Collaborative Filtering: Predict Ingredients to Complete Partial Recipes

Vilius Kaulinskas
s2125570
s2125570@ed.ac.uk

Alun Owen
s2108808
s2108808@ed.ac.uk

Adrian Pintille
s2116106
s2116106@ed.ac.uk

Jing Ting Chong
s2116497
s2116497@ed.ac.uk

Abstract

Nearly one-third of the food produced worldwide gets wasted annually. Over the past few years, there were many attempts to reduce food wastage: from advertisement campaigns to new governmental funding. As every individual plays a role in combating food wastage, this project implemented Collaborative Filtering (CF) using memory-based approach to build a recommendation system which suggests ingredients to partial recipes, motivated to help individuals in utilizing ingredients that could be otherwise disposed as waste. We implemented both user-based and item-based approaches using different similarity measures, and compared the performances. The user-based approach was also evaluated over the dimensionality reduced dataset derived from logistic PCA, however, the performance on the latter was inferior. The highest performance was achieved with item-based approach using cosine similarity, with 54% Recall@10 in predicting a missing ingredient from partial recipes.

1 Introduction

According to United Nations, approximately 931 million tons of food was wasted in 2019 [1]. When food is wasted and disposed through landfills, huge amount of methane is released into the atmosphere [2], resulting in climate change. In fact, by reducing food waste, 2 of 17 sustainable development goals set by the United Nations would be addressed (Zero Hunger and Climate Change) [3]. Recently, many websites and mobile applications have been designed to help utilize food ingredients that could otherwise be neglected and thrown out to waste. Examples of such can be seen on the Tesco[1] website. A commercial product fully based on this idea is the SuperCook mobile app[2] and website[3]. Given a list of ingredients, these applications suggest full recipes for the user or predict missing ingredients from a partial recipe. The problem of recommending items to users such as suggesting missing ingredients for partial recipe is known as collaborative filtering (CF) that has seen significant research over the last decades [4]. In particular, it is widely adopted in movie streaming services (e.g. Netflix) and e-commerce platforms (e.g. Amazon) where based on previous user activity such as item ratings, activity history, etc. users are presented with personalised recommendations of suggested items. One of the best known challenges in the area of CF that has attracted significant interest from community was the Netflix Prize [5] in 2006 where the goal was to beat Cinematch recommendation system

[1] https://realfood.tesco.com/what-can-i-make-with.html

[2] https://play.google.com/store/apps/details?id=com.supercook.app&hl=en_GB&gl=US

[3] https://www.supercook.com/#/recipes

used by Netflix platform at a time. The system was based on Pearson's correlation coefficient [6] that measures similarity between items and suggests movies that are most similar to the ones that user has liked. The winning team has developed algorithm that used a modification of singular value decomposition (SVD) algorithm [7] even though it was never used in practise [8]. In addition to these algorithms, there exist several other known algorithms that make recommendations for users (e.g. KMeans). Recently, deep learning methods were also applied for CF systems [9].

In this paper we discuss some of these CF methods in more detail and apply them to the dataset of 4236 recipes from 12 different cuisines. We start by conducting exploratory data analysis of the dataset followed by application of several dimensionality reduction methods in order to address the issue of high-dimensionality and sparsity of the dataset. We then implement memory-based algorithms, both user-based and item-based approach, with the goal of predicting missing ingredients from partial recipes. Finally, the performance on the two approaches is discussed and presented.

2 Exploratory data analysis

2.1 Numerical Data Description and Data Visualization

We begin the exploratory data analysis by taking a quick look at the data. The dataset is in the form of recipe/ingredient matrix: it contains 709 categorical attributes representing the presence (value of 1) or absence (value of 0) of different ingredients, and 4236 observations representing recipes belonging to specific cuisines. The total number of cuisines is 12 and there are no missing values in the dataset. The sparsity of dataset is 0.985 as defined by

$$S = 1 - \frac{|R|}{|I| * |U|} \tag{1}$$

Here, $|R|$ is total number of ingredients (ones), I - number of ingredients, U - number of recipes [10]. This reflects that only 0.015 of entries in data matrix are non-zero.

Due to the nature of the data (i.e., all attributes are categorical), and also the fact that most ingredients are present in only a small number of recipes, the mean of each attribute is close to zero. The same can be said about the standard deviation of the values in each attribute.

