Recipe Suggestion for Meal-Sharing Online Marketplaces

Julien Robert
University of Twente
P.O. Box 217, 7500AE Enschede
The Netherlands
j.c.a.robert@student.utwente.nl

ABSTRACT

Meal-Sharing platforms, being part of the sharing economy, need strong user retention. Based on the principle that a recipe suggestion algorithm can help to increase user retention of such platform, this paper investigates the design of a recommender system for recipe suggestions. Different recommender system model are evaluated and compared using metrics such as meal sellability, RMSE, Recall and Precision. As no data is available of meal-sharing platform, this paper investigates on how to create a dataset that can be used by the recommender system. The comparaison of the models is done thanks a recommender system prototype.

Keywords

Suggestion Algorithm, Recommender Systems, Recipe Recommender System, Meal-Sharing Platform

1. INTRODUCTION & RELATED WORK

Meal-Sharing Online Marketplace are web platforms that have the goal to let *food-lovers* buy from (private) cooks. Many meal-sharing platforms failed due to the lack of users or their bad execution [8]. Their problem is their user retention [8].

Suggestion algorithms, also known as Recommender Systems, permit finding information of interests customized according to users' preferences [4]. Recommender Systems (RS) is a computer science topic researched since several years [2, 14] with already production ready techniques. Most early recommendation approaches are based on machine learning concepts or information retrieval [17]. According to *Alan Said et al.* [17], there is a difference between "*an item being recommended* to a user (recommendation) and *a query being posted* in order to find a relevant item (information retrieval)". This paper focuses on how to recommend a recipe to a meal-sharing platform application user. Recommender Systems have different approaches [19], only the collaborative filtering and content filtering approaches for performing location-based recommendation are investigated. *Collaborative filtering systems*, in our case, recommend recipes (*items*) based on the similarity between different users and or items (rating) [13]; on the other hand, *content-based systems* take in account properties of recipes. This means recommending recipes based on the user meal order history and ratings [13].

The business advantage of a recommender system is to increase the user retention, the user engagement and the cross-sales (for an online shop) [11]. While user engagement and retention are crucial for any online marketplace, the focus of the research is made on food related online marketplace such as meal-sharing platform. A Recipe Recommender System (RRS) for such platform can be useful to increase the user retention, based on the principle that people looking for a meal to cook would use the application, and hopefully sell the prepared meals in the platform.

The above mentioned RS approaches have different pros and cons. For instance, *collaborative filtering*, while being useful because not the whole characteristics of items has to be known, require a decent user base in order to be relevant and that users have rated the items. A non rated item cannot be recommended by the system. This problem is called the *cold start* issue. The target of this paper is for business owners or IT managers that want to improve their meal-sharing platform user retention via a recipe recommender system. It is then took in consideration that starting meal-sharing platform will not have at first a decent user base. Contrary to collaborative filtering, a *content-based filtering* requires extensive items' description [13]. *Content-based filtering* is hence not be subject to the cold start problem, however, by recommending items close to what the user purchase there is significantly less discovery of recipes than with the collaborative filtering method [15]. For a recipe, items information such as the ingredients of the recipes or its type of cuisine are hence required. This is easier information to get than acquiring users.

All of this together leaves us with trying to investigate how to build and create a recommender system, given not much or no data. Others than the best methods to choose or to combine, other decision must be made while designing a recommender system. The exhaustivity of the thoughts about the requirements was a hard thing to do before the research that has been on building a Recommender Canvas [18]. The requirements and the use of the Recommender Canvas are discussed in Section 3.

2. PROBLEM STATEMENT

The main contribution and topic of this paper is to investigate how to build a recipe recommender system for a meal-sharing platform. The research will contribute as well by introducting how to build a dataset for the meal-sharing platform recommender system, and sharing the end result (dataset). The goal is to help those platforms

33rd *Twente Student Conference on IT* July. 3rd, 2020, Enschede, The Netherlands.

to keep the users they need; given that the business owners agree with the assumption that a recommender system can help their business.

To fulfill these goals, the research answers the following questions:

1. **RQ1** What are the design requirements for a Recipe Recommender System?
2. **RQ2** How to prepare data to train a Recipe Recommender System?
   (a) How to efficiently scrape recipes from the Web?
   (b) How to generate user (preferences) data?
3. **RQ3** What state-of-art recommendation techniques should be used for a Recipe Recommender System?
   (a) How to evaluate the sellability of a meal?

3. DESIGN REQUIREMENTS

User retention is crucial for a peer-to-peer meal-sharing platform, so any possibility to increase it should be investigated. RS are a possible way of increase of user retention and engagement [10]; its effectiveness in the meal-sharing field needs to be proven, but for that, a Recipe Recommender Systems, must be first developed. As there is much research on RS, we can base ourselves on different existing (state-of-art) techniques and (possibly) combine them. The fact on combining different techniques has proven its effectiveness in the Netflix Challenge where the winning solution was a combination of different filtering algorithms [13].

