Recommendation System for Amazon Fine Food Products

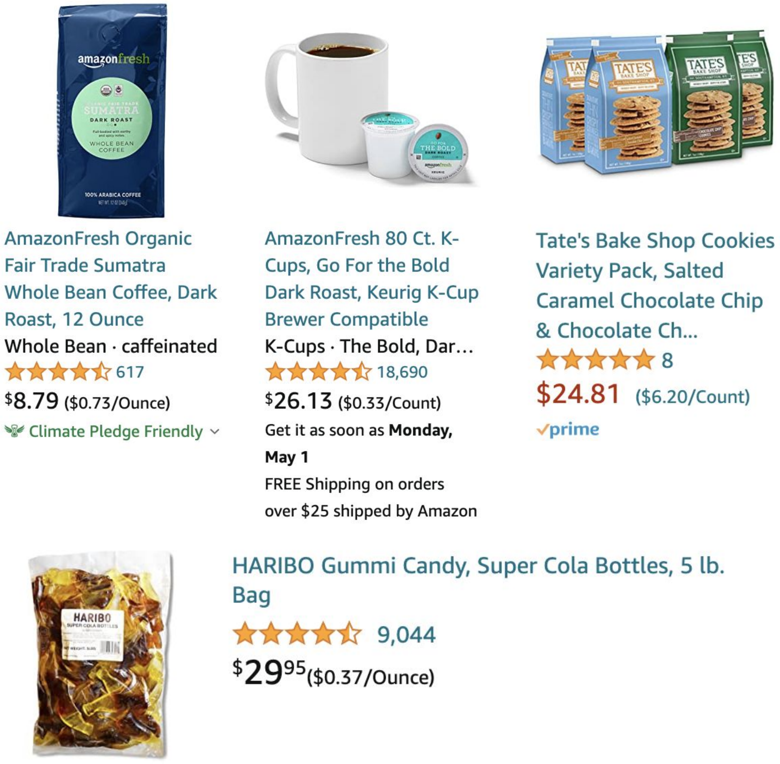
CSE6240 Final Project

Jinlin Ye, Junyang Tang, Jianing Li, Jinsong Zhen

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 like collaborative filtering(CF) and Singular Value Decomposition(SVD) have their own limitations which struggled by sparse data, scalability, the cold start problem, so we implement the SVD++ and Neural collaborative filtering to boost personalization and offer relevant product recommendations.



A peek into Amazon's fine food category

Dataset Description

We acquired the dataset from Kaggle, where it is hosted under the title "Amazon Fine Food Reviews."

- **Helpfulness Numerator**: 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.
- **Helpfulness Denominator**: 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.

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.

Methods

01 Collaborative Filtering

Data preprocessing for CF:

- We focus on a more representative subset by selecting only users and items with more than 10 interactions. To account for the importance of each rating, we use the helpfulness rate as a weight for the corresponding ratings. Ratings considered more helpful by users are given higher weight in the subsequent calculations,by using the below formula, we get the rate for training:

$$weight = \frac{raw\_data.HelpfulnessNumerator}{raw\_data.HelpfulnessDenominator + 10^{-8}}$$

$$Score\ for\ training = raw\_data.Score \times weight$$

- This process may generate ratings with a value of 0, which we remove to further reduce the sparsity of the dataset.

Methods

01 Collaborative Filtering

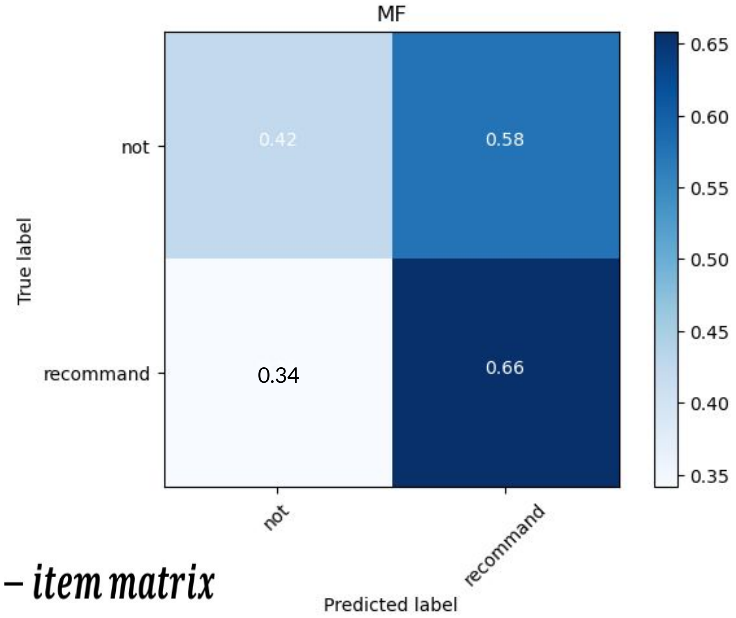
Training and elevation for the model:

- We use **RMSE** and **Confusion Matrix** for evaluating the accuracy of our model.

