Completing Partial Recipes using Collaborative Filtering with PCA and TF-IDF Modified Rating Matrix

Zheng Huang
s1933134
z.huang-41@sms.ed.ac.uk

Yannan Huang
s1875614
y.huang-89@sms.ed.ac.uk

Tianze Wei
s1932446
t.wei-8@sms.ed.ac.uk

Luoye Wang
s1894929
l.wang-99@sms.ed.ac.uk

Abstract

Many recommender systems use collaborative filtering with interesting similarity measures, to make recommendations to users. These methods can also apply to help with complete partial recipes, however sometimes they simply take similarity between the different ingredients and recipes, but do not take more information into account. Therefore, in this project we try to improve the algorithms by taking more information, for example, the country of a cuisine, into consideration when making recommendation. We firstly implement the basic item-based collaborative filtering method with different similarity measures and PCA to develop a recommender system to suggest additional ingredients for a partial recipe, and based on these methods, we further combine the TF-IDF measures to calculate a new rating matrix so the effect of "country" will be considered, and find that with PCA and TF-IDF modified matrix, we have the best result which achieves a recall@10 of 47%.

1 Introduction

1.1 Description of Task

This project is initially designed to learn the fundational principle of recommender with collaborative filtering by exploratory experiments with technical adjustments on the real-life dataset of recipe[9]. Furthermore, to our knowledge, try to improve parts of algorithm in existing structure of item-based collaborative filtering to suggest more accurate ingredients to the partial recipe.

1.2 Background and Previous Work

According to the task description, many destinations can be set for this recipe dataset, such as clustering, cuisine prediction and collaborative filtering. Our final choose is collaborative filtering, that is given a partial recipe, to suggest other ingredients that could go along with it, which is fairly new but attractive to us.

The objective of this task as such, to provide the original ingredient to the partial recipe as possible, is in line with the essence of Recommender System (RS), that is suggesting items to be of use to a user [7]. Content-based recommenders ,such as decision trees and neural networks, match the history of the users preference tags with item description to predicte future interest of the user.[6] Latent factor models(LFMs) , such as Singular Value Decomposition (SVD), transform both items and users to a latent factor space, and then make prediction according to the new rating matrix[4].

Collaborative Filterings (CFs), as the main technique of RS except to Content-based recommenders and LFMs, can be divided into below categories in different views: memory-based or model-based models, user-based or item-based models.[8, 1] Memory-based models predict item ratings based the users-items matrix, while model-based models develop machine learning models or rules first to get user ratings and then predict the user's rating for a new item by a probabilistic approach. User-based models calculate the similarity between user x and all other users, then recommend items favoured by the set of similar users while item-based models recommend the items based on the user purchase history.

De Clercq et al. implements several model-based CFs with alternative modules, such as matrix factorization and two-step regularized least squares, to complete partial recipes. Similarly, Cueto, Roet, and Słowik build quality item-based CF algorithm to suggest ingredients to partial recipes based the "What's Cooking" dataset[9] after dimensionality reduction with Principal Components Analysis (PCA).

1.3 Relevance of our Task

Inspired by above papers and the knowledge we learned from courses, we first explore the work-frame of item-based CFs with different calculation of similarity and the vanilla PCA used by Cueto, Roet, and Słowik compared to kernel PCA, which provides comparison on a real-life dataset containing the implicit feedback and suffered the property of sparsity (consisting of binary data and mostly are zero). The result of experiment could be a reference for RS developers.

Then, we propose a new representing form for rating vectors derived from word embedding technique from natural language processing(NLP) which can incorporate external features. It has a significant improvement on accurate ingredient complement proved by our experiment in section 5.

2 Data Preparation

Our data is originally from Kaggle, "What's Cooking"[9]. The basic preprocessing steps are based on [2], with some differences to better serve our models. The raw training data is stored in JSON format and it contains three labels: id, cuisine and ingredients. After imported the training data set, we have 39774 recipes with 6703 unique ingredients and these recipes belong to 20 cuisines.

