessing. We performed Exploratory Data Analysis (EDA) and found no missing values, eliminating the need for imputation. We identified the helpfulness ratio as a valuable feature for collaborative filtering by regarding it as a weight feature, potentially improving recommendation quality by prioritizing more helpful reviews. Users with consistently high helpfulness ratios may be considered more reliable, resulting in better-personalized suggestions. We removed the "profile name" and "summary" columns as they held limited value for our tasks.

The existing Amazon fine food dataset is sufficient to achieve our goal, which aims to develop a recommendation system and sentiment analysis models. As for the recommendation system implementation, the dataset provides user-item interaction in form of reviews and ratings. techniques such as traditional collaborative filtering and neural collaborative filtering can be implemented to generate personalized recommendations based on this interaction data. What's more, the helpfulness ratio will help us identify more valuable reviews and weight reviews based on other users' perceived helpfulness.

As for the sentiment analysis task, the review texts accompanied by ratings will enable us to employ Natural Language Processing (NLP) techniques (For example, Tokenization, Stemming, Lemmatization) to analyze review text and discern user sentiment toward specific products.

3.3 Raw Data Statistics

Attribute	Value
#Users	256,059
#Items	74,258
#Ratings	~568k
Avg. Interaction	~2.22

Table 1. Summary statistics of the dataset

Note: # of sessions metric is not applicable for this dataset since there is no user interaction data that typically defines a session.

There are three quantitative variables for this dataset:

- HelpfulnessNumerator: Number of users who found the review helpful, with a mean of 1.74 and a standard deviation of 7.64. The range is from 0 to 866.
- HelpfulnessDenominator: Number of users who indicated whether they found the review helpful or not, with a mean of 2.23 and a standard deviation of 8.29. The range is from 0 to 923.
- Score: 1 to 5 scale rating, with a mean of 4.18 and a standard deviation of 1.31.

4 Experimental Settings Description

Our primary task is to predict user-item interactions in the form of ratings, thereby facilitating personalized recommendations to users based on their preferences and previous interactions with the products.

To assess the performance of our models, we used the Root Mean Square Error (RMSE) as the evaluation metric. RMSE is a widely used metric for rating prediction tasks, as it quantifies the discrepancies between predicted and actual ratings while accounting for the magnitude of errors. Lower RMSE values indicate better model performance, which in turn signifies a more accurate recommendation system.

The experimental settings will include dividing the dataset into training (80%) and testing (20%) sets, ensuring a fair evaluation of the models. We utilize 5-fold cross-validation to ensure robustness and minimize potential biases. The experiments will be run on Google Colab to ensure adequate computational resources. The available Random Access Memory (RAM) will be 12.7 GB, and we used Google Cloud Graphics processing unit (GPU) resources.

5 Methods

5.1 Hybrid Collaborative Filtering

Neighborhood-based collaborative filtering is one of the earliest and most popular methods. The predicted score is based on similarity between users or items, which is given by cosine similarity:

$$sim(i,j) = cos(\vec{i},\vec{j}) = \frac{\vec{i}\cdot\vec{j}}{||\vec{i}_2|| * ||\vec{j}_2||}$$

where i, j are item or user vectors.

Sarwar et al. [8] introduced item-based collaborative filtering recommendation algorithms in their paper. The item rating prediction for each user $P_{u,i}$ is calculated using weighted sum, given by:

$$P^I_{u,i} = \frac{\sum_{\text{all similar items, } I}(s_{i,I} * R_{u,I})}{\sum_{\text{all similar items, } I}(|s_{i,I}|)}$$

where s is similarity and R is rating.

Paul et al. [7] proposed a collaborative filtering algorithm to predict users' preferences based on the ratings of similar users, which is one of the earliest applications of user-user CF for recommending articles in Usenet newsgroups.

The user rating prediction for each user $P_{u,i}$ is calculated using weighted sum, given by:

$$P^U_{u,i} = \frac{\sum_{\text{all similar users, } U}(s_{u,U} * R_{U,i})}{\sum_{\text{all similar users, } U}(|s_{u,U}|)}$$

where s is cosine similarity and R is rating.

While our initial attempts at employing solely user-user based or item-item-based yielded unsatisfactory results, which we presume that it may due to the traditional weakness of Collaborative Filtering such as the cold start problem, data sparsity, and less tunability. So we developed a hybrid approach to enhance the recommendation system's performance.

