Using Tags and Latent Factors in a Food Recommender System

Mouzhi Ge
Free University of Bozen-Bolzano
Bolzano, Italy
mouzhi.ge@unibz.it

Mehdi Elahi
Free University of Bozen-Bolzano
Bolzano, Italy
mehdi.elahi@unibz.it

Ignacio Fernández-Tobías
Universidad Autónoma de Madrid
Madrid, Spain
ignacio.fernandezt@uam.es

Francesco Ricci
Free University of Bozen-Bolzano
Bolzano, Italy
francesco.ricci@unibz.it

David Massimo
Free University of Bozen-Bolzano
Bolzano, Italy
david.massimo@stud-inf.unibz.it

ABSTRACT

Due to the extensive growth of food varieties, making better and healthier food choices becomes more and more complex. Most of the current food suggestion applications offer just generic advices that are not tailored to the user's personal taste. To tackle this issue, we propose in this paper a novel food recommender system that provides high quality and personalized recipe suggestions. These recommendations are generated by leveraging a data set of users' preferences expressed in the form of users' ratings and tags, which signal the food's ingredients or features that the users like. Our empirical evaluation shows that the proposed recommendation technique significantly outperforms state-of-the-art algorithms. We have found that using tags in food recommendation algorithms can significantly increase the prediction accuracy, i.e., the match of the predicted preferences with the true user's preferred recipes. Furthermore, our user study shows that our system prototype is of high usability.

Categories and Subject Descriptors

H.4.2 [**Information Systems Applications**]: Types of Systems—*Decision support*; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*information filtering*

Keywords

Recommender systems, smart health, food, matrix factorization, tags

DH'15, May 18–20, 2015, Florence, Italy.

http://dx.doi.org/10.1145/2750511.2750528.

1. INTRODUCTION

With nowadays' increasing changes in the food sector and lifestyle, many people are facing the problem of making better and healthier food choices [10]. A typical daily activity is to prepare and cook a meal. In order to decide what to cook, people tend to select recipes that satisfy their personal taste. However, because of the extensive growth of food varieties, making food choices by manually exploring a full recipe catalog can be time-consuming and frustrating. We believe that there is a need for decision-aid systems that can suggest and recommend personalized food options while taking into account the user's preferences and eating history.

Recommender systems (RSs) are information search and filtering tools that help users to make better choices while searching for products or services such as movies, vacations, or electronic products [21, 15]. The fundamental goal of a RS is to reduce the information overload and to provide personalized suggestions that can assist users' decision making. RSs are playing an important role throughout the Internet. They have been applied in a large number of Internet applications such as Amazon, YouTube, Netflix, Yahoo, Tripadvisor, Last.fm, and IMDB [13]. Moreover, social networks such as LinkedIn and Facebook have also introduced recommendation technologies to suggest groups to join, people to follow or posts to like [1].

Among the various recommender systems application domains that have been explored in last years, food is an emerging one and it is attracting a considerable research focus. The problem is that food, for many people, has become a black box and is associated with bad eating habits and poor healthy conditions. For that reason, some food recommender systems have been proposed.

However, most of the existing applications provide just generic food advices that are not tailored to user's specific tastes or poorly match them. One of the best recent food recommender system is described in [11]. This is a meal planner system that provides personalized recommendations using a content-based recommendation technology. We have conjectured that the accuracy of the system predicted user's preferences (ratings) can be improved by better modelling

and acquiring user preferences, i.e., by letting the users, with tags, to signal what are in their opinion, the most important ingredients and features of the recipes. We also focussed on the overall system usability, and tried to improve it by designing an effective human-computer interaction.

This paper therefore aims at filling a research gap by proposing a mobile food recommender system that is easy to use and offers high-quality personalized food recommendations. To this end, we summarize our main contributions as follows.

