Recipe recommendations for individual users and groups in a cooking assistance app

Toon De Pessemier[1*], Kris Vanhecke[1], Anissa All[2], Stephanie Van Hove[2], Lieven De Marez[2], Luc Martens[1], Wout Joseph[1] and David Plets[1]

[1*]Information technology, imec-WAVES-UGent, iGent - Technologiepark 126, Ghent, 9052 , Oost-Vlaanderen, Belgium.
[2]Communication sciences, imec-MICT-UGent, Miriam Makebaplein 1, Ghent, 9000, Oost-Vlaanderen, Belgium.

*Corresponding author(s). E-mail(s): toon.depessemier@ugent.be;
Contributing authors: kris.vanhecke@ugent.be;
anissa.all@ugent.be; stephanie.vanhove@ugent.be;
lieven.demarez@ugent.be; luc1.martens@ugent.be;
wout.joseph@ugent.be; david.plets@ugent.be;

Abstract

Recommender systems are commonly-used tools to assist people in making decisions. However, most research has focused on the domain of recommendations for audio-visual content and e-commerce, whereas the specific characteristics of recommendations for recipes and cooking did not receive enough attention. Since meals are often consumed in group (with friends or family), there is a need for group recommendations, taking into account the preferences of all group members. Also cuisine, allergies, disliked ingredients, diets, dish type, and required time to prepare are important factors for recipe selection. For 13 algorithms, we evaluated the recommendations for individuals and for groups using a dataset of recipe ratings. The best algorithm and a baseline algorithm based on popularity were selected for our mobile kitchen experience and recipe application, which assists users in the cooking process and provides recipe recommendations. Although significant differences between both algorithms were witnessed in the offline evaluation with the dataset, the differences were less noticeable in the online evaluation with real users. Because of the cold-start problem,

the advanced algorithm failed to reach its full accuracy potential, but excelled in other quality features such as diversity, perceived usefulness, and confidence. We also witnessed a better evaluation (about half a star) of the recommendations by the more advanced cooks.

Keywords: Recommender System, Group Recommendations, Recipes, User assistance

1 Introduction

Recommender systems are tools and techniques that assist the users in finding the most interesting products or content thereby addressing the information overload problem of (online) services Ricci et al (2011). Personal preferences are extracted from the users' historical feedback (e.g., ratings) in order to suggest each user the most suitable items. Nowadays, a lot of open source software packages for recommendations exists Hug (2023); Scikit Learn (2023); Tensorflow (2023); Frederickson (2023); Ekstrand (2023b) each providing a set of algorithms, which makes if difficult to answer the question "which algorithm is the best one?". The answer may depend on the content domain of the items and characteristics of the dataset. For popular domains in the field of recommender systems, such as movies Abul-Fottouh et al (2020); Ma et al (2023) or music Biswal et al (2021); Bertram et al (2023), extensive evaluations have been performed. These domains received an exceptional amount of attention, also due to the fact that publicly available datasets exist for these domains such as the Movielens Grouplens (2023) and Million Song dataset Bertin-Mahieux et al (2023, 2011). For other domains, such as recipes, much less research has been performed into the best recommendation algorithm. This brings us to research question 1 (RQ1) of this study: "Which algorithm is the most accurate to recommend recipes?".

Another challenge is the usage of recommender systems for consuming items in group. Although the majority of the currently deployed recommender systems are designed to generate personal suggestions for individual users, in many cases content is selected and consumed by groups of users rather than by individuals. E.g., people go to restaurants, bars, and (cultural) events with their friends, movies or TV shows are often watched in a family context, and choosing a holiday destination is mostly a joint decision of the travel group.

These scenarios introduce the need for discovering the most appropriate group recommendation strategy. It is the method that is used to convert a recommendation algorithm for individual users, into a recommendation algorithm for groups of users, taking into account the (possibly conflicting) preferences of the group members.

In our study, we investigate these strategies in the context of a recipe recommender, which is a typical case for group recommendations. Indeed, family members or roommates often eat together on a daily basis. So, the choice of

their meal should reflect the preferences of all group members. A group recommender system for recipes can help to choose the meal that best matches the tastes of all people that will consume the food. This brings us to research question 2 (RQ2): "Which group recommendation strategy provides the most accurate recipe recommendations?'.

A distinction in group recommendation strategies can be made based on the place in the information flow where the data aggregation takes place. Group recommendations can be generated by aggregating the users' individual profiles with preferences into a preference profile of the group (i.e., aggregating profiles). Here, the data aggregation takes place at the input of the recommendation algorithm, so at the beginning of the information flow. As an alternative, group recommendations can be generated by aggregating the users' individual recommendations into recommendations for the whole group (=aggregating recommendations). In this case, the data aggregation takes place at the output of the recommmendation algorithm, so at the end of the information flow. In this paper, we refer to these strategies as *aggregation strategies*.

The first aggregation strategy (i.e., aggregating profiles with preferences) merges the preferences of different users of the group into group preferences. The starting point is the profile with personal preferences of each individual group member. These profiles are merged into a group preference profile reflecting the interests of all members. The specific merging method is an important aspect of this aggregation strategy, and therefore, different merging approaches have been proposed Masthoff (2004). But still, no consensus exists about the optimal solution to aggregate the members' profiles and how this aggregation performs Baltrunas et al (2010). In this study, we aggregate the users' profiles by taking the average of the users' ratings for a specific item. After aggregating the profiles of the group members, the resulting preference profile of the group is treated as a pseudo user. Subsequently, a general recommendation algorithm (for individual users) is utilized to calculate recommendations for the group based on this pseudo user profile. The advantage of aggregating profiles is the rather high probability of finding serendipitously valuable recommendations for the group in contrast to the strategy that aggregates recommendations (as described in the next paragraph) De Pessemier et al (2014a). The linked disadvantage is that aggregating profiles into a group profile may lead to group suggestions that lie outside the range of any individual recommendation list, which may be disorienting to the users and difficult to explain Quijano-Sanchez et al (2017).

The second aggregation strategy (i.e., aggregating recommendations) generates first recommendations for each individual user of the group using a general recommendation algorithm. Subsequently, the resulting recommendation lists of all group members are aggregated into one group recommendation list which (hopefully) satisfies all group members. Just like with the first strategy, different approaches to aggregate the recommendation lists have been proposed in previous studies De Pessemier et al (2014a). Most of them perform the aggregation based on the algorithm's prediction score, i.e., a prediction of

the user's rating score for the recommended item. A higher prediction score means a better match between the user's preferences and the recommended item. In this study, we adopted a classic aggregation method for the recommendation lists of the group members, more specifically, calculating the average of the prediction scores of the users' recommendation lists per item, and ordering the recommendations based on the resulting average score. The advantage of aggregating the recommendations of individual users into group recommendations is that the resulting recommendations can be linked directly to the recommendations of one individual of the group. This makes the group recommendations easy to explain based on the recommendations for the individuals (e.g., "the group gets this recommended because this item is recommended to two of the group members"). The disadvantage is that this link between the group recommendations and the individual recommendations makes it less likely to identify unexpected, surprising items De Pessemier et al (2014a).

Finally, we also evaluated the added value of recommendations for new users, with a limited number of ratings (10 in this experiment). Therefore, the recommendations of the most accurate algorithm were compared with the recommendations that are randomly sampled from the most popular recipes in the dataset. This research should answer the third research question (RQ3): "Do personalized recommendations result in better valued recipes compared to non-personalized recommendations?".

The contributions of this paper are as follows:

- It provides an extensive offline study of the usage of 13 state-of-the-art algorithms to generate recipe recommendations for groups of people.
- An online experiment with 54 participants evaluates various quality attributes of the algorithm that performs best in the offline study.
- A qualitative evaluation provides insights into the users' expectations and experiences with the proposed cooking assistance app.

The remainder of this paper is structured as follows. Section 2 discusses scientific work related to recommendations (Section 2.1), group recommenders (Section 2.2) and different strategies to generate group recommendations (Section 2.3). In Section 3, we discuss our research approach. We provide an overview of the used recommendation algorithms (Section 3.1), the dataset (Section 3.2), the used methodology to evaluate recommendations in an offline setting (Section 3.3) and how to handle group recommendations (Section 3.4). Next, we discuss our approach to evaluate the recipe recommender in an online study with real users (Section 3.5). In Section 4, the results of our study are provided. Firstly, the results of the offline evaluation with 13 algorithms (Section 4.1). Secondly, the results of the online study with users (Section 4.2). Finally, conclusions are drawn in Section 5.