We prepared our data set in four steps. Step one is to vectorize the rating matrix R. In this step, we used the $CoutVectorizer$ from $sklearn.freture\_extraction.text$ package, as well as the $pd.DataFrame$ from $pandas$ library, so we had a 39774×6703 matrix R, where the rows are 39774 recipes and the columns are 6703 ingredients, and $r(i, u)$ is 0 if recipe u has no ingredient i, else $r(i, u)$ is 1.

From step two, we started to clean the data so some useless recipes and ingredients would be removed. We removed 3029 ingredients with 3 or less occurrence in all recipes, and also we removed quantity qualifiers by removing the special characters and numbers (e.g., "(10 oz.)", "1%") so similar ingridients are combined. In step three, we combined the ingredients with different versions by using their longest common substring, for example, "bbq sauce" and "hot sauce" would be combined as "sauce" only. We also simplified the ingredients' name by only using the substrings that appear in more than 30 different ingredients or in more than 1000 recipes, so we could keep the dominant common ingredients and remove the infrequent ingredients.

In step four, we continued to reduce the dimension of the data by removing the infrequent ingredients and oversimple recipes. To achieve this, we removed ingredients show up in 250 recipes or less, and the recipes with two or less ingredients. Different to [2], we did not remove the very frequent ingredients such as "salt" and "water", which are considered as "stop words" in [2], since we believed it will be reasonable if they are recommended as they can be a "must" ingredient in some recipes. However, it is reasonable to consider the influence of commonness of ingredients, so we punished these very common ingredients but gave more weights for those by introducing TF-IDF measure into our rating matrix, which will be shown in section 4.4. Moreover, we went through the processed data manually and decide whether to remove or combine anything still strange, for example, "chile", "chili" and "cayenne" are combined as "chili" since they are very similar. Finally, we had our ready-to-use dataset and it has 161 ingredients and 38472 recipes.

3 Exploratory Data Analysis

Before implementing our algorithm about collaborative filtering, we firstly did exploratory data analysis to show our data. The whole dataset has 38472 recipes and 161 ingredients after doing data preprocessing step. The data is high-dimensional and we can use some visualization and statistical description to exact some features from the dataset.

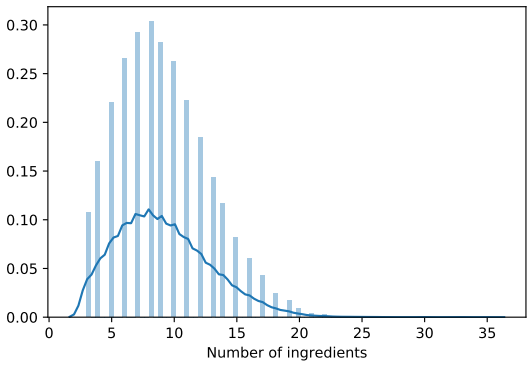

Figure 1: Recipe length distribution

The Figure 1 shows the recipe length distribution, which represents how many ingredients a recipe has. The maximum length of recipe is 35 and the minimum is 3. Besides that, after calculation we found that both mean and median of recipe length is 9 ingredients. We also can see from the figure that the mode of the recipe's length lies between 7 and 10. The whole dataset has 161 columns and these analysis shows that our data matrix is a sparse one. Also, the recipe length distribution has positive skewness of 0.65 and kurtosis of 0.47, which shows that the recipe in our dataset tends to have small number of ingredients, though still some outliers have many ingredients.

The Figure 2 shows the top 50 ingredients in our processed dataset. We can see that the top 10 most popular ingredients that can be found in more than 7500 recipes are *salt*,*onion*, *garlic*, *olive*, *oil*, *chill*, *water*, *tomato*, *black pepper*, and *butter*. This result matches our real life experience since our daily food always also use these ingredients very frequently, especially *salt*, *onion* and *garlic*, they are very common ingredients in many cuisines and all showing up in more than 15000 recipes, which may be highly recommended by our models. However, some ingredients appear less times and in specific recipes and cuisines. This information can help us modify our model which penalizes the most common ingredients and gives the special ingredients more chance to be recommended when needed.