- We show how to incorporate user selected tags, which describe important attributes of food, in a recommender algorithm based on matrix factorization, i.e., which uses latent factors for modelling both users and recipes.
- We describe how to exploit this algorithm, which is able to predict the rating that a user will give to a not yet rated item, by designing a complete human-computer interaction in a tablet-based application.
- We show that the proposed solution offers personalized food recommendations with high prediction accuracy, i.e., the recommended food recipes receive high evaluations from the users. In fact, our proposed algorithm significantly outperforms state-of-the-art algorithms, such as that described in [11], in terms of rating prediction error (MAE and RMSE).
- We show that in a food recommender system, managing user assigned tags can become an important success factor that can lead to a significant prediction accuracy improvement.

We have developed an Android-based food recommender system that is focused on supporting a user who wants to choose what to cook at home. This system can be applied to choose a recipe that the user is not necessarily cooking by herself. We believe that the developed system will have an important social and economic impact. In fact, more and more users would like to interact with food applications on mobile or embedded devices. For example, it is more convenient to read the cooking instructions from a tablet, which can be easily integrated into many cooking appliances, rather than using a traditional computer. Our application is implemented in a mobile setting, it supports a novel interaction design, and can be further developed and extended to cover a full range of user requirements related to food management and cooking. In our previous work [5], we have introduced the research goals and the general features of our system. In this paper, we describe the implemented rating prediction algorithm in detail and we report the results of an offline and online evaluation of our system.

The remaining of the paper is organized as follows. The next section reviews recent research works that are related to recommender system and existing food applications. Afterwards, in Section 3 we describe our novel system HCI and system prototype. In section 4 we describe our proposed algorithm in detail. In section 5.2 we describe the offline comparison of the recommendation algorithm with other competing algorithms and the analysis of the full system prototype in a live user study. Finally section 6 concludes the paper by discussing the evaluation results and outlining future work.

2. RELATED WORK

Recommender systems tackle the problem of generating personalized suggestions for digital content, products or services, that suit the user's needs and constraints better than the popular or the mainstream products [3]. In every day life, getting recommendations from others by words, reference letters, media reports, or travel guides, are common practices. Recommender systems may enhance this process by helping people to explore or search for available items, such as, books, articles, webpages, movies, music, or even jokes. Recommender systems suggest to users items that are judged to be desirable based on the users' preferences. They turn out to be essential applications in e-commerce and information retrieval, helping users to find those items that are most suitable for their needs and tastes by reducing large space of options. Such a task can be very difficult when there are thousands of items available to people for watching, reading, visiting, traveling, or buying.

The core computation of a recommender system is the prediction of the user evaluation, typically in the form of ratings, for the items that the user has not evaluated yet. These predictions are computed by using machine learning algorithms that leverage a data set of already collected users' evaluations for a sample of the domain items. When a target user approaches a RS, for all the items in the catalogue the system predicts the ratings that the user is expected to assign to the items and recommends the items that have the largest predicted ratings [21].

The most popular recommendation techniques are Content-Based, Collaborative Filtering, and Hybrid [22]. Systems using Content-Based techniques suggest items of interest based on their associated features [18]. For instance, a content-based news recommender considers the terms appearing in the news articles as distinctive features and recommend news articles that have the features possessed by the articles that the user preferred before. Systems adopting Collaborative Filtering techniques predict a target user's ratings by only using the ratings that the various users of the system gave, and then recommends the items with the highest predicted ratings [16, 4]. Hybrid RSs combine two or more techniques, e.g., collaborative and content-based, in order to cope with the specific limitations of a single technique.

Focusing on collaborative filtering, nowadays, the most popular rating prediction technique is matrix factorization [16]. Fundamentally, in matrix factorization a given matrix of numerical ratings (e.g., between 1 to 5) of users for items $\{r_{ui}\}_{m \times n}$ (m users and n items) is decomposed in two lower dimensional matrices, so that a rating prediction for all the unknown entries in the original matrix $\{r_{ui}\}$ can be computed [16]. In matrix factorization both users and items are modelled with vectors of abstract factors that are learned by mining the available ratings. For this reason matrix factorization algorithms are also called "factor models." In a factor model, because users and items share a common representation, it is possible to establish the similarity of users and items. These similarities and the relationships between users and items are exploited for predicting missing ratings and generating recommendations.

