# Recipe Recommendation

Chenxin Xiong (168449)
CS340 report
KAUST

December 3, 2019

## 1 Introduction

Cooking recipe websites have provided a very convenient, useful and necessary platform people who love cooking to search, share recipes they are interested in for a long time. Take Food.com for example, people can easily find various recipes of different cuisines on it and can give ratings, reviews for the recipes they like or dislike as well, which means there are lots of interactions between different users and recipes.
As we all know, recommender systems have been widely adopted by many online services, including E-commerce, online news and social media sites by modelling users' preference on items based on their past interactions [3]. Therefore, a recipe recommender system could be built based on abundant, available user-recipe interactions from Food.com.

## 2 Dataset

The dataset used to implement recipe recommender system in this project was released on Kaggle in 2019 [5]. The whole complete dataset has already been split into training, validation and test dataset. The number of interactions for each dataset is shown in the table below.

| Dataset | Interactions |
|---|---|
| Training | 700k |
| Validation | 7k |
| Test | 12.5k |

What does the content of dataset look like is shown in Figure 2. There are six fields included, but only three of them will be used in this project which are u (preprocessed ID of users), i (preprocessed ID of recipes) and rating.

| | user_id | recipe_id | date | rating | u | i |
|---|---|---|---|---|---|---|
| 0 | 2046 | 4684 | 2000-02-25 | 5.0 | 22095 | 44367 |
| 1 | 2046 | 517 | 2000-02-25 | 5.0 | 22095 | 87844 |
| 2 | 1773 | 7435 | 2000-03-13 | 5.0 | 24732 | 138181 |
| 3 | 1773 | 278 | 2000-03-13 | 4.0 | 24732 | 93054 |
| 4 | 2046 | 3431 | 2000-04-07 | 5.0 | 22095 | 101723 |

Figure 1: Training data examples

In training dataset, there are 25,075 different users and 178,265 unique recipes. The sparsity of the interaction matrix is 99.98%, which is highly sparse.

| Interaction | User | Recipe | Sparsity |
|---|---|---|---|
| 698,901 | 25,076 | 178,265 | 99.98% |

Figure 2: Information of training dataset

# 3 Experiments

In order to implement recipe recommender on the basis of the dataset. Rating prediction is tried first. Thus, the task is to predict the rating one user will give for a specific recipe given as accurate as possible, under this implementation, we can recommend the recipes with high ratings predicted to each user.

## 3.1 Rating prediction

Rating prediction is a regression task, two models are built and trained. One is the framework named Neural collaborative filtering (NCF) (as Figure 3 shows) proposed in this paper [3], and the other is Generalized Matrix Factorization (GMF), which could be interpreted as a special case of the NCF framework (as Figure 6 shows).