The requirements of the recipe recommender system can be thoughts in two manners:

- Business Requirements, meeting the expectation of the meal-sharing platform
- System Requirements, design permitting to fulfill the business requirements

Concerning a few business requirements, the Recipe Recommender System should suggest meals that match the taste of the user. Those recipes must, as well, be beneficial to the surrounding users (match their taste). Those two requirements are made in assumption that it will increase the meals sellability on the platform and its user retention.

The research is based on the hypothesis that a meal-sharing platform contains the following data:

D.1 User basic information

D.2 User food preference

D.3 Orders Rating & History

D.4 Recipes data

D.1 contains limited and simple user information as the user ID (ID), full name (name) and its delivery location (latitude, longitude).

D.2 contains what characteristics a user likes about recipes. Those recipes characteristics are named *tags*.

D.3 contains all orders that took place in the platform and their ratings (user_id, recipe_id, rating).

D.4 contains the recipes' information, such as their categories (*tags*) and ingredients.

We can assume that the hypothesis is valid because of the necessity of these data for the good functioning of the platform.

The taste of the user can be measured using *user food preferences* and the sellability can be assumed with the *surrounding user food preferences*.

For designing such a recommender system, there needs to be some guidance to set precisely and organized the recommender system requirements. Thanks to the development of the Recommender Canvas by *G.van Capelleven et al.* [18], there is now a way to identify the key requirements of a Recipe Recommender System.

The Recommender Canvas has been used to define the requirements of the meal-platform suggestion algorithm, cf. Figure 1. The Capelleven's Recommender Canvas is composed of six main area:

RC.1 Goals

RC.2 Domain Characteristics

RC.3 Functional Design Considerations

RC.4 Technique Selection

RC.5 Interface Design

RC.6 Evaluation & Optimization

Those different aspects and the Recommender Canvas in itself find design requirements for a Recipe Recommender System.

The answers to the different sections of the canvas are straightforward, however techniques selections are explained further in Section 4.

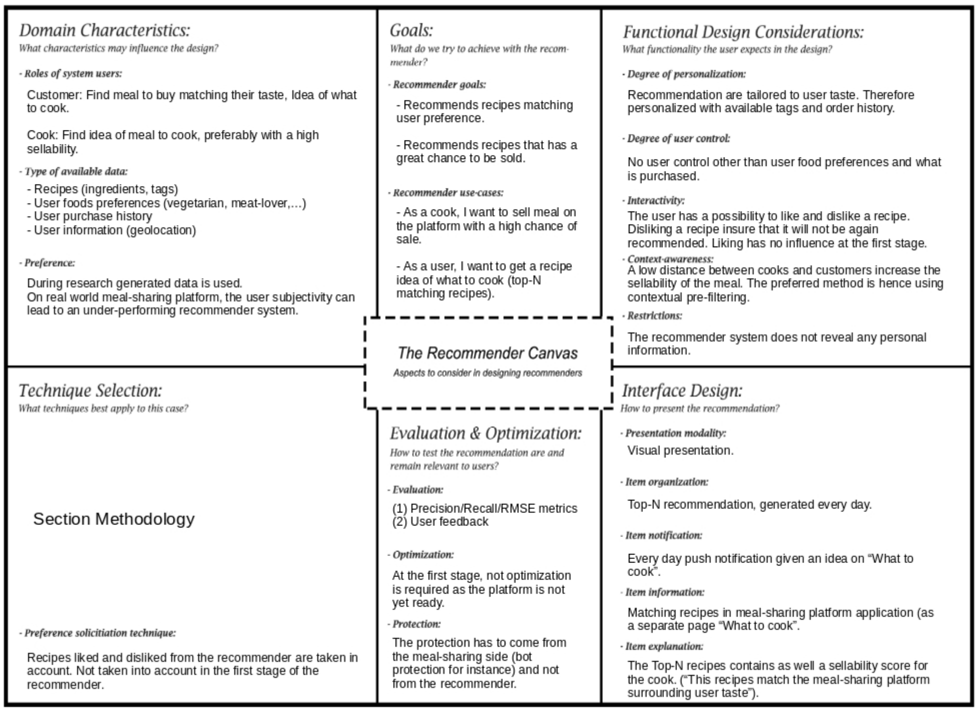
4. METHODOLOGY

There are different techniques to use for building a Recommender Systems. The focus is made on how perform some state-of-the-art methods for a recipe recommender system. We investigate the requirements of each method and compare them with the requirements and constraint of a meal-sharing platform. The research of *Candillier et al.* [6] resumes the different techniques used in collaborative, and content-based filtering.