To ensure that each rating is appropriately weighted, we incorporated the helpfulness rate as a multiplier for the corresponding ratings. This approach ensures that ratings deemed more helpful by users receive higher weight in subsequent calculations. The formula of the helpfulness weight W_h is as below:

$$w^h_{u,i} = \frac{h^N_{u,i}}{h^D_{u,i} + 10e^{-8}}$$

where $h^N_{u,i}$ and $h^D_{u,i}$ are HelpfulnessNumerator and HelpfulnessDenominator described in the dataset.

The adjusted rating $R^h_{u,i}$ for user u and item i is given by

$$R^h_{u,i} = R_{u,i} \cdot w^h_{u,i}$$

For the Simple Hybrid Method we developed, we applied the adjusted ratings and combined the user-user and item-item predictions by linearly interpolating them using a weight parameter (α):

$$R_{pred} = \alpha P^U_{u,i} + (1-\alpha) P^I_{u,i}$$

where the calculations of item-item predictions and user-user predictions are calculated similarly to the traditional neighborhood Collaborative Filtering.

In future research, we plan to fine-tune the alpha parameter to further optimize the recommendation system's performance which can resolve the less tunability of the traditional Collaborative Filtering. In addition, this hybrid method has complementary strengths over the sole ones since it can provide a more comprehensive understanding of user preferences and item relationships. It can also potentially alleviate the cold start problems by combining predictions from both methods.

5.2 Matrix Factorization: SVD++

To better achieve scalability and handle sparse data, we continue to look into SVD++, an extension of the Singular Value Decomposition algorithm for collaborative filtering. The goal is to exploit all available interactions between users and items by integrating implicit interactions, as well as user and item biases. SVD++ is an advanced matrix factorization method that provides a further indication of user preferences in addition to explicit ratings. This is especially beneficial for users who gave significantly more implicit than explicit input. [1]

The model first describes the general properties of the item and the user, without accounting for any involved interactions. The next step captures implicit ratings, which is the fact that a user rated an item, regardless of the rating value. Also, a bias component is included.

The predictive rating $\hat{r}_{ui}$ is given by:

$$\hat{r}_{ui} = \mu + b_u + b_i + q_i^T \left(p_u + |R(u)|^{-\frac{1}{2}} \sum_{j \in I_u} y_j \right)$$

μ is the global average rating across user-item pairs. A set of these factors are used to characterize users based on the set of items that they rate and prefer. b_u is a user bias term, accounting for user-specific deviations from the average. Similarly, b_i is the item bias term. p_u represents the user-factors vector and q_i is the item factor vector for item i. p_u is learned from the given explicit ratings. $|R(u)|$ or I_u is a set of items rated by user u. y_j is the implicit feedback factor vector for item j. $|R(u)|^{-\frac{1}{2}} \sum_{j \in I_u} y_j$ refers to the perspective of implicit feedback. The user bias b_u (the rate habitats of a user) is independent of the item's characteristics (a rate given for an item); vice versa, the item bias b_i is independent of the user interest. We represent users through the items that they prefer.

Since the y_j's are centered around zero (by the regularization), $|R(u)|^{-\frac{1}{2}}$ normalizes the contribution of the implicit feedback and therefore the sum to stabilize its variance over the full range of observed values of $|R(u)|$

The model parameters are learned and determined using gradient descent to minimize the corresponding regularized squared error function via stochastic gradient descent. We iterate over all available ratings. The following are the computational algorithms

$$b_u \longleftarrow b_u + \gamma \cdot (e_{ui} - \lambda_5 \cdot b_u)$$

$$b_i \longleftarrow b_i + \gamma \cdot (e_{ui} - \lambda_5 \cdot b_i)$$

$$qi \longleftarrow qi + \gamma \cdot (e_{ui} \cdot (pu + |R(u)|^{-\frac{1}{2}} \sum_{j \in R(u)} y_j) - \lambda_6 \cdot q_i)$$

$$pu \longleftarrow p_u + \gamma \cdot (e_{ui} \cdot qi - \lambda_6 \cdot p_u)$$

$$\forall j \in R(u): \; y_j \longleftarrow y_j + \gamma \cdot (e_{ui} \cdot |R(u)|^{-\frac{1}{2}} \cdot q_i - \lambda_6 \cdot y_j)$$

where γ controls the step size in gradient descent, $e_u i$ is observed error between the actual rating and the predicted rating for user u and item i, λ_5 and λ_6 are regularization parameters for user, item biases and implicit feedback factors.

Thanks to `Surprise` package [4] in Python, we can implement the SVD++ algorithm by using the package. By setting the appropriate values for the model parameters, the algorithm will automatically determine the relative relevance of each source of implicit feedback.