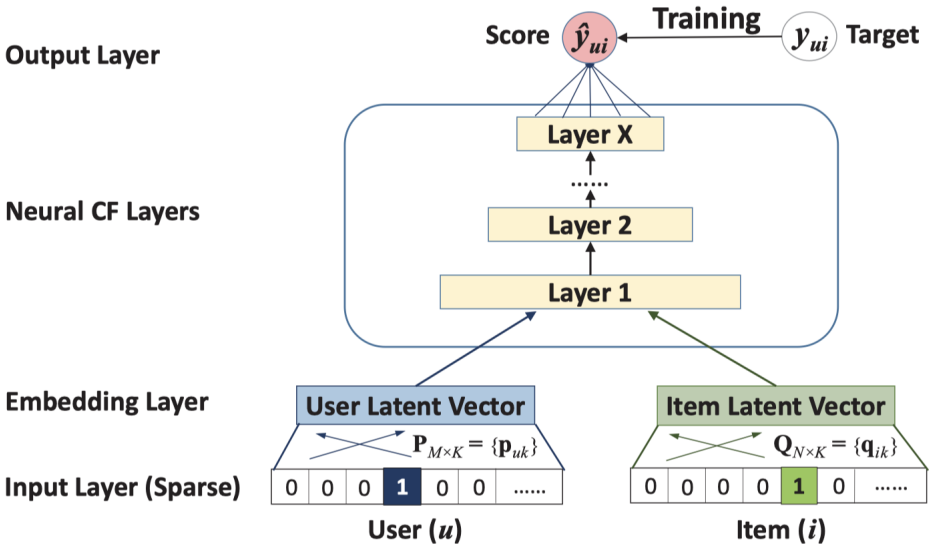


Figure 3: Neural collaborative filtering framework

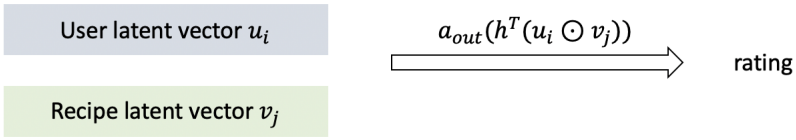


Figure 4: Generalized Matrix Factorization

GMF model has two embedding layers which are used to getting the latent vectors of users and recipes respectively. One fully connected layer with $ReLU()$ activation function and $sigmoid()$ as the ouput activation could make this model more expressive than the regular linear MF model which only produces the output by calculating the element-wise product of vectors. The original ratings whose values belong to the set [0, 1, 2, 3, 4, 5] are scaled in the range of [0, 1] by min-max scale operation.
As mentioned before, there are totally 25,075 different users and 178,265 unique recipes, and the embedding dimensions of users and recipes are the same, 100. Therefore, the size of embedding layer of users is $25075 \times 100$ and that of recipes is $178265 \times 100$. The implementation code of GMF model is shown in Figure 6. Besides, Figure 5 shows the training and validation loss.

```
RecipeGMFModel(
  (users_embedding): Embedding(25076, 100)
  (recipe_embedding): Embedding(178263, 100)
  (fc): Linear(in_features=100, out_features=1, bias=True)
  (sigmoid): Sigmoid()
)
```

Figure 5: Implementation of GMF

| Epoch | Training loss | Validation loss | time |
|---|---|---|---|
| 8 | 0.0364 | **0.073** | 11.57 min |

Figure 6: Training and validation loss of GMF

About the implementation of NCF model, with the consideration of the extremely high sparsity of the dataset working on, three hidden layers are stacked to ensure that the model have enough capability to predict ratings as accurate as possible.

```
RecipeNCFModel(
  (users_embedding): Embedding(25076, 100)
  (recipe_embedding): Embedding(178263, 100)
  (drop): Dropout(p=0.2, inplace=False)
  (hidden): Sequential(
    (0): Linear(in_features=200, out_features=300, bias=True)
    (1): ReLU()
    (2): Dropout(p=0.5, inplace=False)
    (3): Linear(in_features=300, out_features=200, bias=True)
    (4): ReLU()
    (5): Dropout(p=0.5, inplace=False)
    (6): Linear(in_features=200, out_features=100, bias=True)
    (7): ReLU()
    (8): Dropout(p=0.5, inplace=False)
  )
  (fc): Linear(in_features=100, out_features=1, bias=True)
  (sigmoid): Sigmoid()
)
```

Figure 7: Implementation of NCF

Figure 8 shows the training and validation loss as well. By comparing the validation losses of GMF and NCF, it can be noticed that the performance of NCF is slightly better that that of GMF, with using five times more training time. Moreover, the overfitting problem happens to both GMF and NCF due to the gap between training and validation loss. Actually, it's not surprising to see this kind of result cause the high sparsity of the interaction matrix, which is not capable to provide enough information for models to learn.

| Epoch | Training loss | Validation loss | time |
|---|---|---|---|
| 28 | 0.0303 | **0.068** | 54.55 min |

Figure 8: Implementation of NCF

Based on the analysis above, it's not appropriate to directly use the original explicit feedback data. Therefore, working on implicit feedback data will be more reasonable so that we can not only utilize the information provided by the interactions observed, but also the information beyond the dataset to make the model have better performance.
The implicit feedback data is transformed through Equation 1 [7].

$$Y_{ij} = \begin{cases} 0, & if\ R_{ij} = unk \\ 1, & otherwise \end{cases} \tag{1}$$

Thus, the goal becomes to do Top-K recommendation instead of rating prediction now.