2 Related Work

2.1 Recommender systems in different domains

In the research field of recommender systems, recommendations for individual users received most attention in various application domains such as movies Ma et al (2023); Choudhury et al (2021) or hashtags for movies Panchal and Prajapati (2023), points-of-interest De Pessemier et al (2014b), jobs Giabelli et al (2021), fashion Dong et al (2020), events in social networks Liao et al (2021), and in healthcare Tran et al (2021).

Food and diet are complex domains for recommender technology, among other things due to the requirement to sustain long term engagement Freyne and Berkovsky (2013).

Unhealthy diet and obesity can cause chronic diseases such as diabetes, which is still on the rise, especially in Western countries. Therefore, it is really important to reinforce health awareness and support self observation and behavior change using technological advances, such as data analysis of food consumption and web and mobile applications, to steer people towards a healthy diet that fits within their taste preferences Holzinger et al (2010). In this respect, recommender systems for meals or recipes can assist users in making a healthy choice while maintaining user motivation through personalized food recommendations that users are likely to appreciate.

The analysis of food recommenders is often based on datasets with user ratings on a set of recipes in order to judge the applicability and practicality of a number of personalization algorithms Freyne and Berkovsky (2013). Given the complexity of user restrictions (e.g., allergies, diabetes) and goals (e.g., diets), the interaction between human and machine is an important aspect of the food recommender. In this respect, Cora is an example of a conversational system that recommends recipes aligned with its users' eating habits and current preferences Pecune et al (2020). The results show that a conversational recommender system that engages its users improves users' perception of the interaction as well as their perception of the system.

Another example of a conversational recipe recommender shows that critiquing as a way of feedback is effective for conversational interactions Abbas et al (2021). This way, the user provides feedback on recommended items to refine subsequent recommendations. The authors also indicate the importance of diversity for recipe recommendations and they state that diversifying the recommended items during exploration can help increase user understanding of the search space Abbas et al (2021).

In our study, we also include a conversational aspect, which we call the onboarding. It helps us to capture user preferences, allergies, and diets, before the recommender makes a selection of recipes.

2.2 Group recommender systems

Although multiple recommender systems for recipes or food have been proposed Jia et al (2022), most of them ignore the characteristic that eating is often a group activity; so group recommendations would be more appropriate. In the late nineties, first scientific publications regarding recommender systems for groups were published McCarthy and Anagnost (1998). From then, many researchers have been investigating how the state-of-the-art recommendation algorithms for individual users can be adapted in order to generate group recommendations.

In 1998, one of the first group recommender systems, called MusicFX McCarthy and Anagnost (1998), was presented. The aim of MusicFX was to select background music for a group of people working out in a fitness centre. The recommender of MusicFX constructed a group profile based on the people present in the fitness centre by aggregating their individual music preferences and subsequently it selected a music channel including some randomness in the choice procedure to ensure variety. A quantitative assessment showed that the vast majority of fitness centre members who were involved in this trial were pleased with the group recommendations. Also for the content domain of movies, group recommendations are crucial to provide the best experience in case of a group activity. For the well-known movie recommender MovieLens, an extension called PolyLens enables recommendations for groups O'Connor et al (2001). This movie recommender utilizes the classic collaborative filtering to predict what users like based on their historical star ratings. For group recommendations, PolyLens uses an algorithm that merges the recommendation lists of the individual users; this is what we called the aggregating recommendations strategy in the introduction. The aggregation algorithm avoids movies that any member of the group has already rated (and therefore seen). To enable group recommendations, PolyLens allows users to create and manage their own groups of people who intent to watch the movie together. An evaluation with a survey as well as an observation of user behaviour showed that group recommendations are valuable and desirable for the users of PolyLens. This study also proved that users are willing to share their personal recommendations with the group, thereby trading some privacy for group recommendations.

Given the different strategies to compose group recommendations, the question rises which of these strategies is the best one. One of the first studies that compared group recommendation strategies was performed in the context of people watching TV together Yu et al (2006). This comparative study analyzed and compared three alternative strategies for generating group recommendations: a common group profile, aggregating profiles with preferences, and aggregating recommendations. A common group profile represents all group members and can be considered as a virtual user of the system. From a technical viewpoint, it is an easy solution to enable group recommendations, but it places all initiative and responsibility with the group members. Ratings or

feedback has to be provided for the group as a whole, so users cannot evaluate content individually. As a result, users have to find a way to convert the opinions of the group member into one group rating and make a consensus in case of disagreements. Because of this disadvantage, the common group profile was not considered as an option for group recommendations in our study. The comparison of the three strategies showed that aggregating profiles was the optimal solution for group recommendations for TV content Yu et al (2006). To aggregate the profiles, a method based on total distance minimization was proposed, which guarantees that the merged result is close to most users' preferences. The study concluded that the recommendation strategy was effective for multiple viewers watching TV together and that the resulting group recommendations were appropriately reflecting the preferences of the majority of the members within the group.

2.3 Comparison of group recommender strategies

In the domain of recipe recommendations, a recipe recommender for families has been proposed Berkovsky and Freyne (2010). Since family members typically eat a joint meal at least once a day, choosing a recipe and consuming the food are good examples of a group activity. In the context of this recipe recommender, the aggregating profiles strategy and the aggregating recommendations strategy were compared. An evaluation with a number of families showed that for users with a low density profile (i.e., having a small number of ratings), the aggregated recommendation lists yield slightly better results than the aggregated profiles. For users with a higher density profile on the other hand, the recommendations obtained by aggregating the users' profiles showed to be more accurate than the aggregated recommendation lists. This recommender system is based on collaborative filtering and the data of individual group members is aggregated in a weighted, domain-dependent manner, such that the weights reflect the observed interaction of the group members. As was already remarked by other researchers, this is only one type of recommendation algorithm and one of the many possible approaches for aggregating profiles or recommendation lists Baltrunas et al (2010). An extensive comparison of the two aggregation strategies is still missing in literature.

Research regarding the strategy that aggregates the individual recommendation lists into a list of group recommendations (cfr. aggregating recommendations strategy) has demonstrated that the influence of the data aggregation method is limited Baltrunas et al (2010). The data aggregation method specifies how the recommendation lists for individual users are aggregated into a group recommendation list, and is often based on the score that the algorithm assigns to the recommended items. The study compared group recommendation lists generated using four commonly-used aggregation methods (least misery, Borda count, Spearman footrule and the average) and found similar results in terms of accuracy for all methods. In addition, these group recommendations were compared to recommendations for individuals (i.e., recommendations for a single user) in terms of accuracy. The results showed that

for small groups, the group recommendations were only slightly less effective than the individual recommendations. In contrast for large groups, the group recommendations were significantly inferior than the individual recommendations. Moreover, the study revealed that if the groups are selected in such a way that the members have preferences that are quite similar, the effectiveness of group recommendations does not necessarily decrease when the group size increases. However in practice, groups are not always composed of like-minded people with similar preferences.

To cope with conflicting opinions in group decision making, consensus feedback mechanisms have been proposed. An example is the general harmony degree, which determines the before/after feedback difference as the difference of the feedback on the original and revised opinions Cao et al (2020). Another approach is to identify inconsistent subgroups and reach consensus through a two-stage feedback mechanism Wang et al (2022). Since these solutions are based on aspect such as trust degree scores and / or personalized feedback on a proposed solution, they require additional input or interactions from the group members. This makes them less suitable for group recommender systems, which often consider thousands of alternative items for which this input would be needed.

To conclude, we see no agreement in literature about the best way to generate group recommendations. In different domains, different strategies have shown to be the best. Limited research has been done in the domain of group recommendations for recipes. Similar work that investigated the best aggregation strategy was limited to one algorithm, the traditional collaborative filtering Berkovsky and Freyne (2010). In contrast in our research, we thoroughly investigate the two different strategies to generate group recommendations by comparing their accuracy for 13 different algorithms. Moreover, our research performs a two-stage evaluation approach. Firstly, the algorithms are evaluated for groups and for individual users in an offline setting with a dataset. Secondly, the best performing algorithm of the offline evaluation is benchmarked against a baseline algorithm in an online evaluation with 54 real users. According to our knowledge, our study is the first that evaluates a large number of state-of-the-art algorithms for group recommendations. Whereas most research regarding (group) recommendations is performed in the domain of movies O'Connor et al (2001) or music McCarthy and Anagnost (1998), we chose for the domain of recipe recommendations. This domain is not only less explored by researchers, it is also very characteristic for group recommmendations, since eating is often a social activity.