We chose SVD++ for recommendation systems because it solves data sparsity problems and gives high-quality tailored food recommendations. It is more tolerant of sparsity than nearest-neighbor models. Compared to latent factor models, SVD++ incorporates implicit feedback, which enhances the model's predictive power. SVDpp improves prediction accuracy by taking advantage of both near-neighborhood and latent factor approaches. It is sufficient for making food recommendations for customers for its ability to account for both global and localized user-item interaction information and shows a faster training time than neural models.

5.3 Neural Collaborative Filtering

The usage of a simple and fixed inner product to estimate complex user–item interactions in the low-dimensional latent space is one of the limitations of matrix factorization methods. Thus, we seek new methods for collaborative filtering.

He et al. [3] proposed a novel collaborative filtering framework based on deep neural networks. This research aims to address the limitations mentioned above by introducing a more expressive and flexible model.

The general NCF's predictive model is given as:

$$\hat{y}_{ui} = f(P^T v_u^U, Q^T v_i^I | P, Q, \Theta_f)$$

where P and Q are the latent factor matrix for users and items respectively, and v_u^U and v_i^I are the feature vectors for users and items. f is the multi-layer neural network, and Θ_f denotes the parameters.

The authors introduce two instantiations of the NCF framework: Generalized Matrix Factorization (GMF) and Multi-Layer Perceptron (MLP). GMF is a linear model, while MLP is a nonlinear model. The final model Neural Network Matrix Factorization (NeuMF), is a fusion of the two, while instead of sharing the same embeddings, Author proposed to concatenate the output of the two models before reaching the final layer to combine both latent features and capture non-linearity.

With p_u and q_i stand for user and item latent vector, the fused model is formulated as:

$$\phi^{GMF} = p_u^G \odot q_i^G,$$

$$\phi^{MLP} = a_L(W_L^T(a_{L-1}(...a_2(W_2^T \begin{bmatrix} p_u^M \\ q_i^M \end{bmatrix} + b_2)...)) + b_L),$$

$$\hat{y_{ui}} = \sigma(h^T \begin{bmatrix} \phi^{GMF} \\ \phi^{MLP} \end{bmatrix}),$$

where W_x, b_x and a_x denotes the weight matrix, bias vector and activation function respectively. h is the weight for the output layer. In our implementation, we use a uniform vector of 1 as h since we didn't have pre-trained GMF and MLP models. As shown, the proposed structure provided great flexibility for the model. We implement this method using `TensorFlow` and referencing the author's code repo at Github.[2]

6 Experiments and Results

Methods	RMSE
Hybrid CF	1.503
SVD++	1.017
NCF	1.066

Table 2. RMSE comparison of methods

6.1 Hybrid Collaborative Filtering

With

$$R_{pred} = \alpha \cdot P_{u,i}^U + (1 - \alpha) \cdot P_{u,i}^I$$

, we set alpha = 0.5, giving equal weight to both user-user and item-item predictions in the rating prediction. We applied a 70% train and 30% test split, and the RMSE reported 1.503.

6.2 Extension (++) of Singular Value Decomposition

Upon fine-tuning the hyperparameters, we adjusted the range of values in the parameter grid for each parameter. Through this optimization process, we identified the best set of parameters as follows:

- Factors: 50
- Epochs: 35
- Learning rate: 0.093
- Regularization term: 0.051

We used the training data (75%) and test data (25%) obtained from the train-test-split function applied to the users, items, and numeric ratings. The SVD++ model optimizes the performance was built using the following steps: First, we created a parameter grid to identify the best combination of parameters, such as the number of factors, number of epochs, learning rate, and a regularization term. Next, we applied the GridSearchCV function, which internally performed cross-validation during the hyperparameter search process. Finally, we utilized K-fold cross-validation to train and select the most suitable model for our dataset.

Utilizing these optimal parameters, the SVD++ algorithm achieved an RMSE of 1.017 on the selected test dataset. This demonstrates the effectiveness of the algorithm after careful hyperparameter tuning in providing accurate predictions.