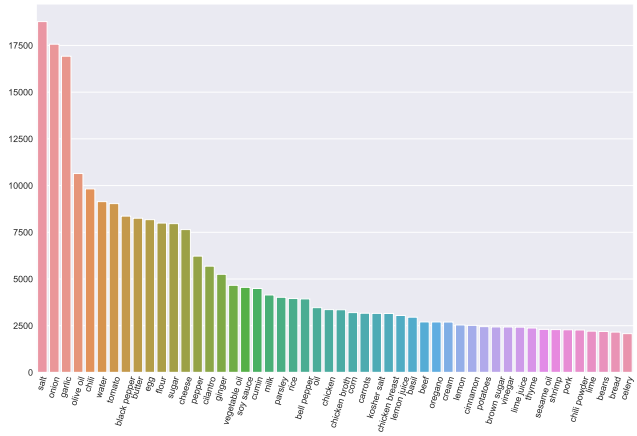

Figure 2: The frequency of ingredients

Moreover, in order to visualize these ingredients, we used Principal Component Analysis (PCA) on the recipes to reduce the whole dimension to two dimensions and this result can have a better

visualization of ingredients. The Figure 3 shows the top 50 ingredients since it may have a better interpret ability. From this figure, we can see that some common ingredients like *bread*, *sugar*, *lemon* and *cinnamon* are in a small cluster on the middle of left side. This result demonstrate that these ingredients may have a link with each other. However, the most popular ingredients mentioned before like: *oinon*, *oil* and *garlic* are separated and in different position. However, when we used PCA on the ingredients to visualize the recipe, the result is not satisfying since it can not separate some different recipes.

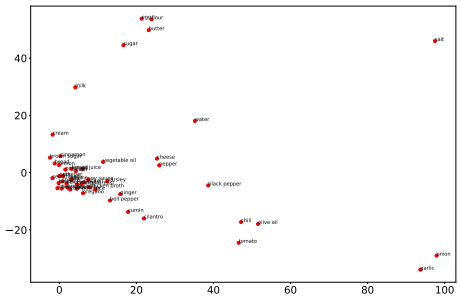

Figure 3: The result of PCA

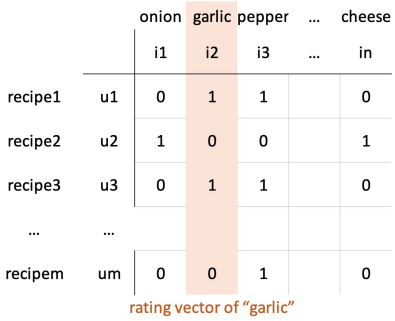

Figure 4: User-item matrix

4 Methodology

In this section, all the formula will be discussed based on our objective of task, that is to suggest ingredients to the partial recipe.

4.1 Workframe of Item-based CF

Generally, the workframe of item-based CF [2] can be summarized to two steps: $Step$ 1, to calculate the similarity between any two ingredients. The specific formula of similarity will be given in the next subsection. $Step$ 2, to calculate the extent of fit between an ingredient i and a recipe u and predict a descending list of ingredient recommendations to a recipe. The extent of fit between an ingredient i and a recipe u is denoted by $P(u, i)$.

$$P(u,i) = \frac{\sum_{j \in N_i} s_{ij} \cdot r_{uj}}{\sum_{j \in N_i} |s_{ij}|} \tag{1}$$

N_i : the set containing the k neighbours most similar to ingredient i
k: for each ingredient, find k most similar ingredients (i.e. k neighbours)
s_{ij} : the similarity between ingredient i and j

The overall idea of the construction $P(u, i)$ is representing the fit by a division where the numerator is the similarities of the ingredients in a recipe between the target ingredients and the denominator is the sum of all similarities of the neighbourhood in target ingredient i (r_{uj} is use to judge whether ingredient j exists in recipe u, 1 if it exists, 0 if it does not exist). If P is 0, it means that there is none of the ingredient's k closest neighbours appear in the recipe u. If P is 1, it means that all of its neighbours appear of the target ingredient, that is, recipe u is very suitable for ingredient i.