## 3.2 Top-K recommendation

Considering the aim has been changed, it's not proper to work with the original dataset. Thus, the whole complete dataset is first processed by only keep the users who have more than 20 interactions because over 30% users have less than five interactions, which makes it not easy to evaluate collaborative ltering algorithms, and then is split into training dataset and validation dataset.
Leave-one-out evaluation [6] is used to form the validation dataset by holding one of the interactions for each

unique user. During evaluation process, for each interaction in the validation dataset, the recipe will be ranked among 100 recipes that are not interacted by the user [1], and the performance of a ranked list is judged by Hit Ratio (HR) [2] of top 10.
It could be noticed that the number of interactions, users and sparsity all decreases in Figure 9.

| Interaction | User | Recipe | Sparsity |
|---|---|---|---|
| 580,015 | 5,312 | 178,265 | 99.94% |

Figure 9: Information of processed training dataset

Before train the model, negative sampling ratio needs to be determined. In NMF paper, 5 is used as the ratio [3]. With the assumption that the negative samples randomly sampled are all different, for movie lens and pinterest these two datasets, 26.8% and 1.65% information of the whole interaction matrixes will be covered respectively during one training epoch. If the dataset used in this project wants to reach the same cover degree, the ratio should be 438 and 26 respectively.
After built the model by fusing GMF and MLP (as Figure 12 shows), experiments have been done using different negative sampling ratios.

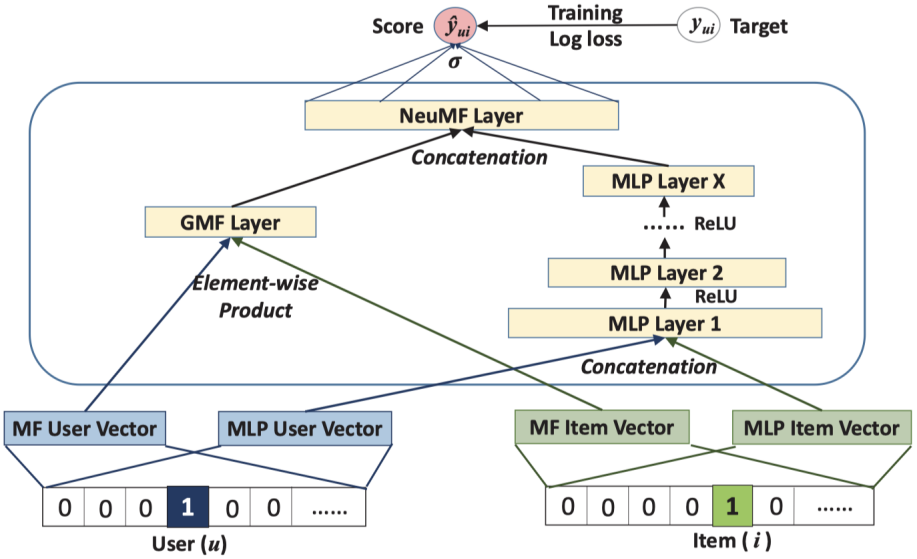


Figure 10: Neural matrix factorization model

It turns out that the HR@10 increases when ratio is enlarged.

| Negative sample ratio | Epochs | Hit Ratio |
|---|---|---|
| 4 | 18 | 12.9% |
| 8 | 23 | 12.97% |
| 20 | 9 | 13.48% |

Figure 11: HR@10 with different negative sampling ratios of 4, 8 and 20

But the HR is still very small when the ratio is 20, so integrating the features of recipes is tried for improving the model's performance. There are 178,265 distinct recipes, and the number of different ingredients of all the recipes is between the range of [4, 20].
Use skip-gram model to get the embedding of each ingredient and then get the embedding of each recipe by taking average of the embeddings of the ingredients it contains.