Matrix factorization originated in the context of the Netflix prize[1] around 2007, and have since then attracted a lot of attention from the research community as well as the indus-

[1] http://www.netflixprize.com/

try. As a result, numerous variants have been developed. Some of them exploit additional data in order to improve the prediction accuracy achieved by only relying on users' ratings. One of the most popular variant is SVD++ [16], which considers user provided implicit feedback such as the browsing history of the user, or the user's purchased items. The use of implicit feedback proved to be valuable for the prediction algorithm, and was eventually part of the winning solution to the Netflix competition.

Along this line, recent works have also successfully integrated explicit content metadata into matrix factorization. For instance, Enrich et al. [8] modified the SVD++ algorithm to introduce tagging information instead of implicit feedback. They achieved higher accuracy in the *cold-start*, which is a situation that arises when there is not enough user preference data to provide the user with accurate suggestions. Manzato [19] proposed gSVD++, an extension of SVD++ in which, besides the user's implicit feedback, the item model is also enhanced with attribute factors. Similarly to [8], Fernández-Tobías and Cantador [9] adapted gSVD++ to separately integrate user's and item's tags into the factorization model, further improving the performance of the system in the cross-domain recommendation setting. In this work, we build on the algorithm presented in [9], based on the hypothesis that the tags assigned by the users to recipes can provide additional useful information about their preferences and therefore, if used in the rating prediction algorithm, can lead to a better system performance.

Food is an important and emerging application domain for RSs. Here the elements of interest are the food ingredients, which can be combined in recipes [10] and menus [17], or the nutritional concerns [12, 26]. In general, ingredient information are of primary importance and must strongly influence the recommendation output. Food RSs can make personalized recommendations for food items such as recipes, diet plans, meals or ingredients [25, 10]. In this paper, we focus on a food RS that suggests recipes that the user can cook. It exploits explicit food preferences, expressed in the form of ratings and tags regarding what user has eaten or cooked. to train the recommendation model [10].

In this domain there are some important elements to be considered when generating recommendation such as: the combination or sequencing of food recommendations, the ways of preparing [10], as well as the nutritional utility of a particular food for a user or a group. This makes it almost impossible to build a recommender that can consider all these important factors, hence a focus has to be defined, by only considering very few significant aspects of the overall problem [11].

In this paper, we focus on a recommender system that can identify recipes that the users will like. So far, food recommender systems have been developed using a variety of recommendation techniques [23, 24, 10]. Freyne and Berkovsky [11] proposed a content-based recommender (CBFB), which suggests to the user recipes based on user's preferred recipes. In this system first recipe preferences are converted into ingredient preferences: as one recipe is typically constructed by a set of ingredients, a user's rating for a recipe can be transferred to a rating for the ingredients in this recipe. If one ingredient appears in different recipes, an average of the user's assigned recipes' ratings can be estimated to be the rating of the user for that ingredient. Thus each ingredient can be associated with a score that is obtained from the user's ratings for the recipes. In order to predict the rating of a certain recipe, the scores of the ingredients in this recipe are averaged while considering all the ingredients to be equally important. We considered this technique as a baseline solutions, and therefore we compared it with our recommender in the evaluation section.

3. SYSTEM INTERACTION AND PROTOTYPE

In this section, we will first describe the main steps of the human-computer interaction that is supported by our proposed recommender system and then will illustrate the details of the implemented system prototype.

The supported human-computer interaction is a fundamental aspect of a recommender system. We have designed a complete interaction process for collecting user preferences, in the form of ratings and tags, and presenting the recommendations to the user [5]. The user-system interaction includes the following steps:

1. General Preference Elicitation: where the user rates and tags recipes.
2. Session-Related Preference Elicitation: where the user defines the important ingredients that she will like to use, and therefore should be included in the recommended recipe.
3. Recommendation: where the user is presented with recommendations and can browse them.