Hybrid CF

- Give weights to scores based on the helpfulness of the score

Training sample RMSE = 1.477
Testing sample RMSE = 1.503



$$Rating = user - user\ similarity \times item - item\ similarity \times user - item\ matrix$$

Methods

01 Collaborative Filtering

Summary of CF

- Collaborative Filtering does not provide satisfying result on this dataset due to data sparsity.

- Using the Hybrid Collaborative Filtering (CF) increases the performance of the single item based or user based CF, but only in scope of active user/items.

- Despite these improvements, the collaborative filtering method still has limitations and may not provide optimal results. So we further do the SVD++ and NCF for improving the performance of the recommendation system.

Methods

02 Matrix Factorization-based algorithms: SVD++

Goal:

To exploit all available interactions between user and item, we apply SVD++ to take into account implicit interactions, as well as user and item bias.

SVD++:

- An extension of the singular Value Decomposition (SVD) algorithm for collaborative filtering and advanced MF method that considers implicit feedback from users.

- 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.

Methods

02 SVD++

Rating Prediction formula for SVDpp:

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

µ: user
i: item
b_u: user bias
b_i: item bias
p_u: user-factors vector
I_u: item-factors vector

Interpretation

The term y_i are a new set of item factors that capture implicit ratings, which implies that a user u rated an item j. p_u and I_u are randomly initialized based on a normal distribution.

Include both user bias and item bias to calculate the predicted rating in the formula.

Methods

02 SVD++

Data Preprocessing with SVDpp:

- We only focus on the users, items and numeric ratings.
- Used to train-test-split to sample the subsetted data set into 75% training data and 25% test data.

Building the SVDpp model:

- Build a parameter grid to find the best combination of parameters (# factors, # epochs, learning rate, regularization term, etc)
- Apply GridSearch function, internally performing cross-validation during the hyperparameter search process
- Uses K-folds to train and choose the best model.

Parameters

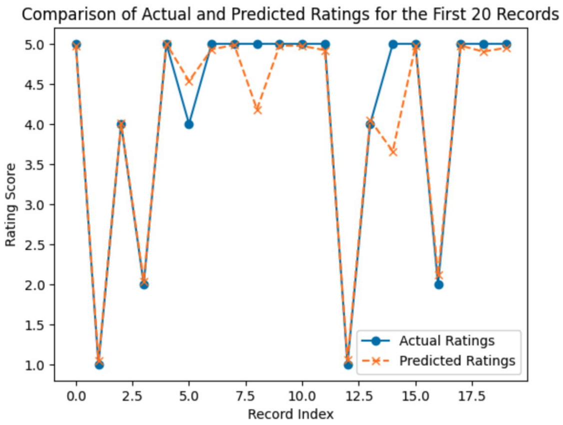
n_factors: 50, n_epochs: 35, Learning rate: 0.093, Regularization term: 0.051

Methods

02 SVD++

Results

- Overall test RMSE: 1.0168
- Plot of the test results:



Samples

```
User ID: A3SGXH7AUHU8GW,
Item ID: B001E4KFG0,
Predicted Score: 4.98,
Actual Rating: 5

User ID: A1D87F6ZCVE5NK,
Item ID: B00813GRG4,
Predicted Score: 1.04,
Actual Rating: 1

User ID: ABXLMWJIXXAIN,
Item ID: B000LQOCH0,
Predicted Score: 4.01,
Actual Rating: 4
```

Limitations

One way to mitigate the cold start problems of SVD++ is to incorporate additional features (demographic information, etc) into the model, but this requires additional and hybrid methods and collecting more information.

Methods

02 SVD++

Summary of SVDpp:

- SVD++ handles sparsity better 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 advantages of both near neighborhood & latent factor approaches.

Methods

03 Neural Collaborative filtering (NCF)

Matrix Factorization (MF):

- Usage of simple and fixed inner product to estimate complex user-item interactions

NCF:

- Address the limitations caused by using simple linear products by introducing a more expressive and flexible model

General NCF's predictive model:

$$\hat{y}_{ui} = f(\mathbf{P}^T \mathbf{v}_u^U, \mathbf{Q}^T \mathbf{v}_i^I | \mathbf{P}, \mathbf{Q}, \Theta_f)$$

where P and Q are the latent factor matrix for users and items respectively. f is the multi-layer neural network, and Θf denotes the parameters.