| Ingredient | Vector |
|---|---|
| great_northern_beans | [0.1419527, 0.050856147, 0.12818474, 0.2158141... |
| yellow_onion | [0.08089653, -0.08723446, -0.04058786, 0.33989... |
| diced_green_chilies | [-0.0064202445, 0.18017997, 0.13981035, -0.111... |
| ground_cumin | [0.038344134, -0.22157283, -0.054828685, 0.113... |
| garlic_powder | [0.081294596, -0.0376514, -0.010471731, 0.2145... |
| fat-free_chicken_broth | [0.12689918, 0.003707604, 0.10743472, 0.179322... |
| fresh_cilantro_leaves | [-0.081476256, 0.099624015, -0.17738055, 0.052... |
| extra_virgin_olive_oil | [-0.008849304, 0.014419843, -0.013997273, 0.39... |
| sour_cream | [0.019840665, -0.04861686, 0.20020878, -0.0403... |
| devil's_food_cake_mix | [-0.3262564, -0.31572446, -0.13684636, -0.1846... |

| i | ingredients | vector |
|---|---|---|
| 0 | ['great northern beans', 'yellow onion', 'dice... | [0.04360911116666666, -0.005143107888888889, 0... |
| 1 | ["devil's food cake mix", 'vegetable oil', 'eg... | [-0.17808505125000001, -0.18320775649999999, 0... |
| 2 | ['mayonnaise', 'salsa', 'cheddar cheese', 'ref... | [-0.045958727461538465, 0.15744469853846155, 0... |
| 3 | ['chicken tenders', 'flour', 'garlic powder', ... | [0.010844033750000003, -0.11082101774999999, -... |
| 4 | ['lamb shoulder', 'salt', 'ground black pepper... | [0.026242995312500003, -0.16281941465, 0.00755... |
| 5 | ['pork chops', 'apple', 'dried apricot', 'cran... | [-0.027189695750000003, -0.16694935141666667, ... |
| 6 | ['nectarines', 'ear of corn', 'shallot', 'haba... | [-0.049376330499999996, -0.0383035832125, -0.0... |
| 7 | ['butter', 'onion', 'garlic', 'potatoes', 'flo... | [-0.018259710176923072, -0.09238103823076924, ... |
| 8 | ['strawberries', 'brown sugar', 'cool whip fre... | [-0.2049909151, -0.2363453425, -0.039119478699... |
| 9 | ['oats', 'whole wheat flour', 'fiber one cerea... | [-0.181577497, -0.24964736769230766, -0.081040... |

Figure 12: Ingredient embeddings (left) and recipe embeddings (right)

After obtain the features of recipes, the original latent vector of NCF model is replaced by the recipe embedding. The changing trends of loss and HR@10 during each training epoch are shown in Figure 17.

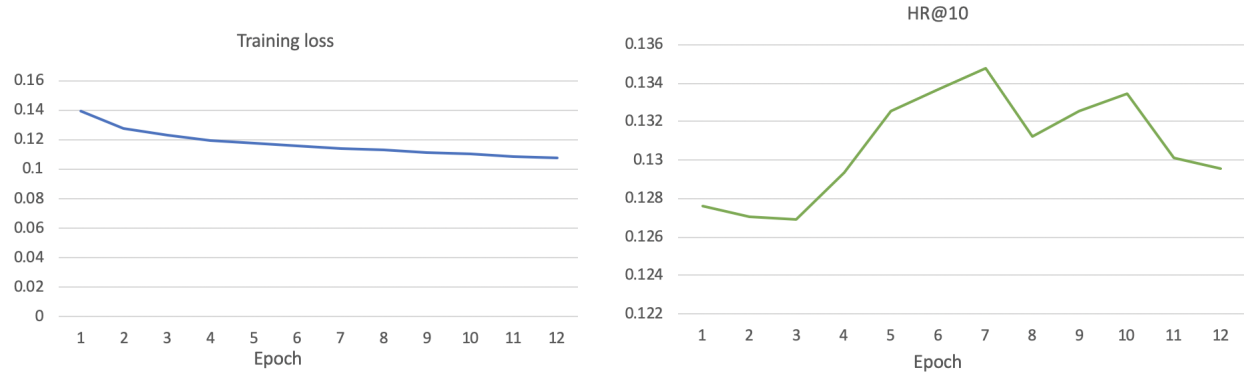


Figure 13: Training loss (left) and HR@10 (right)