To start with some definitions from *Candillier et al.*:

- "*user-based* approaches associate a set of nearest neighbors with each user and then predict the user's rating for unscored items using the ratings given by the neighbors on that item" [6]
- "*item-based* approaches associate an item with a set of nearest neighbors, and then predict the user's rating for an item using the ratings given by the user on the nearest neighbors of the target item" [6]
- "*model-based* approaches, and more specifically those based on clustering, tend to be more scalable, by constructing a set of user groups or item groups, and then predicting a user's rating for an item using the mean rating given by the group members." [6]

A priori none of the approaches needs to be yet eliminated as the dataset contains recipes information (D.4), and orders information (D.3). Item-based methods are however


Domain Characteristics:
What characteristics may influence the design?

- Roles of system users:

Customer: Find meal to buy matching their taste, Idea of what to cook.

Cook: Find idea of meal to cook, preferably with a high sellability.

- Type of available data:

- Recipes (ingredients, tags)
- User foods preferences (vegetarian, meat-lover,...)
- User purchase history
- User information (geolocation)

- Preference:

During research generated data is used.
On real world meal-sharing platform, the user subjectivity can lead to an under-performing recommender system.

Goals:
What do we try to achieve with the recommender?

- Recommender goals:

- Recommends recipes matching user preference.

- Recommends recipes that has a great chance to be sold.

- Recommender use-cases:

- As a cook, I want to sell meal on the platform with a high chance of sale.

- As a user, I want to get a recipe idea of what to cook (top-N matching recipes).

Functional Design Considerations:
What functionality the user expects in the design?

- Degree of personalization:

Recommendation are tailored to user taste. Therefore personalized with available tags and order history.

- Degree of user control:

No user control other than user food preferences and what is purchased.

- Interactivity:

The user has a possibility to like and dislike a recipe. Disliking a recipe insure that it will not be again recommended. Liking has no influence at the first stage.

- Context-awareness:

A low distance between cooks and customers increase the sellability of the meal. The preferred method is hence using contextual pre-filtering.

- Restrictions:

The recommender system does not reveal any personal information.

The Recommender Canvas
Aspects to consider in designing recommenders

Technique Selection:
What techniques best apply to this case?

Section Methodology

- Preference solicitiation technique:

Recipes liked and disliked from the recommender are taken in account. Not taken into account in the first stage of the recommender.

Evaluation & Optimization:
How to test the recommendation are and remain relevant to users?

- Evaluation:

(1) Precision/Recall/RMSE metrics
(2) User feedback

- Optimization:

At the first stage, not optimization is required as the platform is not yet ready.

- Protection:

The protection has to come from the meal-sharing side (bot protection for instance) and not from the recommender.

Interface Design:
How to present the recommendation?

- Presentation modality:

Visual presentation.

- Item organization:

Top-N recommendation, generated every day.

- Item notification:

Every day push notification given an idea on "What to cook".

- Item information:

Matching recipes in meal-sharing platform application (as a separate page "What to cook".

- Item explanation:

The Top-N recipes contains as well a sellability score for the cook. ("This recipes match the meal-sharing platform surrounding user taste").

Figure 1. Recommender Canvas [18] for a Meal-Sharing Platform.

most suitable in a starting meal-sharing platform as only the recipes data is available.

The research of *Candillier et al.* [6], conclude that collaborative-filtering methods result more often in better predictive performance compared to content-based filtering methods. While it is shown to be true for movies, we investigate if it still holds true for recipes.

Research [18, 6, 11] as well advice that user interaction with the system is important and it must be personalizable, in order to avoid the "one-visit problem". This is particularly important for a recipe recommender of this research as our assumption is that the user-retention of the meal-sharing platform will be high thanks to the recommender. The personalization can be done using the user food preferences, the model must hence allow personalization with the user food preferences and the surrounding user food preference (assumption on sellability).

Table 1 resumes which state-of-art recommendation techniques that could be used for our recipe recommender system. Only models giving personalized recommendation, in order to fulfill the above defined requirements, are included. For instance, *ItemPop*, which is not a personalized model and always recommend the top-N popular items to all users, is not included and thus not investigated. Moreover, implicit rating model such as *SVD++*, *WMRF* or *KNN Implicit* does not need to be investigated because the meals rating are always expressed. The table does not aim to be exhaustive. The way works the different methods are not relevant for this research but can be learned via [15, 5, 3, 2, 17]. A description of the different models from Table 1 is present below; the descriptions of the collaborative filtering methods are extracted form Gorse[1] source code and documentation[2].

1. **Keyword-based Vector Space Model** [15] is used for item representation. For a meal-sharing recipe recommender system, the item are represented by tags as in table 2. In our case, as the recipes already contain tags, hence TF-IDF is not necessary.

2. **KNN** [16] is a neighbors-based model that predicts ratings from similar users

3. **CoClustering** [9] is a novel collaborative filtering approach based on weighted co-clustering algorithm that involves simultaneous clustering of users and items.