Two instantiations of the NCF framework:

- Generalized Matrix Factorization (GMF): Linear model
- MultiLayer Perceptron (MLP): Non-linear model

Methods

03 Neural Collaborative filtering (NCF)

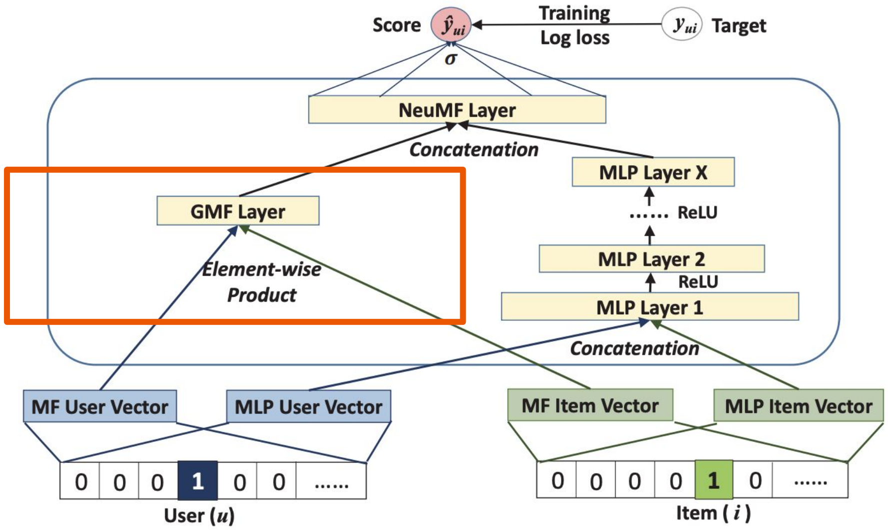
Pre-processing

- Include only relevant columns (UserID, ProductID, Score)
- Split into training and test sets and normalized tc between 0 and 1.

Generalized Matrix Factorization (GMF)

Steps

1. Forward pass: User and item indices are used as inputs to obtain their corresponding latent factors from the embedding layers
2. Element-wise multiplication: Once latent factors are obtained, GMF output is computed by element-wise product of these factors.



Methods

03 Neural Collaborative filtering (NCF)

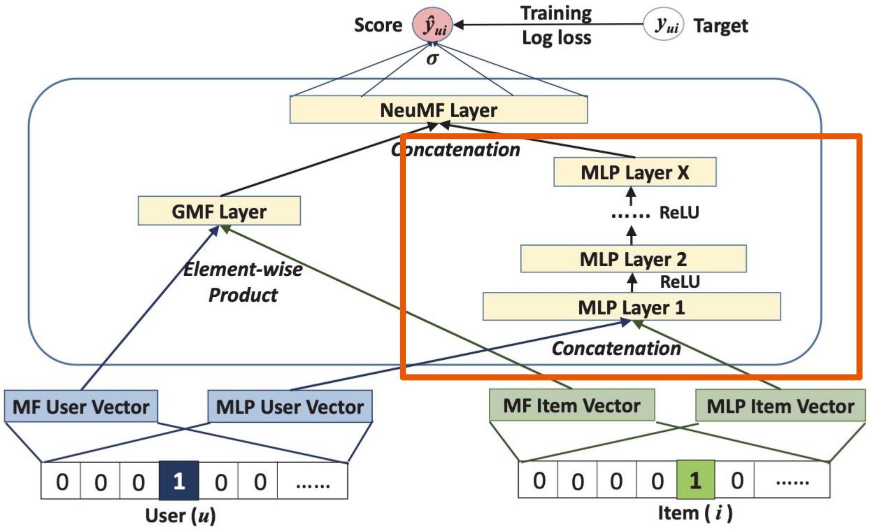
Multi-Layer Perceptron (MLP)

Steps

1. <u>Concatenation</u>: Combine user and item embeddings using concatenation
2. <u>Hidden Layers</u>: uses ReLU as activation; layers size: 32 -> 16 -> 8

Effects

- Learn the interaction between user and item latent features.
- Model large level of flexibility and non-linearity, learn interactions between user and items