However, HR@10 improves a little but is still bad, even the negative sampling ratio is large (30).

| Negative sample ratio | Epochs | Hit Ratio |
|---|---|---|
| 30 | 7 | 13.5% |

Figure 14: HR@10 with integrating recipe features

Under this situation, directly using the ingredient embeddings may help cause by taking average to get recipe embeddings might result in the lost of information. Because each recipe could be regarded as a sentence so it is able to be represented by a vector after convolutional layer and max pool operation, which is follwing this paper [4]. Then, a fully connected layer is used to project this vector to the latent space with dimension of 16. Besides, the dimension of user embeddings is 16 as well.
In order to speed up the training process, 8 is chosen as the negative sampling ratio and the values determined of other hyperparameters are shown below.

| | |
|---|---|
| user latent vector dimension | 16 |
| recipe latent vector dimension | 16 |
| negative sampling ratio | 8 |
| batch size | 512 |
| learning rate | 1e-4 |

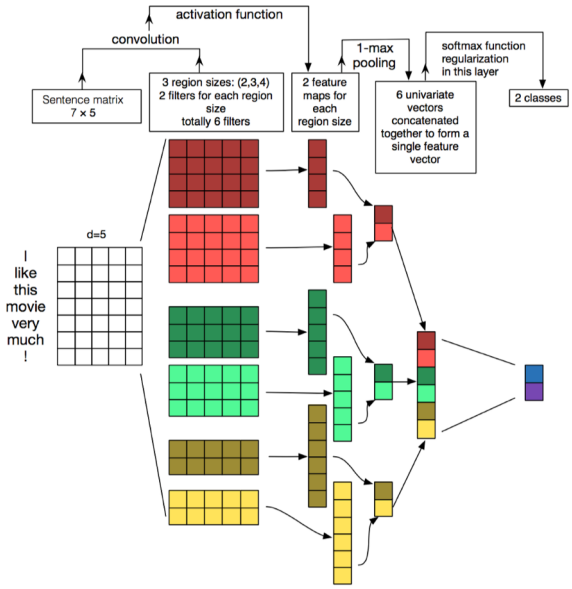


Figure 15: CNN architecture for text classification

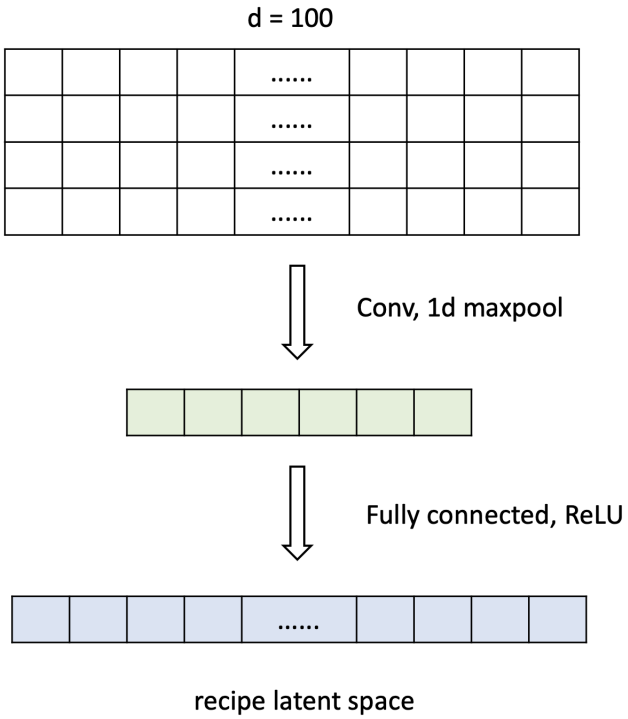


Figure 16: CNN for utilizing recipe features

# 4 Results and Analysis

As we can see, the performance of the model is not as good as expected and somehow is difficult to be improved. Some possible reasons could be come up with to analyse the bad result.

First, it must be an issue that the high sparsity of the interaction matrix, although the users who have less or equal to 20 interactions have been removed from the dataset, which may arise another obvious issue that the number of recipes is way larger than that of users.