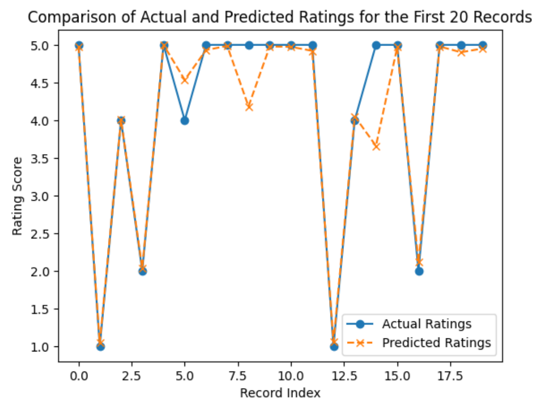

Figure 1. Comparison of the actual and predicted scores for the first 20 ratings from the Amazon Food Review Dataset

We observed that the predicted score for the 14th rating record significantly deviated from the actual value, which can be attributed to the cold start problem. To address this issue in the future, we could consider incorporating additional factors into the model, which would better handle new users and improve its predictive accuracy.

6.3 Neural Collaborative Filtering

From user ID and item ID, we generated GMF user latent vectors p_u^G and item latent vectors q_i^G at size of 8. For MLP, we generated p_u^M and q_i^M at size of 16, which makes a concatenated MLP input layer size of 32. We applied a tower structure (32, 16, 8) for the multiple layers.

Due to the overfitting problem we observed, the training process is early stopped at Epoch=4 to reduce overfitting.

Finally, the RMSE of NCF method's predictions is 1.066, which places it between collaborative filtering (1.503) and SVD++ (1.017). Despite not having the lowest RMSE, NCF still performs better than the traditional collaborative filtering method.

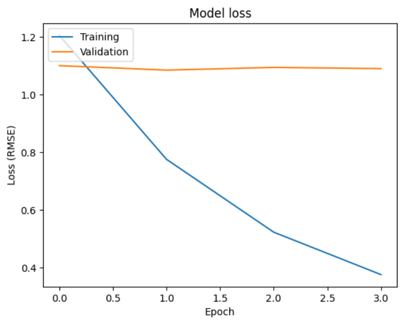

Figure 2. Training and Validation losses as #epochs increases

We observed overfitting problems exhibited by the NCF method, which is expected since we did not perform hyperparameter tuning. Such behaviors may arise due to suboptimal batch size or learning rate input. However, due to the size of our dataset and limited computational resources, we did not conduct hyperparameter tuning to explore potential ways to avoid overfitting. Nevertheless, we believe that NCF could yield better performance, as it can effectively model nonlinearity.

7 Conclusion

Our experimentation has provided us with valuable insights into the effectiveness of different collaborative filtering techniques in recommendation systems. Our findings indicate that while traditional CF offers simplicity and interpretability, it falls short in terms of performance. On the other hand, NCF shows great potential with its flexibility, but its complexity may lead to overfitting. SVD++ yielded the best results, suggesting it is capable to capture user-item interactions effectively in this dataset.

7.1 Shortcomings

Overfitting in the NCF model may be caused by poor hyperparameter values. Although the hybrid collaborative filtering approach makes an effort to address the cold start issue, there is still potential for progress. The performance of the recommendation system may be impacted by the hybrid collaborative filtering method's unoptimized alpha parameter. The text comment data in the dataset is underutilized, as more preprocessing is required to effectively incorporate this information into the recommendation system. Additionally, although the SVD++ algorithm attains high accuracy, its primary limitation is the relatively low computational efficiency and the inability to handle new users.

7.2 Future Directions

Utilize methods like grid search or random search to determine the best configuration for NCF hyperparameter tuning, including batch size, learning rate, number of layers, and number of neurons per layer. To better handle the cold start issue for new users or items, consider incorporating content-based approaches or other strategies like clustering. Find the ideal balance between user-user and item-item predictions by adjusting the alpha value in the hybrid collaborative filtering approach using cross-validation. To properly use this data in the recommendation system and maybe enhance the performance of the model, execute extra preprocessing on the text comment data.

8 Contribution

All team members have contributed a similar amount of effort.

References

[1] Ricii Francesco, Lior Rokach, Bracha Shapira, and Paul B. Kantor. 2010. $Recommender_systems_handbook. Springer, Bolzano, Italy.153 - -154pages.$

[2] Xiangnan He. 2017. Neural Collaborative Filtering. https://github.com/hexiangnan/neural_collaborative_filtering. GitHub repository.

[3] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*. 173–182.

[4] Nicolas Hug. 2020. Surprise: A Python library for recommender systems. (2020). https://surprise.readthedocs.io/en/stable/matrix_factorization.html

[5] Yehuda Koren. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 426–434.

[6] Julian J McAuley and Jure Leskovec. 2013. From amateurs to connoisseurs: modeling the evolution of user expertise through online reviews. In *Proceedings of the 22nd international conference on World Wide Web*. 897–908.

[7] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. 1994. GroupLens: an open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work*. 175–186.

[8] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*. 285–295.