Next, we continue the exploratory data analysis by becoming familiar with the training data. The most frequent ingredient across all recipes is garlic, occurring in more than 2000 recipes as can be seen in Figure 1. The least frequent ingredients, such as gelatin, occur only one time. We also analyzed the number of ingredients across cuisines. The Japanese cuisine appears to have the lowest median on the number of ingredients for each recipe, while the Moroccan cuisine appears to have the highest median on the number of ingredients. Furthermore, as it can be seen in Figure 2 (left), the Thai cuisine appears to be the cuisine that contains the recipe with the largest number of ingredients (i.e., 31). Figure 2 (left) also shows that a significant number of outliers (in terms of the number of ingredients) are present in the dataset. While these recipes are outliers from a technical point of view, we decided not to remove them as the range of the number of ingredients is a sensible one.

We also analyse the frequency of the ingredients across cuisines. As it can be observed from Figure 1 (right), the predominant ingredients across cuisines are not unique. The onion represents the predominant ingredient in 5 cuisines. It is worth noting, however, that these 5 cuisines are very distinct from each other. In contrast, the soy sauce is predominant in two relatively similar cuisines (i.e., Chinese and Japanese). It is also worth noting that while soy sauce is only the 11^{th} most frequent ingredient across all cuisines, this ingredient is present in more than 70% of recipes in both the Japanese and the Chinese cuisines. Figure 1 (right) also shows that garlic is the most frequent ingredient in southern European cuisines (with the exception of the Thai cuisine). Lastly, the olive oil appears to be the most frequent ingredient only in the Greek cuisine.

Another aspect of the data that was investigated was the pairwise standard correlation of the ingredients. As Figure 2 (right) shows, the correlation matrix reveals meaningful information. For example, a strong positive correlation can be noticed between the rice wine and the soy sauce, ingredients that are oftentimes used together in recipes. The matrix also reveals negative correlations, such us the relationship between the olive oil and the soy sauce, ingredients that are rarely used together.

A further analysis reflects a significant difference between the average and the median values of each attribute. While the median value is zero for each attribute, the average is usually larger than zero. We consider this not to be caused by outliers, but by the nature of the attributes (i.e., categorical) and data (i.e. sparse and high-dimensional).

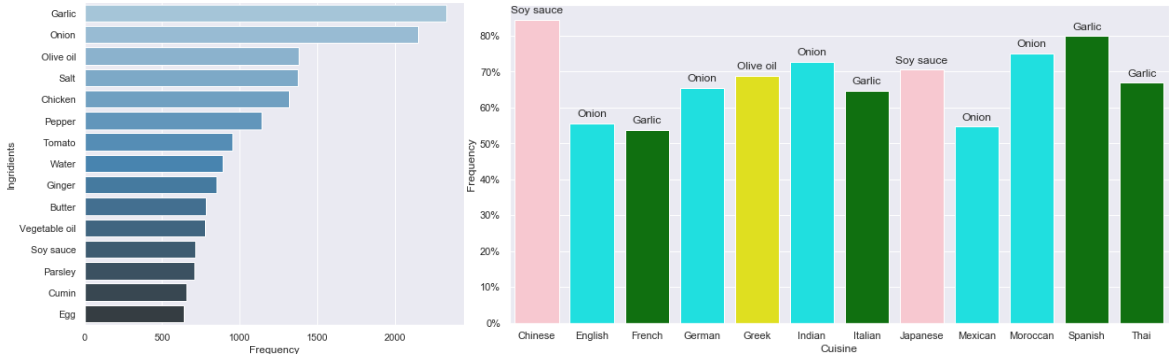

Figure 1: Ingredients frequency *(left)* and Frequency per cuisine *(right)*.