3 Method

3.1 Algorithms for Offline Evaluation

Through an offline evaluation on the dataset, we assessed the accuracy of the algorithms that are part of the Lenskit recommender toolkit. Version 0.13.1 was used; at the time of the user tests, this was the most recent version of

Lenskit. For all algorithms, the parameters where chosen based on tests with the dataset and the default values (starting values of the test). The following algorithms were investigated.

- **BiasedMF**. This is the classic biased matrix factorization (MF) algorithm trained with alternating least squares (ALS) and suitable for explicit feedback data Ekstrand (2023a). As solver for the optimization step, coordinate descent Takács et al (2011) was used. This solver is adapted for a separately-trained bias model and to use weighted regularization as in the original ALS paper Zhou et al (2008). The number of features to train was set to 50 and the number of iterations was 20. For the regularization factor 0.1 was chosen. The damping factor for the underlying mean was set to 5.
- **FunkSVD**. This Singular Value Decomposition (SVD)-like algorithm is a regularized biased MF technique trained with feature-wise stochastic gradient descent Funk (2006). The number of features was set to 50 and the number of iterations was 100. The learning rate was 0.001 and the regularization factor was set to 0.015. The damping value was set to 5.
- **IMF**. This is the implicit matrix factorization (IMF) algorithm trained with ALS Hu et al (2008) of Lenskit. If its input data contains rating values (as in our case), these will be used as the 'confidence' values. The number of features was set to 50 and the number of iterations was 20. The regularization factor was set to 0.1.
- **Impl_ALS**. This is an implicit-feedback recommender algorithm of Ben Frederickson's 'implicit' library Frederickson (2023) that connects with Lenskit. The prefix 'Impl' refers to this implicit library. Although our dataset contains ratings, this algorithm considers the data as implicit feedback. The algorithm is a factorization model that is trained with the ALS method Hu et al (2008). We set the number of features to 50 to be consistent with the other algorithms, although the default value is 100. The regularization factor was 0.01. The number of iterations was 15.
- **Impl_BPR**. This is also one of the implicit-feedback recommender algorithms of the 'implicit' library Frederickson (2023). Bayesian personalized ranking (BPR) Rendle et al (2009) is a recommender model that learns a MF embedding based on minimizing the pairwise ranking loss. The number of features was 50 (default value is 100); the learning rate was 0.01; and the regularization factor was 0.01.
- **ItemItem**. This is a traditional item-item nearest-neighbor collaborative filtering with ratings Linden et al (2003). The (maximum) number of neighbors for scoring each item was set to 20. The minimum number of neighbors for scoring each item is 1.
- **Pers. Mean**. This is the user-item bias rating prediction algorithm that calculates a personalized (Pers.) mean. The rating prediction is calculated as the sum of the global mean rating, the item bias, and the user bias. The damping factor was set to 0.0, meaning there is no damping.
- **Random Popular** This algorithm selects a set of random items from the most popular items in the dataset. The popularity of the item is determined

based on the rank that is calculated with the ratings for that item. This is the only algorithm that generates recommendations that are not personalized according to the ratings of the user that receives the recommendations.

- **Scikit_SVD** This algorithm implements biased MF for implicit feedback using SciKit-Learn's SVD solver Scikit Learn (2023). This is a pure SVD implementation. The algorithm operates by first computing the bias, then computing the SVD of the bias residuals. The number of features was set to 50; the damping factor was 5.
- **TF_BPR** The algorithms with the prefix 'TF' use TensorFlow for the optimization Tensorflow (2023). This algorithm implements BPR with MF, optimized with TensorFlow. The number of features was set to 50; the regularization factor was 0.02, and the batch size was 10.
- **TF_BiasedMF** This algorithm implements the standard biased matrix factorization model, like BiasedMF, but learns the model parameters using TensorFlow's gradient descent instead of the alternating least squares algorithm. The number of features was set to 50; the damping factor was 5; the regularization factor was 0.02, and the batch size was 10.
- **TF_IntegratedBiasMF**. This is a biased MF model for explicit feedback, optimizing both bias and embeddings with TensorFlow. This implementation uses TensorFlow to fit the entire model, including user/item biases and residuals, and uses TensorFlow to do the final predictions as well. The number of features was set to 50; the regularization factor was 0.02 both for the embedding vectors and the bias vectors, and the batch size was 10.
- **UserUser**. This is the traditional user-user nearest-neighbor collaborative filtering with ratings Goldberg et al (1992). The (maximum) number of neighbors for scoring each item was set to 20. The minimum number of neighbors for scoring each item is 1.

3.2 Recipe Data Set

To evaluate the recommendation algorithms and the grouping strategies, we used a dataset from an online recipe platform that allows users to consult and review the recipes. Besides a textual description of the recipe with ingredients and preparation method, the original dataset contains also 573,678 reviews with a rating score for a recipe. These review scores are used as input for the algorithms. In total 62,986 recipes and 321,442 different reviewers are available in the dataset. However, many of these users provided only a few ratings, which makes it difficult to generate accurate personal recommendations. In other words, they suffer from the cold start problem. A commonly-used solution is to ignore the users who have only a few ratings, and limit the recommender to generate only recommendations for users with at least T ratings, where T is a threshold value. This strategy is also applied in the Movielens dataset Grouplens (2023). Each user in the MovieLens dataset has rated at least 20 movies. In our recipe datasets, we ignore the users with less than 10 ratings. This results in a new dataset with 4,620 users. In total, the new dataset contains

95,478 ratings of these "more active" users, which are used to train and test the recommendation algorithms.

3.3 Offline Evaluation

To evaluate the recommendation algorithms, the data set is first split into training set and test set. The training users are users for whom all their ratings are added to the training set and used for training of the algorithm. It is a random selection of 80% of the users. The remaining 20% of the users are the test users. For these test users, 80% of their ratings are added to the training set so that the algorithm can learn their preferences. The remaining 20% of the ratings of the test users are added to the test set for evaluating the algorithm. This partitioning of the data per user is a common practice for evaluating recommender systems and is also one of the standard evaluation methods in Lenskit.

For each user-item pair, the recommender calculates a prediction of the rating score. Subsequently, the items are ordered by their rating prediction, and the top 10 items are selected as recommendations for the user. These top 10 items are used for evaluating the recommender.

The accuracy of the recommendations is evaluated using a commonly-used metric, Normalized Discounted Cumulative Gain (nDCG). The nDCG is a ranking metric that measures whether the algorithm ranks the items in the same order as the user. To calculate the nDCG, first the DCG (Discounted Cumulative Gain) of a list of recommendations is calculated. It accumulates the gain of a list of items and multiplies the gain with a discount factor based on that item's position in the list.

$$DCG(u, L) = \sum_{i=1}^{|L|} gain(u, i) * disc(i) \tag{1}$$

Where L is the ordered list of items offered to the user u, as recommendations. As common practice for data sets with ratings, the gain is defined as the rating the user u gave to item i, i.e r_{ui}:

$$gain(u, i) = r_{ui} \tag{2}$$

The discount factor for each item is proportional to its position within the list and defined as follows:

$$disc(i) = \frac{1}{log_2(rank(i) + 1)} \tag{3}$$

Where $rank(i)$ is the position of the item within the list. Because of this discount factor, items that are presented first are more important than items at the back of the list.

This mimics the behavior of real users in the way that users' attention will be more drawn to and focused on the first items of the list. Gradually, their attention typically weakens over time towards the end of the list. After calculating the DCG of the proposed recommendation list, it can be normalized as follows:

$$nDCG(u, L) = \frac{DCG(u, L)}{DCG(u, L_{ideal})} \tag{4}$$

Where L_{ideal} is the ideal list to present to the user. This ideal list is a list of items, ordered according to the user's ratings from highest to lowest rating. These ratings can be available in the test set for evaluation. Of course, this ideal list is limited to the ratings that are available for the user and will almost never be the true ideal list for that user. The normalized discounted cumulative gain has a value between 0 and 1, the higher the better. After calculating the nDCG for each individual user, the mean nDCG over all users of the dataset U is calculate as evaluation metric of the algorithm.

$$nDCG = \frac{\sum_{u=1}^{|U|} nDCG(u, L)}{\mid U \mid} \tag{5}$$

This process of calculating the mean nDCG over all users is repeated for every algorithm of Section 3.1. For each algorithm, the calculations are repeated in three iterations and the mean value of the nDCG of these iterations is reported in Section 4.1.

3.4 Evaluating Group Recommendations

Besides recommendations for individual users, we also want to evaluate group recommendations for groups of people. A major issue in the domain of group recommender systems is the evaluation of the effectiveness, i.e., comparing the generated recommendations for a group with the true preferences of the group. Performing online evaluations or interviewing groups can be partial solutions but are not feasible on a large scale or to extensively test various combinations of alternative configurations. Therefore, we chose to perform an offline evaluation with a dataset with ratings.