4. **Baseline** predicts the rating for given user and item by $\hat{r}_{ui} = b_{ui} = \mu + b_u + b_i$. If user u is unknown, then the Bias b_u is assumed to be zero. The same applies for item i with b_i.

5. **SlopeOne** [12] predicts ratings by the form $f(x) = x + b$, which precompute the average difference between the ratings of one item and another for users who rated both.

6. **SVD** algorithm, as popularized by Simon Funk during the Netflix Prize. The prediction $\hat{r}_{ui}$ is set as: $\hat{r}_{ui} = \mu + b_u + b_i + q_i^T p_u$. If user u is unknown, then the Bias b_u and the factors p_u are assumed to be zero. The same applies for item i with b_i and q_i.

7. **BPR** means Bayesian Personal Ranking. It is a pairwise learning algorithm for matrix factorization /model with implicit feedback. The pairwise ranking between item i and j for user u is estimated by: $p(i >_u j) = \sigma(p_u^T(q_i - q_j))$

[1] https://github.com/zhenghaoz/gorse

[2] https://gorse.readthedocs.io/

Approaches	Type	Methods
Content-based	Keyword-based Vector Space Model [15, 2]	TF-IDF
Collaborative	Nearest Neighbors [2]	KNN [16]
	Clustering [2]	CoClustering [9]
	Regression-based [2]	Baseline, SlopeOne [12]
	Matrix Factorisation [2]	SVD, BPR

Table 1. Selected Filtering Approaches & Methods

5. EXPERIMENT

For privacy reason, time and lack of complete datasets, recipes data (D.4) has been scrapped from cooking websites, and the rest of the dataset (user data D.1, order history D.3 and food preferences D.2) generated. All the experiment has been performed using the programming language Go[3]. The code snippets are hence written in Go.

5.1 Web Scraping

Recipes data is information easily available on Internet. There are many recipes websites that contain recipes information in a somewhat structured manner. The choice has been made for *Albert Heijn Allerhande* website however the following would work with any recipe website.

During the experiment the web scraping has been facilitated by the Go package Colly[4].

```
//get title
c.OnHTML("h1.title.hidden-phones", func(e *colly.HTMLElement) {
  r.Title = e.Text
})

//get tags
c.OnHTML("section.tags", func(e *colly.HTMLElement) {
 e.ForEach("a", func(_ int, i *colly.HTMLElement) {
  r.Tags = append(r.Tags, strings.ToLower(strings.TrimSpace(i.Text())))
 })
})
```

The goal of the scraping was to build a dataset of recipes containing recipes information (D.4), such as recipes name, recipe ingredient and recipes categories (*tags*). The previous code snippet is an extract of the complete recipe scrapper. The scrapper loads the given recipe page and read the HTML tags in order to extract the recipe information. The code above for instance, read the value of the CSS class *h1.title.hidden-phones* and associate it to the title of the scrapped recipe. It can append that there is repetitive information available in the HTML page, such as recipes tags. We loop through every element having the CSS class *section.tags* and extract the tag. There is some data formatting happening at the same time by setting the tags lowercase and triming unnecessary scpaces.

Using the method above gives us only one recipe, as we want a dataset full of recipes we need to query multiple pages. *Albert Heijn Allerhande* have a hidden API that the website use itself that permits to get a list of recipes in JSON format. Sending a GET request to *Allerhande* API[5] permits to obtain a list of 100 recipes. Thanks to this API, we can with a simple script marshaling the JSON to get a list of recipes. We can then loop through that list of recipes in order to scrape them.

Recommender System need to have data in a certain format. The list of recipes must be in a matrix form [13] for an item profile. A simple script can transform an array of recipe models to a matrix, while performing data cleaning. Table 2 shows the output of that script, a matrix containing the recipes features (with 1 the recipe having the feature and 0 not):

[3]https://go.dev
[4]https://github.com/gocolly/colly
[5]https://bit.ly/2Z8QNKc

id	tag_vegetarisch	tag_amerikaans	ingredient_tomaat	ingredient_bieten
1	1	0	1	1
2	0	1	1	0
3	1	0	1	0

Table 2. Recipes Matrix Example.

In order to limit the number of recipe feature, the ingredient names have been processed. This is because *AH Allerhande* make a distinction between different type of the same ingredient. The cleaning task was manual (as in specifying word to clean as shown below) because the recipes data was in Dutch and text processing tools did not give a satisfactory result.

```
        data[i] = strings.ReplaceAll(data[i], "zongerijpte", "")
        data[i] = strings.ReplaceAll(data[i], "klein", "")
        data[i] = strings.ReplaceAll(data[i], "grote", "")
        data[i] = strings.ReplaceAll(data[i], "stukjes", "")
        data[i] = strings.ReplaceAll(data[i], "kruimige", "")
```

After the automatic scraping of 4950 recipes and data cleaning, we obtain a recipe matrix of dimensions 4950 (recipes) x 4048 (features); ready to be used by the Recipe Recommender System.