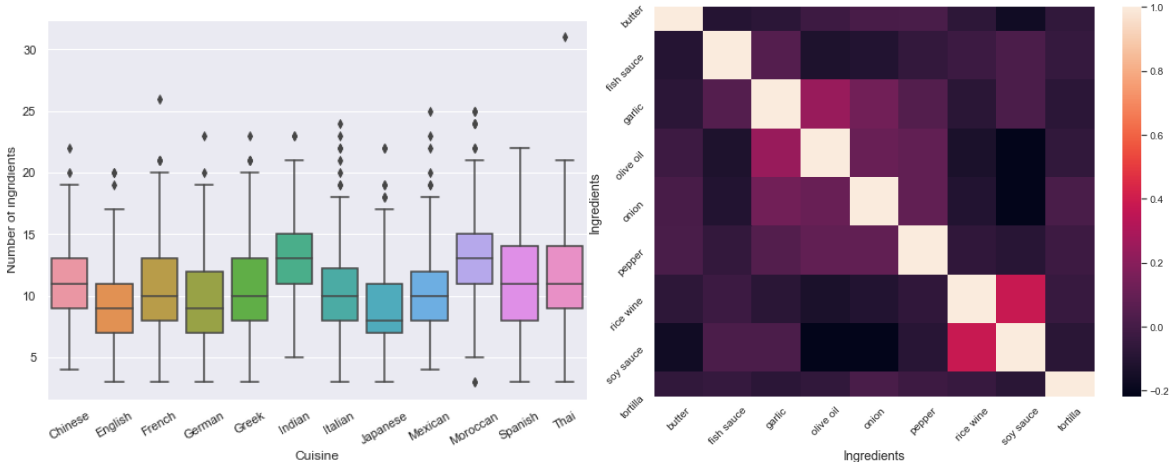

Figure 2: Boxplots for number of ingredients per cuisine *(left)* and Correlation matrix of certain ingredients *(right)*

2.2 Pre-processing

The dataset was cleaned of duplicated observations, and identical observations that belong to different classes (i.e., recipes that belong to multiple cuisines). The total number of observations removed is 17. The data was also centred and scaled to unit variance as input for PCA. For the other dimensionality reduction methods, centred data was used. Finally, a stratified sampling of the dataset was performed, keeping aside 33% of the dataset for a final evaluation of the performance of our models. We performed k-fold cross validation on the remaining 67% of the dataset during the training phase.

2.3 Dimensionality Reduction

In this section we explore both linear and non-linear dimensionality reduction methods to investigate the separation of data in a low dimensional space.

The data is generally not perfectly separated with any of the dimension reduction methods attempted. This is due to the data being sparse and binary as discussed previously. Each of the dimension reduction methods are visualised with their respective primary and secondary component as seen in Figure 3. Exploration of the dimension reduction began with Principal Component Analysis (PCA), its results based on two principal components demonstrate a cluster of European cuisines in the

Figure 3: Dimensionality reduction with PCA, Isomap, Logistic PCA, Sparse Logistic PCA, T-SNE and T-SNE applied on reduced data produced by Logistic PCA/

third quadrant of the coordinate plane. A positive value of the first principal component signifies the majority of the Asian cuisines in the dataset, this separation between the Asian and European cuisines is distinct, however, both Indian and Moroccan cuisines are clustered together near the intersection of both continent clusters with respect to the first principal component. The Indian and Moroccan cuisines both vary from the rest in terms of the second principal component. Although the few promising clusters appearing in the two principal components, two dimensions only account for 1.67% of the variance explained. To achieve 95% of the variance explained with PCA which is deemed as standard [11], it could only be done with a minimum of 554 dimensions. Other PCA methods were implemented through kernel PCA during experimentation, their results demonstrated similar findings or were of no value to the analysis.

Isomap, which performs classical Multidimensional Scaling in the geodesic space, provides a good separation of the data with Cosine metric using 10 neighbours. A small hyperparameter search was produced over different metrics and number of neighbours which realised the configuration. We can observe that Thai, Indian, and Moroccan cuisines are well separated with their own clusters, while the European cuisines, are poorly separated but show better separation in comparison to PCA. Similarly to the PCA, the primary dimension varies the Eastern and South-eastern Asian cuisines with the rest. Also the Chinese and Japanese cuisines occupy the same cluster likewise to PCA. Unfortunately, there is no evaluation metric similar to variance explained that can be used to compare the least amount of dimensions required to capture a large proportion of the data.