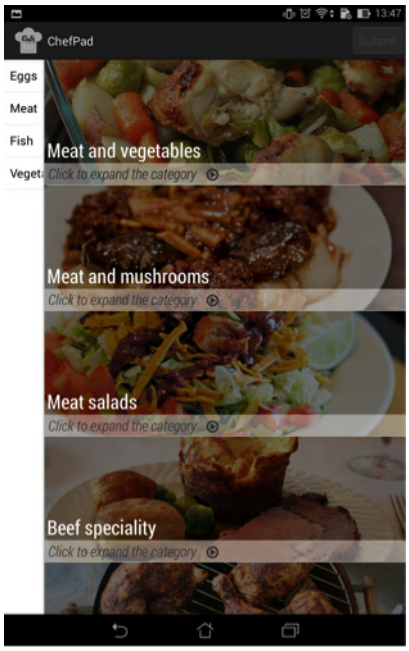

Figure 1: Screen for locating the recipes that the user cooks

We have then developed an Android-based food recommender system prototype that supports our interaction design. The system is focused on supporting a user that want to choose what to cook at home. We focus here on this scenario even though the system can also be applied to choose

a recipe that the user is not necessarily cooking by herself. The first step is general preference elicitation. The goal of this step is to collect the long term (stable) preferences of the users [20], i.e., what she generally likes to cook (or eat). This step includes two stages: (1) where the system asks the user to specify the recipes she cooks at home and (2) where the user gives her ratings and tags to the recipes she cooked.

After the user logs in the system, the user can browse the full catalog of recipes and mark those that she has experience of. This step is required to quickly identify the recipes that the user can actually evaluate. Different categories of recipes such as fish or salads are presented to the user (Figure 1). Users can easily navigate through these categories to find a detailed recipe, for example, Beef -> Roasted Beef -> Roasted Beef with Salad.

Inside each category, there is a list of recipes related to this category. The system will then ask the user to specify which recipes she has experienced. The user can mark her familiar recipes as "Eaten" or "Cooked" by clicking the option box (Figure 2). "Eaten" and "Cooked" options are currently treated by the system in the same way. In the future we plan to exploit the information that the user only eat some dish without cooking it.

Figure 2: Screen for marking the recipes that user experienced

After the user has marked the recipes she has experienced, some recipes are presented to the user for rating and tagging. In fact, for the system to acquire some knowledge of the user preferences the user is asked to rate and tag recipes selected among those that she has declared to have eaten or cooked before. But, the system needs also to explore a bit more the user's preferences and it presents additional recipes for the user to rate. These additional recipes are found by guessing what the user may have experienced but has not yet marked in the first step. In order to find such recipes active learning techniques, i.e., and in particular, the Binary Prediction technique is used [6, 7].

In Figure 3 it is shown the interface where the user is asked to rate and tag the system identified recipes. The rating interface uses a classical 5-star Likert scale (Figure 3). In addition, the users is requested to "explain" the core motivations for an assigned rating by adding to the recipe some tags. In the supported interaction the system gives two tagging possibilities to the user: (a) adding some of the ingredients as tags, (b) adding any other tag that the user deems as relevant to the recipe. Users can decide to tag the recipe with some of the suggested tags, or add their own tags. A list of tags (normally 5-12 tags) will be suggested to the users when they are tagging the recipe. Tags are suggested when they are evaluated by the system as relevant to the recipe. For example, the system can suggest ingredients of the recipe, or key words from the recipe title, or some popular tags like *spicy* or *Asian style* that were assigned by other users to this recipe.

When the user is tagging a recipe she must also indicate whether there is a positive or negative attitude with the selected tags. For example, in Figure 3 the tag *olive oil* is coloured in green because the user believes that this is a positive aspect of the recipe, while *garlic* is coloured in red because she does not like *garlic* in this recipe. In principle the rating and tagging only need to be done once when the user uses the system for the first time. But the user can always access this section to rate additional recipes. We believe that all these different types of tags should be used to build the rating prediction model because they bring precise information about the user food preferences that may more difficult to acquire exploiting only the ratings.