Unit tests permit to see the good functioning of the web scrapping.

5.2 Data Generation

Contrary to recipes data, we cannot simply get user data from Internet. The user data need to be generated. Each generated users was composed of a *name*, *latitude*, *longitude*, *food preferences*, *orders history* and *orders rating*. The *name*, *latitude*, *longitude*, are generated with a Go package named *gofakeit*[6]. The longitude and latitude generated are restraint within the coordinates bounding box of the Netherlands. This means we ensure every generated user are within the Netherlands.

```
        const minLatitudeNL = 51.1028
        const maxLatitudeNL = 53.5842
        const minLongitudeNL = 3.987
        const maxLongitudeNL = 7.8929
```

The *food preferences* required to be generated meaningfully. This mean that the data must make sense and not be a random character string for instance. What comes closest to *food preferences* are recipes *tags*, hence the food preferences of the user is generated based on the available recipes tags. This make a user food preference non-exclusive. One can like vegetarian food but it does not mean that person is vegetarian by the simple fact the tag is in the user food preferences. This leaves the freedom to select any random recipes matching the user preference.

An order is a meal that a user purchased. A meal is linked to a recipe and all orders are rated. We assume that the meal-sharing platform has the recipe of all meals posted on their platform and if the meals is not present, the cook has to add it.

The user order history and rating can then be generated randomly. The selection of recipes is done as following:

[6]https://github.com/brianvoe/gofakeit

1. A user is generated
2. The food preferences are generated randomly (with a maximum of 12 tags per user)
3. The recipes dataset is filtered to match the user food preferences
4. A random number of recipe is randomly taken from the recipe dataset (with a minimum of one order per user and a maximum of 50)

For this research 10 000 users has been generated. This low number of users helps to reflect the lack of users of the meal-sharing platforms. Unit tests permit to see the good functioning of the user generation.

The generated dataset is available online[7].

5.3 Recommender System

The Recipe Recommender System for meal-sharing platform is using the previously generated data and based on the methods from Table 1.

5.3.1 Collaborative Filtering

Collaborative filtering methods, from Table 1, have been implemented using the package Gorse[8]. All the different models has been trained on the *order history* dataset generated earlier. Each model have different hyperparameters. The best parameters has been found using the *GridSearchCV* function from Gorse. It permits calculating metrics and evaluate the given parameters that gives the best metrics. The given parameters to the function have been found with trial and error. What is considered the best parameters is the value that maximize the *Precision*, *Recall* and minimize the *RMSE*. *Sellability* has not been considered in the selection due to lack of available computation power.

For instance, the best parameters for *BPR* are with a *learning rate* of 0.1, *number of factors* of 50 and a *regularization stength* of 0.005.

For *BaseLine*, to maximize the *Precision* and the *Recall*, we use a *regularization stength* of 0.5, a *learning rate* of 0.1.

For *KNN*, the best parameters are a*learning rate* of 0.005, the *similarity function* is using MSD similarity, the *regularization stength* of 0.02. and 80 the *maximum k neighborhoods to predict the rating.*

And finally, *SVD*, has its best parameters with a *learning rate* of 0.005, *number of factors* of 10 and a *regularization stength* of 0.005.

For each model we use a relatively high *number of epoch* being equal to a minimum of 150. Trial and error showed that the metrics does not get better after 500 epochs.

We can maximize whether the *Precision* and *Recall* at the same time or minimize the *RMSE*. The rest of the investigated models does not require hyperparameters.

5.3.2 Content Filtering

The content filtering approach has been specified using a similarity matrix of the different items (recipes) using their tags and ingredients defined in their item profile (Table 2). The similarity between recipes has been calculated using their cosine similarity. An extract of the matrix can be found on Table 3, where R_x is a recipe of id x.

[7]https://bit.ly/2YGMCGx
[8]https://github.com/zhenghaoz/gorse

	R1	R2	R3	R4
R1	1	0	0.077850	0.261116
R2	0	1	0.149071	0.083333
R3	0.077850	0.149071	1	0.298142

Table 3. Extract Recipe Similarity Matrix

Creating the recipe similarity dataset is intensive operation and can become inefficient with a large recipe dataset (in our case already 4950 x 4950 matrix). Each new recipe added in the meal-sharing platform requires a similarity calculation with each recipe of the dataset. This is problematic for a meal-sharing platform as exhausitivy of recipes is impossible to achieve, which will lead to many more calculation required.

To make recommendation to a user, we based ourself on the user order normalized rating history. The weight of each tags of the user food preferences is calculated using the mean of all order ratings containing that tag.

The recipes with the highest ratings having the user top-3 tags will then be used for recommendation. We will recommend the top-3 most similar recipes to those previously selected recipes.