T-SNE yields the best results in a two-dimensional space in comparison to previous methods. The results exhibit a better trend in European cuisines, where all but the English cuisines are generally confined to their own section within the larger European cluster, whereas the English cuisine seems

at large settled around the perimeter of the German cluster. All non-European cuisines are very well separated in with the two dimensional representation from T-SNE. To achieve the dimensionality reduction performed, many distance measures were attempted; it was the cosine distance measure that best illustrated the different cuisines. Similarly to Isomap, T-SNE does not have a metric for comparison over larger dimensions.

Since previously mentioned methods of dimensionality reduction are not specifically designed for binary or sparse data, both logistic PCA[12] and sparse logistic PCA[13] were implemented to ideally further segment the data. Both implementations attempt similar groupings with two principal components, although much overlapping between cuisine clusters, small sections for the majority are captured. There is evidently no divide between German and English cuisines once again with both logistic methods. Between the two methods, it seems that the newest implementation of logistic PCA[12] produces a better two dimensional representation of the data than the sparse logistic PCA. The two methods shows greater variation within the continental clusters in comparison to normal PCA, however, it still seems inferior to that of T-SNE with two dimensions. In the manner that sparse logistic PCA is produced, a value for explained variance is not available[13]. Regarding the logistic PCA, its methodology allows for explained variance. With two dimensions it achieves an explained variance of 13.3% while it can capture 95% of the data with as little as 100 dimensions. This shows significantly better reduction from the logistic PCA in comparison to the standard PCA.

Additionally, a composition of the best PCA and non-PCA method was performed (i.e. logistic PCA and T-SNE). Firstly the data was reduced to 100 dimensions by logistic PCA, then T-SNE was performed on these 100 dimensions. Its performance was inadequate as clusters previously captured by other methodologies. It is interesting to note that combinations of cuisines such as Indian and Moroccan are on completely different sides of the principal component by the composition of methods while all others suggested similar properties between the two cuisines.

For following methodologies of the paper, when approaches open up to the use of dimensionality reduction, the logistic PCA will be opted for since the number of dimensions needed for a fair representation is known, even though the T-SNE demonstrated more promising results in the 2-D plane.

3 Learning methods

In this section, we discuss both memory-based methods and model-based methods. For this project, we chose to implement memory-based methods since they are widely popular in many recommendation systems [14] and common in literature. Although they can be weak in performance with sparse data [14] which hinders scalability of this approach for most real-world problems, however, it is easy to be implemented as there are no optimization involved in the training approach.

3.1 Memory-based

Memory-based methods, or sometimes referred to as the neighbourhood methods for collaborative filtering use solely the data in hand and non-parametric algorithms. It is segmented based on comparing users or items, which generally known as user-based and item-based approach (recipes-based or ingredients-based in our case). Generally, memory-based methods gather the most similar users/items given a user/item using appropriate similarity measures. Although item-based approach is more favorable in many cases compared to user-based approach due to better scalability and improved accuracy [4], in this project, both options were explored and the results were compared. In the following sections, the data at hand will be referred to as the user-item matrix which will be denoted as $r_{u,i}$ for recipe u and ingredient i.

User-based

User-based approach generally evaluates the interest of a target user by comparing with its neighbouring users defined by similar rating patterns. In our case, with the user-item matrix, the similarity between a partial recipe and all the others can be calculated, to obtain the M most similar recipes. From the M most similar recipes, we can recommend the top N ingredients which occur the most which is not present in the partial recipe.

Item-based [15]

In a similar fashion to the user-based approach, the similarity is calculated but between instead of the items (ingredients) rather than the users (recipes). Then, a weighted sum can be calculated which resembles a score $S_{u,i}$ for each recommended item i for a user u:

$$S_{u,i} = \frac{\sum_{j \in J} \rho_{i,j} r_{u,j}}{\sum_{j \in J} |\rho_{i,j}|} \tag{2}$$

where J is the set of existing ingredients of the partial recipe, and $\rho_{i,j}$ refers to the similarity between recommended item i and existing ingredient j. As the binary nature of the dataset has no ratings on the ingredients, $r_{u,j}$ is equals to 1 for all j in the set J. The highest N scoring ingredients are then chosen from a set of scores for the partial recipe.