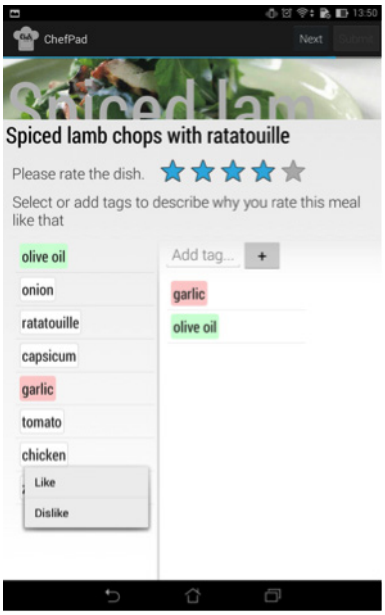

Figure 3: Rate and tag interface

In the session-related preference elicitation step, the user is supposed to be looking for a recipe recommendation and can therefore enter the ingredient (e.g., tuna fish) or feature of the recipe that she wants to cook. This can be done by selecting a tag from the list of suggested tags (Figure 4).

Finally, in the recommendation step, by exploiting the collected user preferences (long term and session specific),

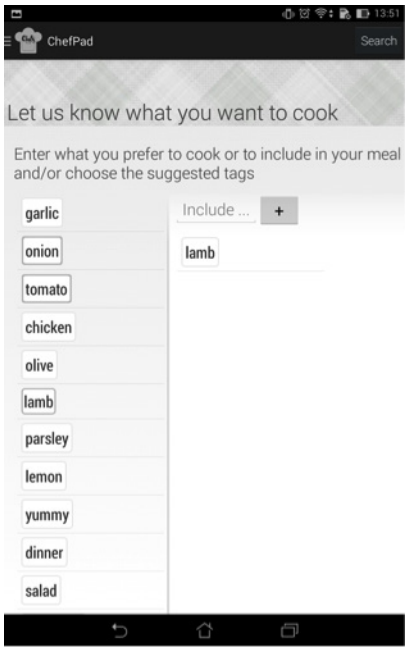

Figure 4: Screen for choosing the distinguished feature of the recipe that the user wants to cook

Figure 5: Recipe ingredients and cooking instructions

the system generates recommendations that are labeled by the tag selected in the previous step. The system presents, one by one, a set of recipe recommendations to the user and asks the user to choose one. The user is supposed to browse the first recommendation and eventually to go to the next one if she is not satisfied. When the user chooses one recommended recipe, the system presents its cooking instructions as shown in Figure 5.

4. RECOMMENDATION ALGORITHM

In this section we describe the core algorithm that is used to predict the individual users' ratings for different recipes.

In order to suggest useful recipes to the user, as we have described above, the system first needs to collect information about her preferences. Either during the general preference elicitation step, or after a particular recipe is tried, the user has the option to tell the system if the recipe fits her tastes. In our system, these user's preferences are collected and represented as *ratings*, i.e., numerical evaluations ranging from 1 to 5, and the goal of the recommendation algorithm is to predict the ratings for recipes that have not been rated yet and the user may like to cook. The recipes with the highest predicted ratings are more likely to be relevant and are therefore suggested to the user. The recommended recipes can be those that have never been tried by the user or those that are worth to be cooked again.

In this work, we have implemented an algorithm that is capable of exploiting tagging information in the rating prediction process [9]. This algorithm builds upon the popular matrix factorization method [16], in which each user u is associated with a parameters' vector $\mathbf{p}_u \in \mathbb{R}^k$ that models her latent features. Likewise, each recipe m is modeled by a vector $\mathbf{q}_m \in \mathbb{R}^k$ that contains its latent features. In matrix factorization, ratings are estimated by computing the dot product of the vectors: $\hat{r}_{um} = \mathbf{p}_u^T \mathbf{q}_m$.

The algorithm proposed in [9] extends matrix factorization by including additional parameters used for modelling the dependencies between assigned tags and ratings. As we have illustrated previously, in addition to ratings, in our system users can also specify their preferences by means of tags, and the tags' valences (positive or negative), assigned to the recipes. For instance, a user may use the tag *spicy* to tell the system she prefers that type of recipes. Similarly, an item can be described with the tag *tomato* if this is one of its ingredients.