However, a data set with ratings originating from groups of people is, according to our knowledge, not available for research purposes. In the literature, group recommendations have been evaluated several times by using synthetic groups of users. Baltrunas et al (2010) used the MovieLens data set to simulate groups of different sizes (2, 3, 4, 8) and different degrees of similarity (high, random) with the aim of evaluating the effectiveness of group recommendations. Chen et al (2008) also used the MovieLens data set and simulated groups by randomly selecting the members of the group to evaluate their proposed group recommendation algorithm. They simulated group ratings by calculating a weighted average of the group members' ratings based on the users' opinion importance parameter.

Quijano-Sanchez et al (2010) used synthetically generated data to simulate groups of people in order to test the accuracy of group recommendations for movies. In addition to this offline evaluation, they conducted an experiment with real users to validate the results obtained with the synthetic groups. To measure the accuracy of the group recommendations in the online experiment, they created groups of participants and asked them to pretend that they are going to the cinema together. One of the main conclusions of their study was that it is possible to realize trustworthy experiments with synthetic data, as the online user test confirmed the results of the experiment with synthetic data. This conclusion justifies the use of an offline evaluation with synthetic groups to evaluate the group recommendations in our experiment. Also in our previous work on group recommendations De Pessemier et al (2014a), we used synthetic groups to evaluate different methods for aggregating the data.

For our research, we adopted the evaluation procedure of the group recommendations, as proposed by Baltrunas et al (2010). This is performed as follows. Firstly, artificial groups are composed by selecting random users from the data set. All users are assigned to one group of a predefined size. Secondly, group recommendations are generated for each of these groups based on the group members' ratings in the training set. Since group recommendations are intended to be consumed in group and to suit simultaneously the preferences of all members of the group, all members receive the same recommendation list. Thirdly, the recommendations are evaluated individually as in the classical single-user case, by comparing (the rankings of) the recommendations with (the rankings of) the items in the test set of the user, as explained in Section 3.3.

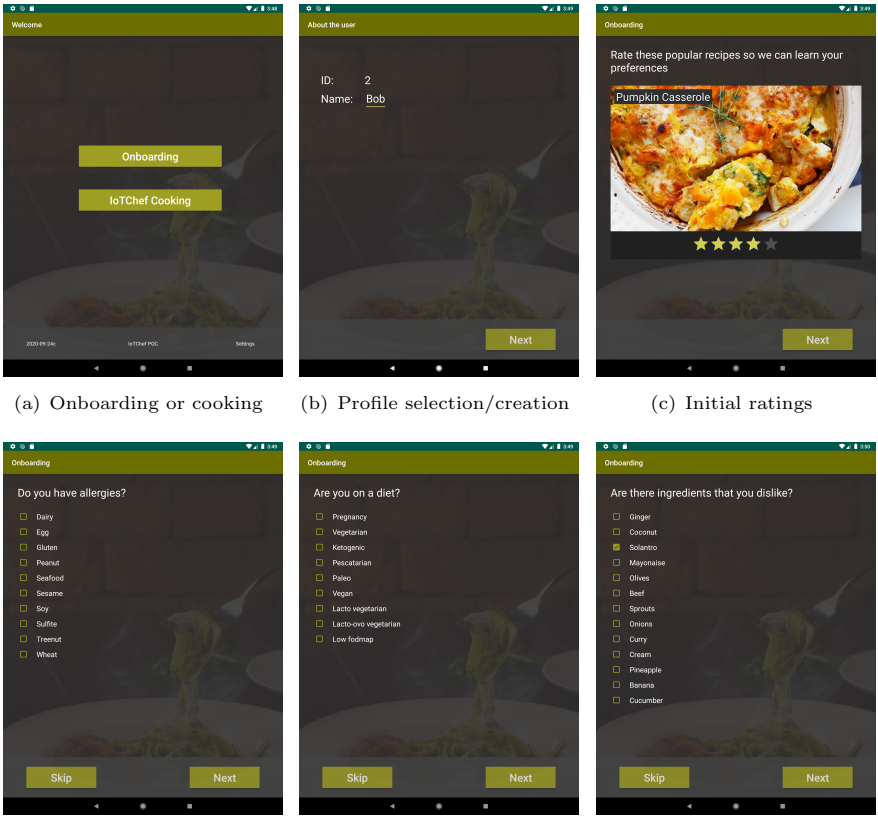
This way, we evaluated group recommendations based on the two different aggregation strategies and using the different algorithms of Section 3.1, and compared these results with the recommendations for individual users. These results are discussed in Section 4.1

3.5 Recommender for Online Evaluation

The online evaluation consisted of three phase. Firstly a focus group was organized to derive the users' expectations. Secondly, the users could test our cooking assistance app while preparing a recipe. Thirdly, we assessed the users' findings with the app and the recommendations through a survey.

3.5.1 Focus group

The goal of the evaluation with real, potential users was to obtain insights in what users think is important for a recipe app. A couple of months before the online evaluation of the app and the recommendations, a focus group was organized with 12 participants to assess the expectations that people have for a recipe app. When sampling the participants of the focus group, the aim was to obtain a diverse mix of participants based on their cooking profile (beginner=4, intermediate=3, advanced=3, professionals=2), gender (5 men, 7 women) and



(a) Onboarding or cooking (b) Profile selection/creation (c) Initial ratings

(d) Allergy specification (e) Diet specification (f) Ingredients specification

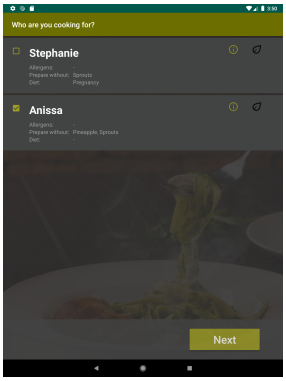
Fig. 1 Screenshots of the app with the onboarding and profile creation

age ($M = 33.93$, $SD = 10.84$). The results of this focus group are discussed in Section 4.2. The findings of the focus group were used in the design of the recipe recommmender app.

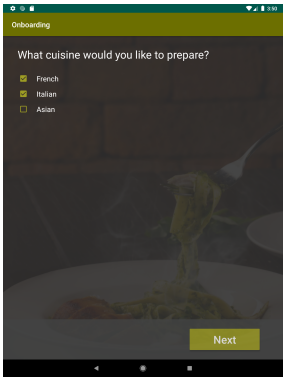
3.5.2 Cooking assistance app

The cooking assistance app consists of 2 parts: the onboarding and profile creation as visualized in Figure 1 and the recipe recommendations, recipe selection and cooking assistance as visualized in Figure 2. When starting the app, the user can choose between the onboarding process and the cooking process (Figure 1(a)). Next, the user can specify her name (Figure 1(b)), which serves as a login for the retrieval and update of an existing profile, or as identifier for the creation of a new profile.

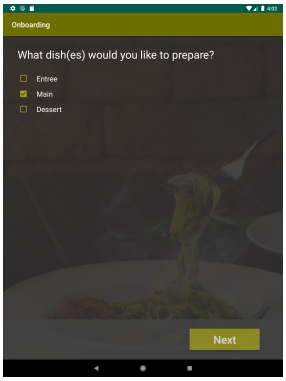
If the user has chosen for the cooking process, the application jumps to the screen of Figure 2(a), where the user can specify for whom the recommendations are intended. If the user chooses for the onboarding, she can create or update her profile. In the onboarding phase, the users are asked to specify their