This method might prohibit discovery of new recipes and let the user in a recipe bubble. The meal-sharing business owner can decide if this is problematic for the vision of his/her platform.

5.3.3 Validation

Because no user testing has been possible during the experiment, mathematical measures have been used in order to test the performance and viability of each model. In the information retrieval field, there are some measures that can be used to test a recommender system. *I. Cantador et al.* listed the different measures that are usually used in order to test a recommender system offline. There is no consensus on which characteristics of a recommender system should be evaluated [5]. The decision has been made to evaluate *precision*, *recall* and *RMSE* based on Figure 2, on [7, 6], and because the users preferences are binary.

$$Precision = \frac{TP}{TP + FP} \qquad (1)$$

Precision is defined as the ratio of relevant items to recommended items [5]. Where TP means True Positive and is the relevant recomment recipes and FP, False Positive, is the non-relevant recommended recipes.

It permits telling how good the retrieved recipes match a user's interest and provides an indication of the quality of the recommendations [1]. It measures the classification accuracy of the recommender system [5].

$$Recall = \frac{TP}{TP + FN} \qquad (2)$$

Recall is the proportion of relevant items that have been recommended to the total number of relevant items [5]. Where TP means True Positive and is the relevant recomment recipes and FN, False Negative, is the non-recommended relevant recipes.

It gives an indication of how many recipes are recommended from the total number of relevant recipes [1]. It measures the classification accuracy of the recommender system [5].

$$RMSE = \sqrt{\frac{\Sigma_{i=1}^{N}\Big(p_i - v_i\Big)^2}{N}} \quad (3)$$

The RMSE evaluates the capability of a method to predict if a user will like or dislike a recipe [6]. It measures the prediction accuracy of the recommender system [5].

$$S = \sum_{j=1}^{R} \cos(\mathbf{u}, \mathbf{v_j}) = \frac{\mathbf{u}\mathbf{v_j}}{\|\mathbf{u}\|\|\mathbf{v_j}\|} = \frac{\sum_{i=1}^{n} \mathbf{u}_i \mathbf{v_j}_i}{\sqrt{\sum_{i=1}^{n} (\mathbf{u}_i)^2}\sqrt{\sum_{i=1}^{n} (\mathbf{v_j}_i)^2}} \quad (4)$$

Where R is the number of neighboring user at x kilometers radius; u the user recommendation and v the neighbor user j recommendation.

S is the notion of sellability has been mentioned thorough the paper. The sellability is the average of the cosine similarities between the user recommendations and the neighbors users recommendation in a x km radius. The sellability increases if a recommended recipe would please more than one user in a x km radius. It evaluates how likely a meal is going to sell with the assumption that a user being recommended a recipe will cook it and sell it on the platform.

6. RESULTS

The results of the experiment on the different models used by the recipes recommender system can be found on Table 4 The testing of the recommendation algorithms has been performed selecting random users from the generated dataset (generated user ID 8 for table 4). The test has been done by requesting 5 recipes to recommend to the user. The sellability has been calculated based on recommendation of users in a 5km radius of user 8 (resulting in 11 users).

The content-based filtering is not trying to predict the ratings of the recipes by selecting a Top-N recipes to recommend; this means there is no RMSE that can be calculated. The same goes for BPR as it is an item ranking algorithm. We can see that each model gives us different recommendations and very few algorithms contains common recommendations. The precision and recall are both pretty low while the RMSE is high. With a few numbers of predictions this is normal that the recall is low; this is because there are more recipes that could have been relevant that the one predicted. However, the high RMSE means that more fitting of the models is required. With this preleminary result we can still get a grasp on the performance of the different models.

The sellability of the recommended recipes seems to increase the less personalized are the recommendation. This makes sense as the less personalized the recommendation are the more they will be alike between user. With the definition we use about sellability this is a normal trade off to take into account.

Based on what is mentioned above, the best performing algorithm, taking in account the measures metrics, the sellability and the execution time seems to be BPR and the Content-Based Filtering in terms of precision and recall and CoClustering in terms of meal sellability. We will not select a best performing model based on RMSE because all of them are relatively close.

The results of the different models are as well subject to the model hyperparameters. Finding the best model is a big task but tweaking the model with the right parameters is another. The finding of the best parameters have been attempted but there is still room to improve if more computational power is available in order to test the models with more different parameters.

Moreover, apart *Recall*, *Precision* or *RMSE* metrics, execution time should be taken in account. The recommendation must be generated every day (according the Recommender Canvas) and the computational power must be available to do so. The execution time of the model should as well enter in account in the decision of picking a model. For instance, the building of the similarity matrix (used in KNN on the content-based method) is a intensive task and using it in production might need to be avoided. According to *Lops et al.*, the nearest neighbor algorithms are effective, but their most important drawback is their inefficiency at classification time [15].