Similarity Measures

A critical step in memory-based methods is to compute the similarity between users/items. Pearson Correlation is widely used in many recommendation systems that compares ratings [4], which in our case not applicable with binary data. With this in consideration, we implemented and compared the results using Cosine similarity, Jaccard similarity, and Hamming similarity. The equations in this section treat items(ingredients) as feature vectors to compute the similarities, nevertheless, the same is applied in the perspective of users(recipes), with a transposed matrix.

Cosine similarity computes the similarity between two feature vectors by measuring the cosine of the angle between them, as shown in Equation 4, where $r_{:i}$ represent the column vector of ingredient i. It therefore captures the orientation of feature vectors but not the magnitude, which in this case applicable for our dataset, as there is no "50$\%$ onion" or "100$\%$ onion".

$$\rho_{i,j} = \frac{r_{:i} \cdot r_{:j}}{\|r_{:i}\| \|r_{:j}\|} \tag{3}$$

With binary data at hand, Jaccard similarity (4) is also considered. It finds the similarities between finite sample sets, which for example in the case of user-based approach, can be interpreted as the size of the intersection of ingredients divided by the size of the union of the ingredients between two recipes. Jaccard similarity, in fact, is arguably a better option than Cosine similarity in our case as it does not consider joint absences to be similar. For example, it does not consider recipe A and recipe B to be similar because both recipes do not have "soy sauce", which is a desired property.

$$\rho_{i,j} = \frac{|r_{:i} \cap r_{:j}|}{|r_{:i} \cup r_{:j}|} \tag{4}$$

Finally, Hamming similarity (5) is also implemented due to its applicability on binary feature vectors with equal lengths. It measures the number of equal components between the two and divided by the length of vector:

$$\rho_{i,j} = \frac{\sum_{u=1}^{n} |r_{u,i} = r_{u,j}|}{n} \tag{5}$$

3.2 Model based

In our experiments, we have attempted implementing the non-negative matrix factorisation (NMF) method that has proved to be very powerful technique in recommender systems [16]. There, $r \times i$ (r - number of recipes, i - number of ingredients) recipe/ingredient matrix V is decomposed into two latent matrices: $r \times k$ W and $k \times i$ H whose product approximately recreates the original matrix V. k is a configurable hyperparameter: larger k means more personalised recommendations for recipe [17]. Based on the domain of the data, k features can be interpreted to have real-world meaning (e.g. how much the ingredient is associated with particular cuisine), however, we do not interpret the meaning of the components in our experiments.

4 Hyperparameter Selection and Performance Evaluation

To evaluate the models, the dataset (both original and dimensionality-reduced dataset) was split into training set and testing set, with the training set further separated to allow for k-fold cross validation.

During validation and testing, the objective was to determine the ability of the models to recommend the ingredients removed from complete recipes.

First, we compared the performance of using different methods (user- or item-based) and similarity measures on the validation set. Then, using the best performing similarity measure and number of neighbours for each method obtained from the validation sets, the algorithm was run to refit on the whole training data, and its generalisation performance was computed on the test set. Emulating real-world task, the recipes in the validation and test set were treated as "new recipes" not seen previously. Then, an ingredient was randomly chosen and removed in each of the "new recipes", and the algorithm had to predict the missing ingredients. Few performance measures have been used in literature such as the mean rank/median rank of the missing ingredients, and percentage of test recipes for which the missing ingredient is present in the top N recommended ingredients [18]. We adopted the latter and refer it as Recall@10 in this report.

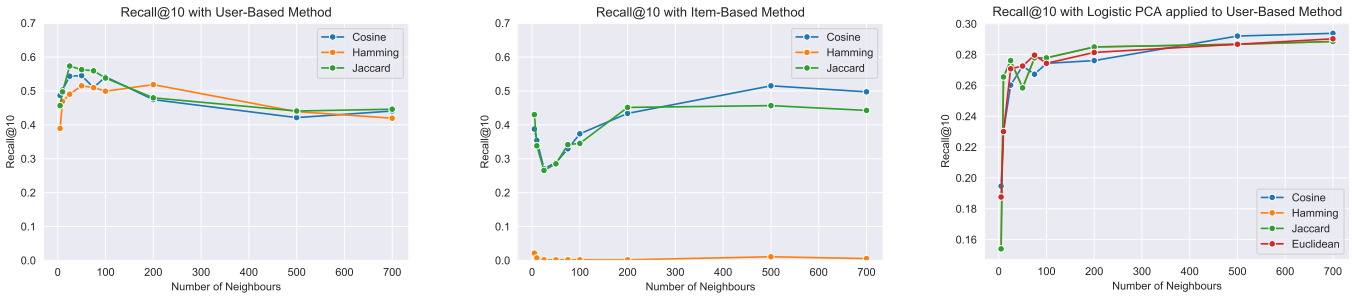

Figure 4: User-based cross validation scores *(left)*. Item-based cross validation scores *(center)*. Cross validation scores for the User-Based method with Logistic PCA *(right)*.