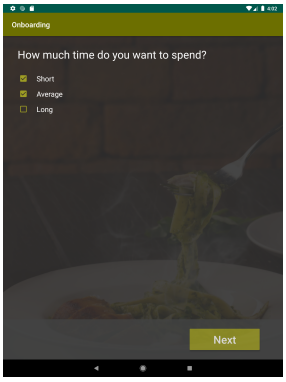
(a) Group specification



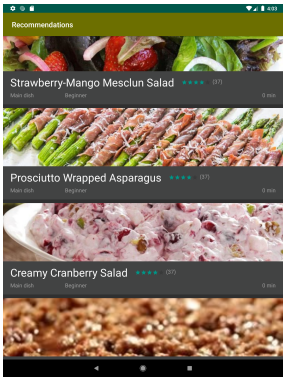
(b) Cuisine specification



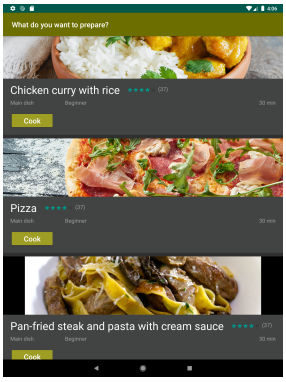
(c) Dish type specification



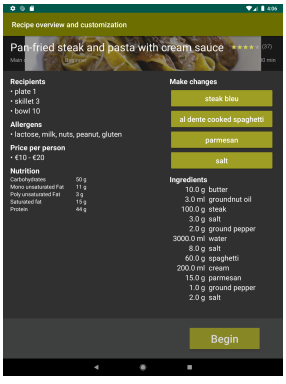
(d) Available time



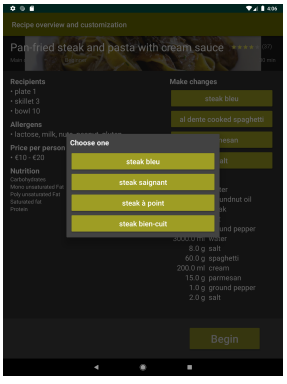
(e) Personal recommendations



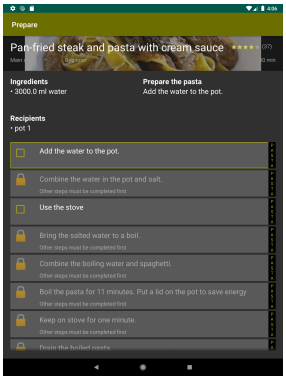
(f) Recipe selection



(g) Recipe overview



(h) Cookedness specification



(i) Step by step instructions

Fig. 2 Screenshots of the app with the recommendations, recipe selection and cooking assistance

ratings for 10 well-known recipes in order to create an initial profile with preferences. These ratings help to cope with the cold-start problem, that occurs when new users enter the system and have not specified enough ratings to calculate accurate recommendations Natarajan et al (2020). Figure 1(c) shows how this rating process is performed. The users can see the title of the recipe with an associated photo and can provide a star rating on a five-point likert scale. This screen is repeated 10 times to obtain a first set of ratings to calculate the recommendations. Subsequently, the users are asked to specify their food allergies, as visualized in Figure 1(d), so that the recommender can take these restrictions into account. In the next screen (Figure 1(e)), the users can specify whether they are on a diet. The users can also specify whether they dislike a certain ingredient as illustrated in Figure 1(f). Recipes containing a disliked ingredient will not be recommended, or the ingredient is replaced by an alternative, if this is possible. The preferences gathered in these screens will be used to construct the user profile that serves as the input for the recommender system.

After specifying the ratings, allergies, diets, and disliked ingredients, the users can specify for whom they will cook in the screen of Figure 2(a). If the users chose the cooking option instead of the unboarding, this screen is also shown after the users have input their name. The input of Figure 2(a) determines the type of recommendations: for an individual user or for a group of users. If recommendations for a group of people have to be generated, the aggregating recommendations strategy, as explained in Section 1, is used, since this strategy outperformed the aggregating profiles strategy in the offline test of Section 4.1. These group recommendations take into account the allergies, diets, and disliked ingredients of the group members by not suggesting any recipe that violates any of these restrictions. In the subsequent screens, more information is gathered about the properties of the dish that users want to cook. This information is used to narrow down the number of potential recipes that can be recommended and this way provide more targeted suggestions. In the screen of Figure 2(b), the users can narrow down the recommendations to a specific cuisine. In the next screen of Figure 2(c), users can specify the type of dish: entree, main or dessert. A final filtering based on the time required to prepare the dish is visible in Figure 2(d). Subsequently, a list of 10 personalized recommendations is presented to the users. In Figure 2(e) only the top-3 recommendations are visible, but the users can scroll down to see the other ones. The users can select a recipe and get more information such as difficulty and the estimated time to prepare the dish (Figure 2(f)). On the next screen (Figure 2(g)), the users get more details about the recipe such as the needed ingredients. Moreover, the users can personalize the recipe by changing some aspects such as the cookedness of meat and pasta, or the addition of herbs and spices. Figure 2(h) shows how the cookedness of the meat can be changed. This is important for the instructions for cooking the recipe that users get. Finally, Figure 2(i) shows these step-by-step instructions for preparing the dish. Through a checklist, the users can indicate which steps

have been performed, so that the app can follow the users' progress and give new instructions.

3.5.3 Survey

In order to obtain a better understanding of the cooking experience while using the app and the user experience with the recommendations, a test of the app and linked survey have been conducted in a prototype kitchen with 54 participants. Users were asked to use the app, do the onboarding, inspect the recommendations, select a recipe, and cook the meal based on the instructions of the app. For evaluating the recommendations, we performed a within-subjects experiment. Users received two different lists, one with recommendations of the popular recommender, which can be considered as a baseline, and one list with recommendations of the Implicit ALS algorithm, as described in Section 3.1 because this algorithm showed to be the best in the offline evaluation (Section 4.1). Users were not aware of the different algorithms that were used to generate these lists. Because of the within-subjects design, we obtained 54 user evaluations for each of both recommender lists, which allows a quantitative comparison of the two algorithms.

This quantitative comparison was performed to assess how pleased the users are with the recommendations. Firstly, users were asked to evaluate each of the 10 recommendations according to their preferences on a 10-point rating scale. Which set users received first, was counterbalanced. Half of the users received firstly the recommendations of the Implicit ALS algorithm and secondly the recommendations of the popular recommender. The other half received the recommendation lists in reverse order. Users were told the aim was to evaluate different algorithms and thus believed both recipe sets were personalized. In reality, the list with random popular items did not take into account the ratings of the users and was thus not personalized.

Subsequently, some questions were asked to assess the quality of the recommendation lists as a whole. Table 1 shows the questions of this survey, together with the quality attribute of the recommendations that we wanted to evaluate. These questions are selected from the ResQue framework, a user-centric evaluation framework for recommender systems Pu et al (2011). Users had to evaluate the two sets of recommendations on a 5-point scale: (1) Disagree completely (2) Disagree (3) Neutral (4) Agree (5) Agree completely. For the processing of the results, we mapped these answers to the numbers 1 to 5. In Section 4.2, we will discuss the results of this survey.

So in this experiment, the independent variable was the type of recommendation algorithm that was used, the dependent variables are the users' answers evaluating the algorithm, and the control variables are the test environment characteristics that remain constant: the experiment was performed in a kitchen environment during weekdays.

4 Results

4.1 Offline evaluation

Figure 3 shows the accuracy in terms of nDCG for the different recommendation algorithms. For each algorithm, the results are shown in a set of three bars of a similar color. The bars that indicate the accuracy of the recommendations for individual users are named: BiasedMF, FunkSVD, IMF, Impl_ALS, Impl_BPR, ItemItem, Pers. Mean, Random Popular, Scikit_SVD, TF_BPR, TF_BiasedMF, TF_IntegratedBiasMF, and UserUser. For each of these algorithms, the recommendations for individual users are compared with the recommendations that are generated for groups of users. In Figure 3, groups of 2 people have been created and the two aggregation strategies have been compared. The bars with the suffix "AggrProf" show the nDCG of the group recommendations that are generated by aggregating profiles of users. The bars with the suffix "AggrRec" show the nDCG of the group recommendations that are obtained by using the aggregating recommendations strategy.

Figure 3 also compares the different algorithms and shows that the algorithm based on alternating least squares of the implicit library (Impl_ALS) generates the most accurate results, with an nDCG around 0.18 for the recommendations for individuals. Although this is an algorithm designed for implicit feedback, it works very well for rating data sets too. Also the implicit matrix factorization (IMF) algorithm and the algorithms using Bayesian personalized ranking (Impl_BPR and TF_BPR) have a high accuracy in comparison with the other algorithms. This answers RQ1: "Which algorithm is the most accurate to recommend recipes?". Based on these results, we opted to choose Implicit ALS as algorithm for the online evaluation of Section 3.5.

The bars (with suffix "AggrProf" and "AggrRec") that represent the group recommendations are typically lower than the bars of the individual recommendations. This could be expected, since these group recommendations may have to make a compromise in case of conflicting preferences of the group members. We also witness that the group recommendations using the aggregating profiles strategy have in most cases (for 12 of the 13 algorithms) a lower accuracy than the group recommendations using the aggregating recommendations strategy. Since the aggregating profiles strategy aggregates that data at the beginning of the information flow to create a complete new group profile, the final group recommendations may differ significantly from the recommendations of the individual users. As a result, these group recommendations may be more serendipitous at the expense of a lower accuracy. This answers RQ2: "Which group recommendation strategy provides the most accurate recipe recommendations?'.