7. CONCLUSION

We have discovered the system requirements and design requirements of a Recipe Recommender System by using the Recommender Canvas. The canvas has permitted us to completely think about the recipe recommender system before its implementation. It permitted reflecting on what data is available and how it should be used and to whom the recommender system is targeted. Furthermore, by thinking of what data can be available in a meal-sharing platform, we generated a dataset matching the defined characteristics. The generation of the user data has be thought and based on the scraped recipes from the web. By generating the data we have thought about how the data needs to be and how to generate reflectively of general platforms design while being ready to use by the recommender system. Moreover, the notion of sellability have been introduced. This is an extra metric that permit to measure how likely the proposed recipes to cook can sell on the platform. The sellability is evaluated by the average cosine similarity of the recommended recipes to the recommended recipes of users in a x km radius The selected recommender system model each has been evaluated using *Recall*, *Precision*, *RMSE* and *Sellability.* After running the experiment, and strictly based on those values, it seems that the best performing models are BPR and Content-Based Filtering in terms of Precision and Recall and CoClustering in terms of meal sellability.

However, for extending the value of this research on Recipe Recommender System for Meal-Sharing Platform, it might be interesting to test the obtained results directly with users. This might however be challenging as are there well-established meal-sharing platform, the models than however be trained with the dataset that this paper has produced[9]. Moreover, the notion of recipe sellability can be more researched by studying the behavior of a user being recommended a recipe; with how many are going to sell the dish on the platform and how many are going to purchase it –a too high $\frac{Availability}{Demand}$ ratio could have the inverse expected effect. Furthermore, a deeper analysis of the parameters of the collaborative filtering model may increase the measured metrics of the recommender system. Finally, future work could investigate the performance of hybrid filtering methods to see if there is any gain combining collaborative and content-based filtering for a recipe recommender system.

[9] https://bit.ly/2YGMCGx

Model	Precision@5	Recall@5	RMSE@5	Sellability@11	Recommendation
BaseLine	0.00007	0.00006	1.47615	0.20000	[845 1298 2470 3571 1080]
SlopeOne [12]	0.00344	0.00344	1.42902	0.08800	[354 330 430 168 700]
CoClustering [9]	0.00012	0.00010	2.27968	0.17946	[3571 3368 3460 3387 2470]
KNN [16]	0.00028	0.00062	1.47962	0.11854	[430 480 555 983 370]
SVD	0.00386	0.00463	2.27968	0.06621	[831 1951 1201 496 203]
BPR	0.14344	0.24350	/	0.15383	[7 3 4 19 18]
Content-Based Filtering	0.20000	0.00786	/	0.09329	[309 55 2 1515]

Table 4. Collaborative/Content-Based Filtering Methods.

8. REFERENCES

[1] Measures of effectiveness, Jan 2012.

[2] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge amd Date Enginrrting*, 17(6):734–749, June 2005.

[3] C. Anderson. A survey of food recommenders. 2018.

[4] D. Bianchini, V. D. Antonellis], N. D. Franceschi], and M. Melchiori. Prefer: A prescription-based food recommender system. *Computer Standards and Interfaces*, 54:64 – 75, 2017. SI: New modeling in Big Data.

[5] F. Cacheda, V. Carneiro, D. Fernández, and V. Formoso. Comparison of collaborative filtering algorithms: Limitations of current techniques and proposals for scalable, high-performance recommender systems. *TWEB*, 5:2, 01 2011.

[6] L. Candillier, K. Jack, F. Fessant, and F. Meyer. *State of the Art Recommender System*, pages 1–22. 04 2009.

[7] I. Cantador, I. Fernández-Tobías, S. Berkovsky, and P. Cremonesi. *Cross-Domain Recommender Systems*, pages 919–959. Springer US, Boston, MA, 2015.

[8] A. R. Davies, F. Edwards, B. Marovelli, O. Morrow, M. Rut, and M. Weymes. Making visible: Interrogating the performance of food sharing across 100 urban areas. *Geoforum*, 86:136 – 149, 2017.

[9] T. George and S. Merugu. A scalable collaborative filtering framework based on co-clustering. In *Fifth IEEE International Conference on Data Mining (ICDM'05)*, pages 4 pp.–, 2005.

[10] C. A. Gomez-Uribe and N. Hunt. The netflix recommender system: Algorithms, business value, and innovation. *ACM Trans. Manage. Inf. Syst.*, 6(4), Dec. 2016.

[11] D. Jannach and M. Jugovac. Measuring the business value of recommender systems. *ACM Trans. Manage. Inf. Syst.*, 10(4), Dec. 2019.

[12] D. Lemire and A. Maclachlan. Slope one predictors for online rating-based collaborative filtering. *CoRR*, abs/cs/0702144, 2007.

[13] J. Leskovec, A. Rajaraman, and J. D. Ullman. *Mining of Massive Datasets.* Stanford, Stanford, 2010.