4.2 Similarity Measures

When we evaluate the similarity between different ingredients, we use four different methods. They are cosine similarity (CS, in (2)), asymmetric cosine similarity (ACS, in (3)), Jaccard similarity (JS, in (4)), point-wise mutual information (PMI, in (5)). In first three expressions ((2), (3), (4), r_i means the column vector of ingredients i which includes all recipes. In PMI, $p(i, j)$ means the joint probability of ingredient i and j which both appear in one recipe and $p(i)$ means the probability of ingredient i which is included in one recipe.

$$CS_{ij} = \frac{r_i \cdot r_j}{\|r_i\|^2 \|r_j\|^2} \tag{2}$$

$$ACS_{ij} = \frac{r_i \cdot r_j}{\|r_i\|^{\alpha} \|r_j\|^{1-\alpha}} \tag{3}$$

$$JS_{ij} = \frac{r_i \cdot r_j}{\|r_i\|^2 + \|r_j\|^2 - r_i \cdot r_j} \tag{4}$$

$$PMI_{ij} = log \frac{p(i,j)}{p(i)p(j)} \tag{5}$$

Compared with other three methods, asymmetric cosine similarity is the only one method which is asymmetric. Therefore, there will be more times of calculation in asymmetric cosine similarity. In other words, asymmetric cosine similarity is less efficient than other three methods. Also, this symmetric property also has some problems. For example, if we firstly get the uncommon ingredients and its similarity, we can predict some common ingredients. But if we get the common ingredients and its similarity firstly, we may not easily predict the rare ingredient.

When using asymmetric cosine similarity, we need to decide the parameter α. At last we follow the configuration of [2] and set $\alpha = 0.05$. JS also has one unique drawback since it may not use the information of exact number when the magnitude of rating is useful. PMI has one unique advantage since the meaning of PMI is intuitive. PMI evaluates the frequency of these two ingredients used together relative to their own frequency and it is easily to be understood by people.

In the following experiments, there will be some comparisons among their own different performances like: recall rate, median of rank, mean of rank in evaluation.

4.3 Principal Component Analysis (PCA)

Principal components analysis (PCA) is technique to analyze and simplify data sets. PCA is often used to reduce the dimensionality of the data set, while maintaining the characteristics of the largest variance contribution in the data set.

Our dataset contains 30,000+ recipes, which is a high dimensional, computation-consumed for the item-based CF, and could contain repetitive information. It is considerable of PCA to transform the highly sparse representation to a dense, low-dimensional model of latent factors.

PCA and kernel PCA (sigmoid rbf) in scikit-learn library is used here to be compared and obtain 161 principal components in the transformed recipe space as many as there are ingredients. The number is close to the number of ingredients, which takes up considerable variance and would contribute to **scalability**[8].

4.4 Term Frequency Inverse Document Frequency (TF-IDF)

Beside the binary data of recipe and ingredient, our dataset also contains a feature "cuisine" indicating the regional information of a ingredient. In [2] and our baseline, the rating of an ingredient is represented by the vector in the user-item matrix. This kind of vector representation reminds us of **word embedding** considering context in NLP, which uses vector to represent word with its context.