Figure 4 shows the accuracy in terms of nDCG for the individual users and for groups of 6 persons. Here we see similar conclusions. Impl_ALS generates the most accurate recommendations and the accuracy of group recommendations is lower than the accuracy of individual recommendations. If we compare Figures 3 and 4, we see a lower accuracy for the group recommendations

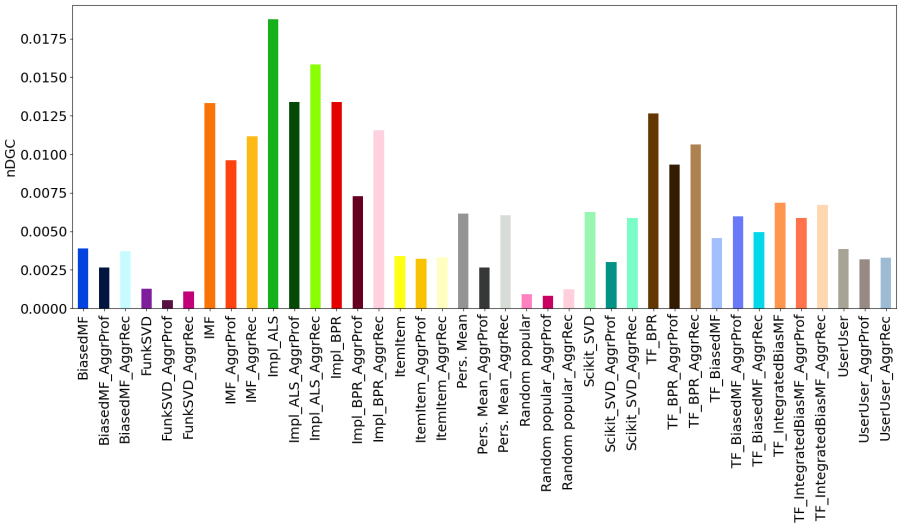


Fig. 3 Evaluation of the recommendation algorithms for individual users, and for groups of 2 users using the aggregating profiles strategy and the aggregating recommendations strategy

for groups of 6 persons. This can be explained as follows. As the group size increases, generating accurate recommendations for the group becomes more difficult. More preference profiles with possibly conflicting preferences have to be aggregated and more compromises have to be made.

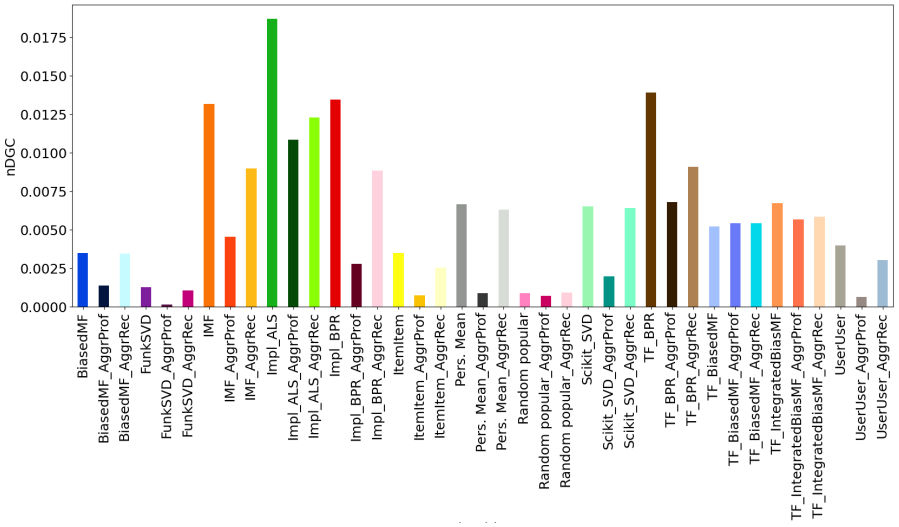


Fig. 4 Evaluation of the recommendation algorithms for individual users, and for groups of 6 users using the aggregating profiles strategy and the aggregating recommendations strategy

4.2 Online evaluation

4.2.1 Focus group

In the focus group that was organized before the design of the app, the following findings were concluded:

1. Users expect a personal experience, but also like to add their own accents. A personal touch can be enabled by personalized recommendations.
2. Users also expect that everything happens implicitly, in the background, almost magically. They want recommendations without giving too much input.
3. According to the participants, companies realize that it is no longer just about interactions between people and computers, but more about the experience around the cooking app. How do you create a surprising and unexpected experience?

These findings of the focus group indicate the importance of personalized recommendations in a recipe app for people. The first finding clearly focuses on personalized recommendations that can be fine-tuned with a personal profile. The second finding shows that users prefer a system with automatic recommendations, for which they have to provide no or limited input and feedback. The third finding emphasizes the importance of serendipity and surprise regarding the recommendations.

4.2.2 Survey

After the focus group and the development of the app, users were asked to use and evaluate the app and the recommendations as discussed in Section 3.5. After the onboarding phase, users received their recipe recommendations. Subsequently, users were asked to rate the 10 recommendations generated with the non-personalized recommender (random popular algorithm) and the ones generated with the personalized recommender (Implicit ALS algorithm). They were not aware of the different algorithms used.

Table 1 shows the results of this evaluation of the recommendations and the survey regarding the quality attributes of the recommendations. The personalized recommendations are generated with the Implicit ALS algorithm based on the ratings gathered in the onboarding phase. In contrast, the set of non-personalized recommendations was generated with the random popular recommender without taking the ratings of the user into account. For each question, the results of the best performing algorithm are shown in bold.

Row 0 of Table 1 shows the mean value of the ratings provided by the users on a scale of 10 for the non-personalized and personalized recommender. The personalized recommendations received a better evaluation from the users. However, the difference was small (Table 1). This small difference between personalized and non-personalized recommendations can be explained as follows. Users typically have a preference for the popular recipes, and give therefore a

Table 1 Questions, quality attributes and the resulting answers of the survey

No	Quality attribute	Question	Non Personalized	Personalized
0	Recommendation quality p=.18	I would evaluate this recommendation on a scale of 10	6.49	**6.63**
1	Accuracy p=.44	The recipes recommended to me matched my interests	**3.54**	3.52
2	Novelty p=.25	The recommended recipes are innovative for me	3.56	**3.65**
3	Novelty p=.39	The recommender system helped me discover new products	3.81	**3.85**
4	Transparency p=.36	I understood why these recipes were recommended to me	3.30	**3.35**
5	Perceived usefulness p=.18	The recommender helped me find the ideal recipe	3.02	**3.15**
6	Perceived usefulness p=.28	Using the recommender system to find what I wanted was easy	3.69	**3.76**
7	Perceived usefulness p=.41	The recommender system gave me good suggestions	3.54	**3.57**
8	Confidence p=.18	I am convinced of the quality of the recipes that were recommended to me	3.24	**3.39**
9	Confidence p=.27	I am confident I will like the recipes recommended to me	3.37	**3.44**
10	Confidence p=.13	The recommender made me more confident about my decision	2.98	**3.15**
11	Trust in system p=.50	The recommender system can be trusted	**3.43**	**3.43**
12	User intention p=.44	I would like to use this recipe recommender again	**3.67**	3.62
13	User intention p=.44	I would use this recommender regularly	**3.46**	3.44
14	Sharing experiences p=.36	I would tell my friends about this recommender	3.57	**3.61**

rating that is not much lower than the rating for the personalized list. Moreover, because of the cold start, the Implicit ALS algorithm might not be as accurate as in the offline evaluation. In the online evaluation, all users had exactly 10 ratings. Gathering more ratings might further increase the accuracy. In addition, some recommended recipes were not so familiar for the participants. This might be due to the fact that we used an American database of recipes for our Belgian test users. This is also reflected by the high rating on recommendation novelty for both sets of recipes (Table 1).

Rows 1 to 14 of Table 1 show the mean value of the answers of the users on the survey on a scale of 5. The personalized algorithm achieved the best

results for 10 of the 14 questions. So the personalized recommender scores better than the non-personalized recommender for the majority of the quality attributes such as novelty, transparency, perceived usefulness, confidence in the recommendations, and sharing experiences with friends.

The non-personalized algorithm performs the best for 3 questions, more specifically for the quality attribute "user intention" (Questions 12 and 13) and for Question 1 that assesses the attribute "accuracy". It is surprising that the personalized algorithm does not perform better in terms of accuracy than the non-personalized algorithm, although the difference is small (0.02) for Question 1. The evaluation of the individual recommendations (row 0 in Table 1) also assesses the accuracy of the recommendations and shows that the personalized recommender scores slightly better than the non-personalized recommender. Therefore, there is no clear winner between the personalized and non-personalized recommender in terms of accuracy. One explanation might be that the popular recommendations are more recognizable for the user and receive therefore a good score in terms of accuracy. The recognizability of recipes might also be the reason why the non-personalized recommender scores better than the personalized recommender in terms of user intention. If users recognize some popular recipes, they might be more willing to use the app again, and use it more regularly. For one question, we have a tie, more specifically for question 11 that assesses the user's trust in the system.