Methods

03 Neural Collaborative filtering (NCF)

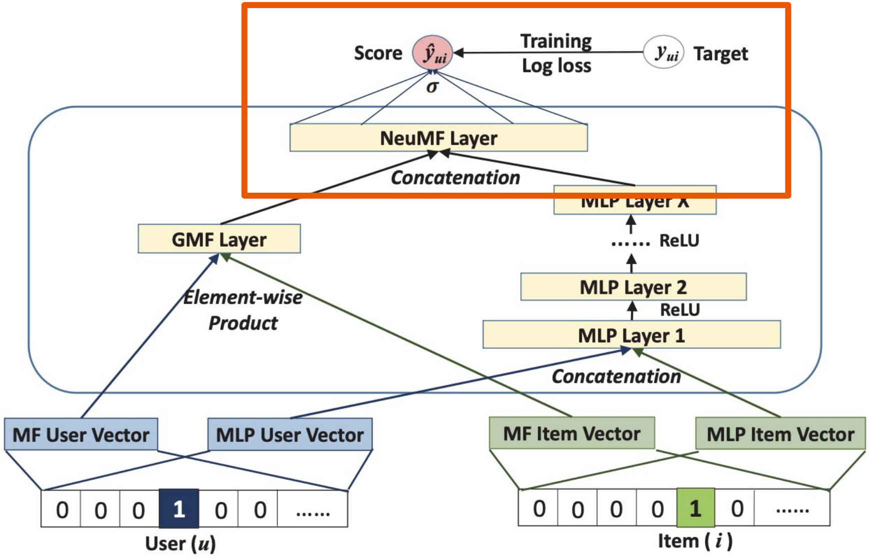
Fusion of GMF and MLP (NeuMF)

Steps

1. Concatenation: instead of sharing embeddings between GMF and MLP, the outputs are combined by concatenation
2. Generates prediction by sigmoid activation

Effects

- Combines the linearity of MF and non-linearity of MLP for modelling user–item latent structures

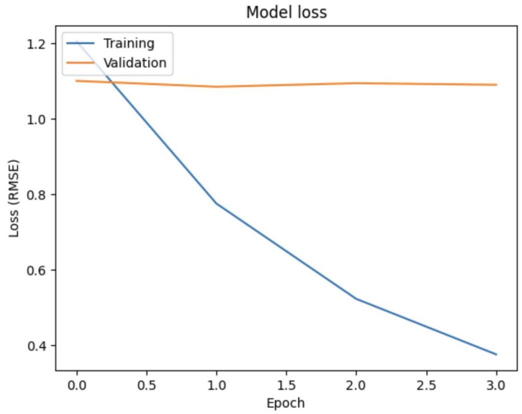


Methods

03 Neural Collaborative filtering (NCF)

Results

- Early stop at epoch = 4 to reduce overfitting
- Overall test RMSE: 1.066



Samples

```
User A1L01D2BD3RKVO, Item
B000EVG8J2, True Rating: 5,
Predicted Rating: 4.9

User A3U62RE5XZDP0G, Item
B0000BXJIS, True Rating: 5,
Predicted Rating: 5.0

User AOXC0JQQZGGB6, Item
B008FHUFAU, True Rating: 3,
Predicted Rating: 3.2

User A3PWPNZVMNX3PA, Item
B006BXV14E, True Rating: 2,
Predicted Rating: 2.8

User A1XNZ7PCE45KK7, Item
B007I7Z3Z0, True Rating: 5,
Predicted Rating: 5.0
```

Limitations

- We observe possible overfitting in this complex model as there’s only little improvement of validation loss over iterations
- The model’s complexity leads to low explainability
- Neural network takes longer to train for large data set

Conclusion

CF

RMSE: 1.503

Pros:

- Simple and fast
- interpretability

Cons:

- Cold-start problem
- Data sparsity
- Popularity bias: Tend to recommend the popular items

SVD ++

RMSE: 1.017

Pros:

- Accounts for both global and localized (user-item interaction) information
- Faster training time than neural models

Cons:

- Scalability concerns: leads to longer training times and takes more memory usage

NCF

RMSE: 1.066

Pros:

- Capture non-linearity and user-item interactions
- High flexibility

Cons:

- Complex model leads to overfitting
- Takes longer time to learn

Experiment & Future Work

SVD ++

RMSE: 1.017

Try a range of different values for parameters in the parameter grid to achieve RMSE of 1.017

- n_factors: 50
- n_epochs: 35
- Learning rate: 0.093
- Regularization term: 0.051

Future Direction

Sentiment analysis

- Filter out items with predominantly negative sentiment and bad ratings

- Ensure that the recommendation system suggests quality items only

Thank you.