So TF-IDF (term frequency – inverse document frequency)[5], a method commonly used weighting technique for information retrieval and data mining, is implemented here to deal with the external cuisine factor - "country" as the formula shown in (6). Then the result is normalised among each columns with (7) and adjusted by a value of weight. Finally we concatenate it (rows of 20 countries and columns of 161 ingredients) to the original representation to calculate the new rating matrix.

$$r(i,c) = tf(i,c) \cdot \frac{N}{df(i)} \cdot \frac{P}{Q(c)} \tag{6}$$

$$z(i,c) = \frac{r(i,c) - min(i)}{max(i) - min(i)} \tag{7}$$

where i denotes the ingredient, c denotes the cuisine, $\frac{N}{df(i)}$ is the ratio of total number of cuisines and number of cuisine appearing in i, $tf(i,c)$ is the ingredient frequency belonging to c, and $\frac{P}{Q(c)}$ normalizes the cuisine-inbalanced data by the ratio of total number of recipes and number of recipes belonging to c.

4.5 Evaluations

Following Cueto, Roet, and Słowik and De Clercq et al., three quantitative measure metrics are used here to evaluate the performance of model with different parameters and modules.

- Recall@10: the percentage of cases where the missing ingredient(which is masked by us so we know what the real missing ingredient is) exists in the top 10 ingredient of the recommendation list.
- Mean rank: mean rank of the missing ingredients in the recommendation list.
- Median rank: median rank of the missing ingredients in the recommendation list.

5 Result and Discussion

This section presents the experiment results and corresponding discussions.

5.1 Exploration of Suitable k, Similarity Measures and PCA

The number of neighbours k is a core parameter, which has a great impact on the memory-based collaborative filtering system. A low k will render high biases towards a small part of the ingredients, while a high k will significantly increase the sensitivity to small changes. To explore the suitable value of k, we shuffle and split 10%(3847) of the dataset for test. For each experiment, we have the random seed to guarantee the same incompleteness of recipes. We then do a parameter sweep over $k = \{10, 20, ..., 70\}$, which sufficiently covers the optimal conditions, with some similarity measures(i.e. Cosine, Asymmetric Cosine, Jaccard, and PMI) and whether applying PCA or not. Figure 5 shows the recall and median rank varying with the number of neighbours under different combinations. Table 1 reports the results of PCA-reduced data with different similarity measures and a suitable k.

It can be observed that overall, PCA-reduced rating matrix shows great advantages in the metrics of recall and median rank. Jaccard similarity is an exception that with and without PCA can both achieve high performance. Among four similarity measures, Cosine appears to perform best due to high robustness, highest recall of 46.04%, lowest mean rank of 28.15 and lowest median rank of 12 in the PCA-reduced case. It can be also seen that, the number of neighbour $k = 40$ is a suitable choice for further experiments.

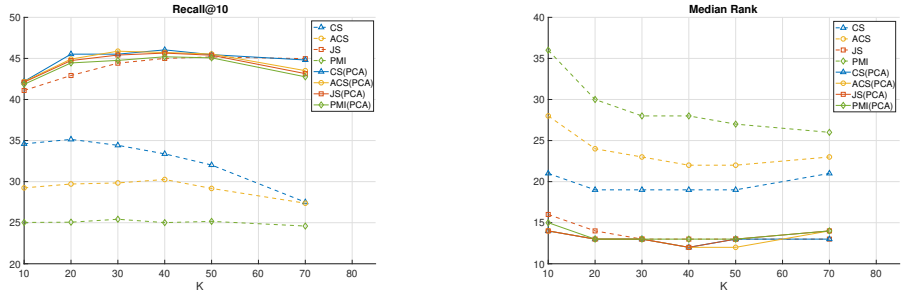

Figure 5: Recall and median rank varying with the number of neighbours k

5.2 Effects of Applying Different Kernel PCAs

From above section Figure 5, we can see that the algorithms with PCA-reduced data perform better then those with non-reduced data, which means PCA can help to improve the results. Since PCA

Similarity measure	Recall@10	Mean rank	Median rank
Cosine	**46.04%**	**28.15**	**12**
Asymmetric Cosine	45.72%	29.18	12
Jaccard	45.65%	29.40	12
PMI	45.20%	30.25	13

Table 1: Results of PCA-reduced rating matrix with different similarity measures and $k = 40$

can be implemented with different kernels, it will be straightforward and interesting to discover if different kernels can help with results improvement. Therefore, we introduced kernel PCA (KPCA) with radial basis function (rbf), sigmoid and cosine kernels. As we can see from Table 2, KPCA with rbf kernel improves the recall by 0.2% only, but KPCA with sigmoid and cosine kernels do not even beat the original PCA. From the results, we believe it is not very helpful to transfer our processed data into a non-linear space before dimensional reduction. Therefore, in the further experiments, we did not use KPCAs as our dimensional reduction methods, but keep using original PCA.