The results of Table 1 give an answer to RQ3: "Do personalized recommendations result in better valued recipes compared to non-personalized recommendations?". For 10 out of the 14 questions, and for the evaluation of the individual recommendations, the personalized list was better evaluated. But based on the rather small differences between both recommendation lists, we can conclude that 10 ratings per user is not enough for completely accurate recommendations with state-of-the-art algorithms in the context of a recipe recommender.

Before the actual testing of the app, the users were partitioned into groups according to their cooking skills. Table 2 shows the number of users of each cooking level, together with the mean of the ratings they gave to the personalized recommendations. The table shows a higher mean rating for people with better cooking skills. We performed an ANOVA analysis to tests if the different cooking levels have a different mean rating value. The test resulted in a p-value of 0.24. So, on a confidence level of 0.05, no significant difference between the 3 groups could be identified with ANOVA. However a T-test between profile 1 and profile 3 resulted in a p-value of 0.07. And the T-test between profile 2 and profile 3 resulted in a p-value of 0.05, so a significant difference. Based on this we can conclude that users with advanced cooking skills typically rate their recommendations higher than novice or intermediate cooks. The difference is about half a star. The reason for this might be that these users are more curious and motivated to try new recipes.

Table 2 Mean rating for the personalized recommendations per profile

Cooking Profile	N	Mean rec. rating
1=Novice	16	6.48
2=Intermediate	26	6.53
3=Advanced	12	7.04

4.2.3 Qualitative user feedback

The users who did the experiment were asked to share their experiences with the cooking assistance app and the recipe recommendations. The users' feedback provided us the following interesting insights. Recipe recommendations should correspond to meal planning habits. Meal planning occurs either at the end of a working day or in the beginning of the week. Users should receive there recommendations than.

Recipe recommendations are especially useful in case of no inspiration. But users might want to filter these recommendations based on criteria such as costs.

Users believe that the difficulty level of the recommended recipes should match the experience of the cook (e.g., novice, expert). Through an additional filtering of the recommendations this can be achieved. Consequently, ratings of dishes should not only be related to flavor, but also to the process (was it indeed an appropriate dish for the cook type?). This can be achieved by asking users to give multiple ratings per recipe, one for each of the different criteria.

Moreover, users said that the rating should be asked a couple of hours after the cooking moment, when users can better evaluate the different aspects of the recipe. This can be achieved by a notification of the mobile app, asking for this rating.

Users like the possibility to combine individual profiles. But, they also request the possibility to create 'collective' profiles, e.g., a family profile, or a guest profile, for occasional visits.

5 Conclusion

In this article, groups recommendation algorithms for recipes were benchmarked using an innovative combination of offline evaluation with datasets and online evaluation with users utilizing a mobile app. We evaluated 13 algorithms and 2 strategies to generate group recommendations for recipes. An offline test showed that the Implicit ALS algorithm of the Lenskit framework generates the most accurate recommendations. For group recommendations, the strategy that aggregates the recommendations of the group members outperforms the strategy that creates a group profile from the ratings of the group members. For enabling an online test, we developed a cooking app that assists users in the complete cooking process, ranging from selecting the people for whom cooking will be done, and receiving recipe recommendations, to step-by-step instructions for preparing the meal. This online test with 54 participants showed that for new users, who suffer from the cold start problem,

the noticeable differences between personalized and unpersonalized recommendations are rather limited. The personalized recommendations of the Implicit ALS algorithm scored better than (11/15) or equal to (1/15) the unpersonalized list with popular items, in terms of the questions regarding the quality attributes. But because of the limited number of ratings of these cold-start users, these differences were small and not significant. Therefore, we concluded that 10 ratings per user are not enough for accurate recommendations with state-of-the-art algorithms in the context of a recipe recommender. Analyzing the cooking skills of the users, showed that advanced cooks typically rate the recommendations higher than novice cooks. These results could be used by designers of cooking apps as a guideline to decide what algorithm to use, how to enable group recommendations, and as an advice to make a distinction between users based on their cooking competences. To cope with the cold start problem, cooking apps should ask their users to rate at least 10 recipes during the onboarding phase; but our offline evaluation showed that state-of-the-art recommendation algorithms have a potentially higher accuracy in case more ratings per user are available. In future work, it would be interesting to investigate more in detail how many ratings are needed to overcome the cold-start problem and exploit the full potential of recommender algorithms for recipes.

Acknowledgments. This study was partially based on the results of the PhD dissertation of the first author, Toon De Pessemier, more specifically on the fourth chapter that handles group recommendations De Pessemier (2013). The work was executed within the imec.icon project IoT Chef, a research project bringing together academic researchers and industry partners. The IoT Chef project was co-financed by imec and received project support from Flanders Innovation & Entrepreneurship.

Declarations

- Funding: This work was co-financed by imec and received financial support from Flanders Innovation & Entrepreneurship. Grand number HBC.2017.0625
- Conflict of interest: The authors declare that there are no conflicts of interest regarding the publication of this article. The authors have no relevant financial or non-financial interests to disclose. The authors have no competing interests to declare that are relevant to the content of this article.
- Ethics approval: All human participants involved in the study gave their informed consent.
- Consent to participate: When participants arrived in the lab they were asked to fill out an informed consent.
- Consent for publication: All partners involved in this project gave their consent for publication of this article.
- Availability of data and materials: The data that support the findings of this study are available from FoodPairing.com but restrictions apply to the

availability of these data, which were used under licence for the current study, and so are not publicly available.
- Code availability: The authors opted to not disclose the source code because of IP restrictions.
- Authors' contributions: All authors contributed to the study conception and design. The first draft of the manuscript was written by Toon De Pessemier and all authors commented on previous versions of the manuscript. All authors read and approved the final manuscript.

References

Abbas F, Najjar N, Wilson D (2021) Increasing diversity through dynamic critique in conversational recipe recommendations. In: Proceedings of the 13th International Workshop on Multimedia for Cooking and Eating Activities, pp 9–16

Abul-Fottouh D, Song MY, Gruzd A (2020) Examining algorithmic biases in youtube's recommendations of vaccine videos. International Journal of Medical Informatics 140:104,175. https://doi.org/https://doi.org/10.1016/j.ijmedinf.2020.104175, URL https://www.sciencedirect.com/science/article/pii/S1386505619308743

Baltrunas L, Makcinskas T, Ricci F (2010) Group recommendations with rank aggregation and collaborative filtering. In: Proceedings of the fourth ACM conference on Recommender systems. ACM, New York, NY, USA, RecSys '10, pp 119–126, https://doi.org/http://doi.acm.org/10.1145/1864708.1864733

Berkovsky S, Freyne J (2010) Group-based recipe recommendations: analysis of data aggregation strategies. In: Proceedings of the fourth ACM conference on Recommender systems. ACM, New York, NY, USA, RecSys '10, pp 111–118, https://doi.org/http://doi.acm.org/10.1145/1864708.1864732, URL http://doi.acm.org/10.1145/1864708.1864732

Bertin-Mahieux T, Ellis DP, Whitman B, et al (2011) The million song dataset. In: Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)

Bertin-Mahieux T, Ellis DP, Whitman B, et al (2023) Million song dataset. http://millionsongdataset.com/, [Online; accessed 01/07/2023]

Bertram N, Dunkel J, Hermoso R (2023) I am all ears: Using open data and knowledge graph embeddings for music recommendations. Expert Systems with Applications 229:120,347. https://doi.org/https://doi.org/10.1016/j.eswa.2023.120347, URL https://www.sciencedirect.com/science/article/pii/S0957417423008497

Biswal A, Borah MD, Hussain Z (2021) Chapter eleven - music recommender system using restricted boltzmann machine with implicit feedback. In: Kim S, Deka GC (eds) Hardware Accelerator Systems for Artificial Intelligence and Machine Learning, Advances in Computers, vol 122. Elsevier, p 367–402, https://doi.org/https://doi.org/10.1016/bs.adcom.2021.01.001, URL https://www.sciencedirect.com/science/article/pii/S0065245821000139

Cao M, Wu J, Chiclana F, et al (2020) A personalized consensus feedback mechanism based on maximum harmony degree. IEEE Transactions on Systems, Man, and Cybernetics: Systems 51(10):6134–6146