4.1 Hyperparameter Selection

When selecting similar recipes for user-based approach or selecting similar ingredients for item-based approach, the decision on the number of neighbours may greatly impact the performance. To tackle this, for each method, we performed 5-fold cross validation to obtain the cross validation scores over 5, 10, 25, 50, 75, 100, 200, 500, and 700 number of neighbours and obtained the results as shown in Figure 4.

We can see that the user-based method performs best when the number of neighbours is relatively small (i.e., in the 25-50 interval), and the performance decreases slightly as the number of neighbours increases. In contrast, the item-based method performs worst when the number of neighbours is small, and the performance increases as the number of neighbours increases. This may be explained by the distribution of the data. When the data is looked at as ingredients, the observations tend to overlap and a higher number of neighbours is required for a correct classification. On the other hand, when the data is looked at as recipes, more clear clusters can be distinguished, and a smaller number of neighbors is optimum.

It is also worth noting the difference in performance achieved with different similarity measures. The user-model approach performs best when the Jaccard similarity is used, achieving a result that is approximately 10% better than the lowest performing similarity measure (i.e., Hamming). In comparison, the item-based approach performs best when the Cosine similarity measure is used. The best result achieved by the item based method, however, is approximately 10% lower than of the user-based method. The Cosine and Jaccard similarity measures behave similarly for the item-based method, but the Hamming measures appears to perform extremely poor regardless of the number of neighbours.

The user-based model was also evaluated over the dimensionality reduced dataset (spanning 100 dimensions) that is derived from logistic PCA. This reduction produces an unbounded space rather than the binary space in the original data; because of this, the euclidean distance is also search on for the hyperparameter search. The search is evaluated on the same number as neighbours as previously. It was also reveals that the performance is at its best with 700 neighbours using the Cosine distance measure. Both Hamming and Jaccard distance measures achieve very similar results since they firstly

convert the data back into a binary form. At its optimum, this model achieves an average of 29% across the 5 folds which is nearly 30% worse than that without dimensionality reduction.

4.2 Performance Evaluation

Using the best hyperparameters found earlier, the performance of the methods was evaluated on the test set. As it can be observed from Table 1, contrary to the results found in the previous section, the item-based approach outperforms the user-based methods by a significant margin (i.e., 8%). Compared to the cross validation scores, the item-based generalization recall increases by approximately 5%, while the user-based generalization recall decreases by approximately 12%.

This may be explained by the fact that the user-based approach deals with a large number of observations when it computes the similarity, while the item-based approach looks at a significantly smaller number of observations. Another possible explanation can inferred from the distribution of the observations. Despite the fact that the same test set was used for both approaches, the distribution of the row vectors and the distribution of the column vectors is different. Therefore, the row vectors from the test set might not be as representative to their real distribution as the column vectors from test set.

Although several implementations of model-based approaches were attempted (primarily NMF and SVD), unfortunately, we were not able to to produce meaningful results from these experiments.

Table 1: The performance of the methods on the test

Method	Similarity Measure	Recall@10
User-Based	Jaccard	46%
Item-Based	Cosine	54%
User-Based with Logistic PCA	Cosine	23%

5 Conclusion

The exploration of collaborative filtering techniques revealed meaningful insights. Our work focused on memory-based methods, using different similarity measures. In retrospect of our experiments, these methods achieve satisfactory results.

Out of the methods implemented, the item-based model together with the cosine similarity measure achieves the highest recall on the test set (i.e., 54%), despite the fact of being outperformed by the user-based method on the training set. In addition, reducing the dimensionality of the dataset appears to drastically reduce the performance of the user-based model.