[14] C.-J. Lin, T.-T. Kuo, and S.-D. Lin. A content-based matrix factorization model for recipe recommendation. In V. S. Tseng, T. B. Ho, Z.-H. Zhou, A. L. P. Chen, and H.-Y. Kao, editors, *Advances in Knowledge Discovery and Data Mining*, pages 560–571, Cham, 2014. Springer International Publishing.

[15] P. Lops, M. de Gemmis, and G. Semeraro. *Content-based Recommender Systems: State of the Art and Trends*, pages 73–105. 01 2011.

[16] X. Ning, C. Desrosiers, and G. Karypis. *A Comprehensive Survey of Neighborhood-Based Recommendation Methods*, pages 37–76. Springer US, Boston, MA, 2015.

[17] A. Said, A. B. Kouki, and A. P. deVries. A top-n recommender system evaluation protocol inspired by deployed systems. 2013.

[18] G. van Capelleveen, C. Amrit, D. M. Yazan, and H. Zijm. The recommender canvas: A model for developing and documenting recommender system design. *Expert Systems with Applications*, 129:97 – 117, 2019.

[19] R. Yera Toledo, A. A. Alzahrani, and L. Martínez. A food recommender system considering nutritional information and user preferences. *IEEE Access*, 7:96695–96711, 2019.

APPENDIX

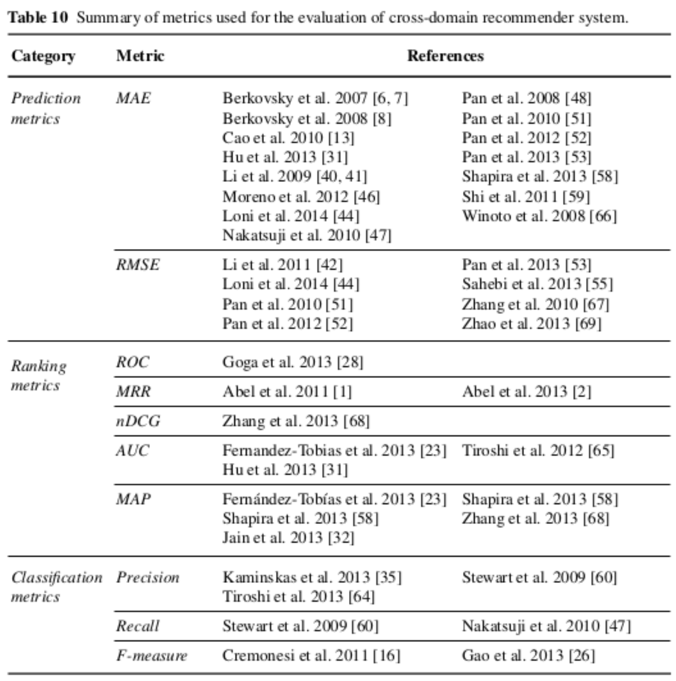


Table 10 Summary of metrics used for the evaluation of cross-domain recommender system.

Category	Metric	References	
Prediction metrics	*MAE*	Berkovsky et al. 2007 [6, 7] Berkovsky et al. 2008 [8] Cao et al. 2010 [13] Hu et al. 2013 [31] Li et al. 2009 [40, 41] Moreno et al. 2012 [46] Loni et al. 2014 [44] Nakatsuji et al. 2010 [47]	Pan et al. 2008 [48] Pan et al. 2010 [51] Pan et al. 2012 [52] Pan et al. 2013 [53] Shapira et al. 2013 [58] Shi et al. 2011 [59] Winoto et al. 2008 [66]
	RMSE	Li et al. 2011 [42] Loni et al. 2014 [44] Pan et al. 2010 [51] Pan et al. 2012 [52]	Pan et al. 2013 [53] Sahebi et al. 2013 [55] Zhang et al. 2010 [67] Zhao et al. 2013 [69]
Ranking metrics	*ROC*	Goga et al. 2013 [28]	
	MRR	Abel et al. 2011 [1]	Abel et al. 2013 [2]
	nDCG	Zhang et al. 2013 [68]	
	AUC	Fernandez-Tobias et al. 2013 [23] Hu et al. 2013 [31]	Tiroshi et al. 2012 [65]
	MAP	Fernández-Tobías et al. 2013 [23] Shapira et al. 2013 [58] Jain et al. 2013 [32]	Shapira et al. 2013 [58] Zhang et al. 2013 [68]
Classification metrics	*Precision*	Kaminskas et al. 2013 [35] Tiroshi et al. 2013 [64]	Stewart et al. 2009 [60]
	Recall	Stewart et al. 2009 [60]	Nakatsuji et al. 2010 [47]
	F-measure	Cremonesi et al. 2011 [16]	Gao et al. 2013 [26]

Figure 2. Summary of metrics used for evluation recommender system from *I. Cantador et al.* [7].