Chen YL, Cheng LC, Chuang CN (2008) A group recommendation system with consideration of interactions among group members. Expert Systems with Applications 34(3):2082–2090. https://doi.org/10.1016/j.eswa.2007.02.008, URL http://www.sciencedirect.com/science/article/pii/S0957417407000863

Choudhury SS, Mohanty SN, Jagadev AK (2021) Multimodal trust based recommender system with machine learning approaches for movie recommendation. International Journal of Information Technology 13:475–482

De Pessemier T (2013) Improved online services by personalized recommendations and optimal quality of experience parameters. PhD thesis

De Pessemier T, Dooms S, Martens L (2014a) Comparison of group recommendation algorithms. Multimedia tools and applications 72(3):2497–2541

De Pessemier T, Dooms S, Martens L (2014b) Context-aware recommendations through context and activity recognition in a mobile environment. Multimedia Tools and Applications 72(3):2925–2948

Dong M, Zeng X, Koehl L, et al (2020) An interactive knowledge-based recommender system for fashion product design in the big data environment. Information Sciences 540:469–488. https://doi.org/https://doi.org/10.1016/j.ins.2020.05.094, URL https://www.sciencedirect.com/science/article/pii/S0020025520304278

Ekstrand M (2023a) Biased matrix factorization with alternating least squares, lenskit 0.13.1 documentation. https://lkpy.readthedocs.io/en/0.13.1/mf.html#module-lenskit.algorithms.als, [Online; accessed 01/07/2023]

Ekstrand M (2023b) Lenskit 0.13.1 documentation. https://lkpy.readthedocs.io/en/0.13.1/index.html, [Online; accessed 01/07/2023]

Frederickson B (2023) Fast python collaborative filtering for implicit datasets. https://github.com/benfred/implicit, [Online; accessed 01/07/2023]

Freyne J, Berkovsky S (2013) Evaluating recommender systems for supportive technologies. User Modeling and Adaptation for Daily Routines: Providing Assistance to People with Special Needs pp 195–217

Funk S (2006) Netflix update: Try this at home. https://sifter.org/~simon/journal/20061211.html, [Online; accessed 01/07/2023]

Giabelli A, Malandri L, Mercorio F, et al (2021) Skills2job: A recommender system that encodes job offer embeddings on graph databases. Applied Soft Computing 101:107,049. https://doi.org/https://doi.org/10.1016/j.asoc.2020.107049, URL https://www.sciencedirect.com/science/article/pii/S156849462030987X

Goldberg D, Nichols D, Oki BM, et al (1992) Using collaborative filtering to weave an information tapestry. Communications of the ACM 35(12):61–70

Grouplens (2023) Movielens. https://grouplens.org/datasets/movielens/, [Online; accessed 01/07/2023]

Holzinger A, Dorner S, Födinger M, et al (2010) Chances of increasing youth health awareness through mobile wellness applications. In: Leitner G, Hitz M, Holzinger A (eds) HCI in Work and Learning, Life and Leisure. Springer Berlin Heidelberg, Berlin, Heidelberg, pp 71–81

Hu Y, Koren Y, Volinsky C (2008) Collaborative filtering for implicit feedback datasets. In: 2008 Eighth IEEE International Conference on Data Mining, Ieee, pp 263–272

Hug N (2023) Surprise. http://surpriselib.com/, [Online; accessed 01/07/2023]

Jia N, Chen J, Wang R (2022) An attention-based convolutional neural network for recipe recommendation. Expert Systems with Applications 201:116,979. https://doi.org/https://doi.org/10.1016/j.eswa.2022.116979, URL https://www.sciencedirect.com/science/article/pii/S0957417422004043

Liao G, Yang L, Mao M, et al (2021) Jam: Joint attention model for next event recommendation in event-based social networks. Knowledge-Based Systems 234:107,592. https://doi.org/https://doi.org/10.1016/j.knosys.2021.107592, URL https://www.sciencedirect.com/science/article/pii/S0950705121008546

Linden G, Smith B, York J (2003) Amazon. com recommendations: Item-to-item collaborative filtering. IEEE Internet computing 7(1):76–80

Ma J, Bian K, Wen J, et al (2023) Srdpr: Social relation-driven dynamic network for personalized micro-video recommendation. Expert Systems

with Applications 226:120,157. https://doi.org/https://doi.org/10.1016/j.eswa.2023.120157, URL https://www.sciencedirect.com/science/article/pii/S0957417423006590

Masthoff J (2004) Group modeling: Selecting a sequence of television items to suit a group of viewers. User Modeling and User-Adapted Interaction 14:37–85. URL http://dx.doi.org/10.1023/B:USER.0000010138.79319.fd

McCarthy JF, Anagnost TD (1998) Musicfx: an arbiter of group preferences for computer supported collaborative workouts. In: Proceedings of the 1998 ACM conference on Computer Supported Cooperative Work. ACM, New York, NY, USA, CSCW '98, pp 363–372, https://doi.org/http://doi.acm.org/10.1145/289444.289511

Natarajan S, Vairavasundaram S, Natarajan S, et al (2020) Resolving data sparsity and cold start problem in collaborative filtering recommender system using linked open data. Expert Systems with Applications 149:113,248

O'Connor M, Cosley D, Konstan JA, et al (2001) Polylens: a recommender system for groups of users. In: Proceedings of the seventh European Conference on Computer Supported Cooperative Work. Kluwer Academic Publishers, Norwell, MA, USA, ECSCW'01, pp 199–218, URL http://dl.acm.org/citation.cfm?id=1241867.1241878

Panchal P, Prajapati DJ (2023) The social hashtag recommendation for image and video using deep learning approach. In: Sentiment Analysis and Deep Learning: Proceedings of ICSADL 2022. Springer, p 241–261

Pecune F, Callebert L, Marsella S (2020) A socially-aware conversational recommender system for personalized recipe recommendations. In: Proceedings of the 8th International Conference on Human-Agent Interaction, pp 78–86

Pu P, Chen L, Hu R (2011) A user-centric evaluation framework for recommender systems. In: Proceedings of the fifth ACM conference on Recommender systems, pp 157–164

Quijano-Sanchez L, Recio-Garcia JA, Diaz-Agudo B (2010) Personality and social trust in group recommendations. In: Proceedings of the 2010 22nd IEEE International Conference on Tools with Artificial Intelligence - Volume 02. IEEE Computer Society, Washington, DC, USA, ICTAI '10, pp 121–126, https://doi.org/10.1109/ICTAI.2010.92, URL http://dx.doi.org/10.1109/ICTAI.2010.92

Quijano-Sanchez L, Sauer C, Recio-Garcia JA, et al (2017) Make it personal: A social explanation system applied to group recommendations. Expert Systems with Applications 76:36–48. https://doi.org/https://doi.org/10.1016/j.eswa.2017.01.045, URL https://www.sciencedirect.com/

science/article/pii/S095741741730060X

Rendle S, Freudenthaler C, Gantner Z, et al (2009) Bpr: Bayesian personalized ranking from implicit feedback. In: Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, pp 452–461

Ricci F, Rokach L, Shapira B (2011) Introduction to recommender systems handbook. In: Recommender systems handbook. Springer, Boston, MA,, p 1–35

Scikit Learn (2023) Machine learning in python. https://scikit-learn.org/, [Online; accessed 01/07/2023]

Takács G, Pilászy I, Tikk D (2011) Applications of the conjugate gradient method for implicit feedback collaborative filtering. In: Proceedings of the fifth ACM conference on Recommender systems, pp 297–300

Tensorflow (2023) An end-to-end open source machine learning platform. https://www.tensorflow.org/, [Online; accessed 01/07/2023]

Tran TNT, Felfernig A, Trattner C, et al (2021) Recommender systems in the healthcare domain: state-of-the-art and research issues. Journal of Intelligent Information Systems 57:171–201

Wang S, Wu J, Chiclana F, et al (2022) Two-stage feedback mechanism with different power structures for consensus in large-scale group decision making. IEEE Transactions on Fuzzy Systems 30(10):4177–4189

Yu Z, Zhou X, Hao Y, et al (2006) Tv program recommendation for multiple viewers based on user profile merging. User Modeling and User-Adapted Interaction 16:63–82. https://doi.org/10.1007/s11257-006-9005-6, URL http://dl.acm.org/citation.cfm?id=1146521.1146531

Zhou Y, Wilkinson D, Schreiber R, et al (2008) Large-scale parallel collaborative filtering for the netflix prize. In: International conference on algorithmic applications in management, Springer, pp 337–348