Second, NCF framework probably is not very suitable for the dataset to get good performance. Other models should be tried.

Except the reasons analyzed above, the Embeddings of recipes which are interacted by one user who has 375

interactions and another user who has 1512 interactions are visualized by PCA before and after training. It could be obviously observed that after training, the recipes interacted by the same user are more close to each other.

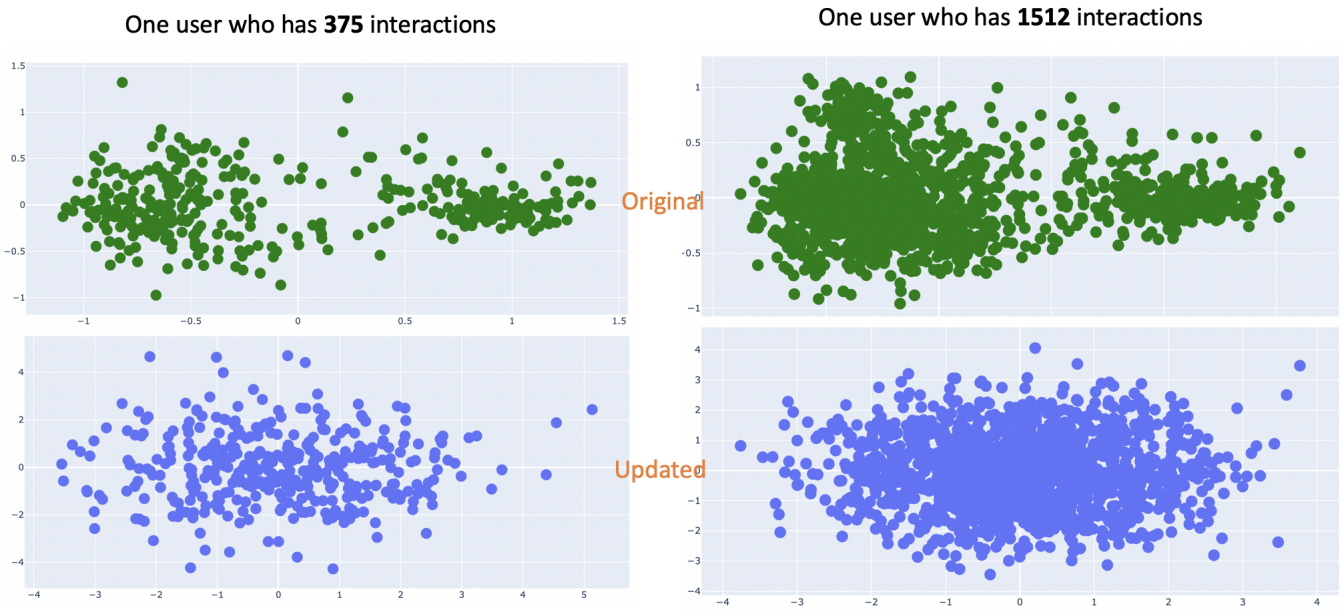


Figure 17: Visualize the recipe embeddings before and after training by PCA

# References


[1] A. M. Elkahky, Y. Song, and X. He. A multi-view deep learning approach for cross domain user modeling in recommendation systems. In *Proceedings of the 24th International Conference on World Wide Web*, pages 278–288. International World Wide Web Conferences Steering Committee, 2015.

[2] X. He, T. Chen, M.-Y. Kan, and X. Chen. Trirank: Review-aware explainable recommendation by modeling aspects. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 1661–1670. ACM, 2015.

[3] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*, pages 173–182. International World Wide Web Conferences Steering Committee, 2017.

[4] Y. Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

[5] B. P. Majumder, S. Li, J. Ni, and J. McAuley. Generating personalized recipes from historical user preferences. In *EMNLP*, 2019.

[6] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*, pages 452–461. AUAI Press, 2009.

[7] H.-J. Xue, X. Dai, J. Zhang, S. Huang, and J. Chen. Deep matrix factorization models for recommender systems. In *IJCAI*, pages 3203–3209, 2017.