	Recall@10	Mean rank	Median rank
PCA	46.04%	28.15	12
KPCA + rbf	**46.24%**	28.29	12
KPCA + sigmoid	44.97%	27.87	13
KPCA + cosine	43.02%	24.84	14

Table 2: Results of applying kernel PCA different kernels on Cosine measure and $k = 40$

5.3 Modified Rating Matrix with Cuisine Factor

The significant drawback of the previous method is ignoring the cuisine information of recipes. We propose a more comprehensive similarity calculation scheme to make the best use of this relationship and further improve the performance of the CF system.

Through the statistical analysis of ingredients and cuisines, and weighting with TF-IDF technique introduced in Section 4.4, we can obtain a new cuisine rating matrix with respect to ingredients(denoted as cuisine matrix $r(i, c)$). We then normalize it to [0, 1] to ensure the balance between the cuisine matrix $r(i, c)$ and the original $r(i, u)$. The idea is to concatenate them together to form a larger matrix. In detail, the shape of the original rating matrix and the cuisine matrix is $(38k, 161)$ and $(20, 161)$ respectively, and we introduce a multiply factor m for repeating the $r(i, c)$ concatenated with the $r(i, u)$, hence the final shape of the modified rating matrix should be $(38k + 20m, 161)$. Multiply factor m reflects the importance of the additional cuisine information. We choose to repeat the concatenation operation for several times instead of directly multiplying the value because the latter is in bad scale when the similarity calculation involves high power arithmetic. After that, we utilize the modified matrix to evaluate the similarities between ingredients.

Concerning the PCA-reduced case, we investigate two methods of concatenating the cuisine matrix $r(i, c)$:

1. **before-PCA-reduced:**
   Concatenate $r(i, c)$ and the original matrix first, then do the PCA for the modified rating matrix.
2. **after-PCA-reduced:**
   Do the PCA to the original matrix first, then concatenate $r(i, c)$ to form the modified rating matrix.

Again we use Cosine similarity measure for the experiments and do a parameter sweep over the multiply factor $m = \{10, 20, ..., 160, 200\}$ to find the optimal performance. The recall and median rank plot is shown in Figure 6 and the optimal performance is illustrated in Table 3.

After concatenation, the performance of the modified original model is significantly boosted and reaches the peak at $m = 100$, which achieves 11.33% improvement on recall and lower mean and

median rank. However, it is still not competitive compared to PCA-reduced cases. It can be seen that the after-PCA-reduced consistently outperforms the before-PCA-reduced. And both concatenation methods can contribute to the performance with suitable m. Modified before-PCA-reduced slightly improves the recall by 0.41%, while modified after-PCA-reduced model obtains the highest recall of 47% so far. Another useful observation is that PCA-reduced cases require significantly lower multiply factor to achieve the optimal performance.