Upon reflection, the performance we achieved is similar to that of another paper [19] which evaluated an item-based model on sparse binary data, although the evaluation metric and datasets used are different, the paper achieves between 50% and 70% accuracy which indicates an upper bound on what can be achieved with such datasets and models. To further research with the sparse dataset, improvements may be obtained by using different metrics, or exploring in depth the model-based approaches which have shown great promise as well as methods based on deep learning techniques. A 2016 paper [20] that introduced a neural autoregressive model built for implicit data, it would allow for a greater insight into the performance of models mentioned in this paper.

References

[1] Food and Agriculture Organization. Technical platform on the measurement and reduction of food loss and waste. `http://www.fao.org/platform-food-loss-waste/en/`. Accessed: 2021-04-01.

[2] RecycleNow. How is food waste recycled? `https://www.recyclenow.com/recycling-knowledge/how-is-it-recycled/food-waste`. Accessed: 2021-04-01.

[3] United Nations. The challenge of reducing food loss and waste during covid-19. https://www.un.org/en/observances/end-food-waste-day. Accessed: 2021-04-01.

[4] Robert Bell Yehuda Koren. *Advances in Collaborative Filtering*. Springer, 2015.

[5] Stan Lanning James Bennett. The netflix prize. https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.115.6998&rep=rep1&type=pdf. Accessed: 2021-04-01.

[6] Joseph Konstan Badrul Sarwar, George Karypis and John Riedl. Item-based collaborative filtering recommendation algorithms, 2015.

[7] Michael Jahrer Andreas TOscher. The bigchaos solution to the netflix grand prize. *Journal of Multivariate Analysis*, 2009.

[8] Mike Masnick. Why netflix never implemented the algorithm that won the netflix $1 million challenge, 2012.

[9] Dit-Yan Yeung Hao Wang, Naiyan Wang. *Collaborative Deep Learning for Recommender Systems*. Wiley, 2015.

[10] Roberto Iriondo Saniya Parveez. Recommendation system tutorial with python using collaborative filtering, 2020.

[11] Graziela B. Eschera Bruno L.Ferreirab Rubén M.Maggio Daniel Granatoa, Janio S.Santosa. *Use of principal component analysis (PCA) and hierarchical cluster analysis (HCA) for multivariate association between bioactive compounds and functional properties in foods: A critical perspective*. Elsevier, 2018.

[12] Andrew J. Landgraf and Yoonkyung Lee. Dimensionality reduction for binary data through the projection of natural parameters. *Journal of Multivariate Analysis*, 180:104668, 2020.

[13] Seokho Lee, Jianhua Z Huang, and Jianhua Hu. Sparse logistic principal components analysis for binary data. *The annals of applied statistics*, 4(3):1579, 2010.

[14] Hao Ma, Irwin King, and Michael R Lyu. Effective missing data prediction for collaborative filtering. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 39–46, 2007.

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

[16] Yehuda Koren; Robert Bell; Chris Volinsky. Matrix factorization techniques for recommender systems, 2009.

[17] Wikipedia. Non-negative matrix factorization.

[18] Bernard De Baets Willem Waegeman Marlies De Clercq, Michiel Stock. Data-driven recipe completion using machine learning methods. *Trends in Food Science Technology*, 2015.

[19] Ayhan Demiriz. Enhancing product recommender systems on sparse binary data. *Data Mining and Knowledge Discovery*, 9(2):147–170, 2004.

[20] Yin Zheng, Cailiang Liu, Bangsheng Tang, and Hanning Zhou. Neural autoregressive collaborative filtering for implicit feedback. In *Proceedings of the 1st workshop on deep learning for recommender systems*, pages 2–6, 2016.

6 Contributions

Pre-interim report

Both Adrian and Jing Ting initially investigated the exploratory data analysis including the base work on dimensionality reduction. Alun and Vilius researched the different types of learning methods that are used for collaborative filtering.

Post-interim report

Jing Ting produced the backbone of the memory based models. Adrian also worked on the user-based method and also worked on more of the exploratory data analysis. Vilius worked on the model-based implementation. Alun developed the dimensionality reduction methods, and produced the user-based method using the dimension reduced dataset.

All the write up was divided equally between all four members as the work came up.