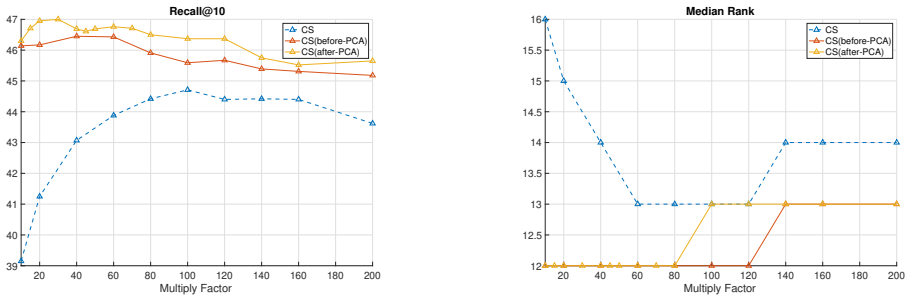

Figure 6: Recall and median rank varying with the multiply factor m

	Recall@10	Mean rank	Median rank	Multiply factor
Original	33.38%	28.73	19	-
Modified Original	**44.71%**	**25.66**	**13**	100
PCA-reduced	46.04%	28.15	12	-
Modified before-PCA-reduced	**46.45%**	27.20	12	40
Modified after-PCA-reduced	**47.00%**	27.34	12	30

Table 3: Results of modified rating matrix with cuisine factor on cosine measure

6 Conclusions

In this project, we implemented basic item-based CFs with four similarity methods and vanilla PCA as our baseline, and tried to improve the algorithms by using KPCA with three different kernels and modified rating matrix which uses TF-IDF to take cuisine factor into account. With different kernels for KPCA, the results are not satisfying and we believe non-linear transformation does not have many benefits for our models. However, with the modified rating matrix combined with PCA, the results improve, and especially the modified after-PCA-reduced achieve the recall@10 of 47%, which improves the recall@10 by 1% compared to original PCA-reduced and 6% higher than [2]. From these results, we believe that taking cuisine factor into consideration is one important step for performance improvement.

There are still many interesting potential improvements can be tried and discovered. For example, improve the size and quality of raw dataset, modify the preprocessing steps, try model blending or introduce some other models such as LFM. Besides the collaborative filtering, we actually experimented basic LFM model with Stochastic Gradient Descent, however the result was not satisfying and due to the time and resource limitation, we could not go further in this attempt, but some modifications and improvements on LFM model to adapt our project can be the future works as well.

7 Contribution of group members

- Zheng Huang: Mainly focused on the implementation of baseline and parts of improved models
- Yannan Huang: Mainly focused on literature review, improved ideas and experiment design
- Tianze Wei: Mainly focused on the implementation of KPCA and parts of improved models
- Luoye Wang: Mainly focused on literature review, improved ideas and experiment design

References

[1] Patrick Bittner. "Review Article A Survey of Collaborative Filtering Techniques". In: *Advances in artificial intelligence, 2009* 3400.Section 3 (2009), pp. 725–733. ISSN: 16175468. DOI: 10.1155/2009/421425.

[2] Paula Fermın Cueto, Meeke Roet, and Agnieszka Słowik. "Completing partial recipes using item-based collaborative filtering to recommend ingredients". In: *arXiv preprint arXiv:1907.12380* (2019).

[3] Marlies De Clercq et al. "Data-driven recipe completion using machine learning methods". In: *Trends in Food Science and Technology* 49 (2016), pp. 1–13. ISSN: 09242244. URL: http://dx.doi.org/10.1016/j.tifs.2015.11.010.

[4] Yehuda Koren. "Factorization meets the neighborhood: A multifaceted collaborative filtering model". In: *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2008), pp. 426–434. DOI: 10.1145/1401890.1401944.

[5] Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to information retrieval*. Cambridge university press, 2008.

[6] Michael J Pazzani and Daniel Billsus. "Content-based recommendation systems". In: *The adaptive web*. Springer, 2007, pp. 325–341.

[7] Francesco Ricci, Lior Rokach, and Bracha Shapira. *Recommender Systems Handbook*. 2011, pp. 1–35. ISBN: 9780387858203. DOI: 10.1007/978-0-387-85820-3.

[8] Badrul Sarwar, George Karypis, and Joseph Konstan. "Item-Based Collaborative Filtering Recommendation". In: *GroupLens Research Group/Army HPC Research Center Department of Computer Science and Engineering* (2001), pp. 286–295.

[9] Yummly. *'What's Cooking' from Kaggle*. 2016. URL: https://www.kaggle.com/c/whats-cooking/data.