We conjectured that tags assigned to an item are correlated to the ratings, and can be used to better predict user's interests. Thus, we have decided to adopt the rating prediction model described in [9] and have introduced additional feature vectors $\mathbf{x}_t \in \mathbb{R}^k$ and $\mathbf{y}_s \in \mathbb{R}^k$ for user's and recipe's tags, respectively. Ratings are now estimated as follows:

$$\hat{r}_{um} = \left(\mathbf{p}_u + \frac{1}{|T_u|}\sum_{t \in T_u} \mathbf{x}_t\right)^T \left(\mathbf{q}_m + \frac{1}{|T_m|}\sum_{s \in T_m} \mathbf{y}_s\right) \quad (1)$$

where, T_u is the set of tags assigned by the user u to *any* recipe, and T_m is the set of tags assigned to the recipe m by *any* user. If $T_u = \emptyset$ or $T_m = \emptyset$ then we ignore the corresponding terms and the model reduces to standard matrix factorization.

We note that by introducing latent feature vectors for users and recipes separately, the algorithm is able to compute predictions even if the user did not assign any tag to the recipe, hence the usage of tags does not impose that a certain number of tags are actually acquired.

Moreover, when incorporating the tags into the model, in the current version of the system, we have used only the positive tags from the user. This makes easier to isolate the effect of the tags on the items' ratings. In a future work we will identify a viable technique for exploiting also the negative tags.

We show in the following the minimisation problem that must to be solved in order to determine the model's parameters:

$$\min_{p_*,q_*,x_*,y_*} \sum_{(u,m)\in\mathcal{R}} (r_{um} - \hat{r}_{um})^2 + \lambda \left(\|\mathbf{p}_u\|^2 + \|\mathbf{q}_m\|^2 + \sum_{t\in T_u} \|\mathbf{x}_t\|^2 + \sum_{s\in T_m} \|\mathbf{y}_s\|^2 \right) \quad (2)$$

where $\mathcal{R}$ is the training set, and the parameter $\lambda \in \mathbb{R}$ controls the complexity of the model; it is used to prevent overfitting. The parameters $\mathbf{p}_u, \mathbf{q}_m, \mathbf{x}_t, \mathbf{y}_s$ are automatically learned during the training phase of the algorithm, by minimizing the regularized squared prediction error. In our experiments, we found $\lambda = 0.02$ to be a good trade-off. The previous minimization problem can be efficiently solved using stochastic gradient descent, as depicted in Algorithm 1.

Algorithm 1 Recommendation model training step

procedure TRAINSTEP
  for $(u, m) \in \mathcal{R}$ **do**
    $e_{um} \leftarrow r_{um} - \hat{r}_{um}$ ▷ Compute $\hat{r}_{um}$ using eq. 1
    ▷ Simultaneously update the parameter vectors:
    $\mathbf{p}_u \leftarrow \mathbf{p}_u - \alpha \left[\lambda \mathbf{p}_u - e_{um} \left(\mathbf{q}_m + |T_m|^{-1} \sum_{s\in T_m} \mathbf{y}_s\right)\right]$
    $\mathbf{q}_m \leftarrow \mathbf{q}_m - \alpha \left[\lambda \mathbf{q}_m - e_{um} \left(\mathbf{p}_u + |T_u|^{-1} \sum_{t\in T_u} \mathbf{x}_t\right)\right]$
    for all $t \in T_u$ **do**
      $\mathbf{x}_t \leftarrow \mathbf{x}_t - \alpha \left[\lambda \mathbf{x}_t - \frac{e_{um}}{|T_u|} \left(\mathbf{q}_m + |T_m|^{-1} \sum_{s\in T_m} \mathbf{y}_s\right)\right]$
    end for
    for all $s \in T_m$ **do**
      $\mathbf{y}_s \leftarrow \mathbf{y}_s - \alpha \left[\lambda \mathbf{y}_s - \frac{e_{um}}{|T_m|} \left(\mathbf{p}_u + |T_u|^{-1} \sum_{t\in T_u} \mathbf{x}_t\right)\right]$
    end for
  end for
end procedure

A single training step iterates over the whole training set updating the model parameters each time. This procedure is typically repeated for a fixed number of iterations, or until convergence is reached. The parameter α controls to what extent the parameters are updated on each step: a too small value can make the learning process very slow, whereas with large values the algorithm may fail to converge. In our experiments, we found good results using $\alpha = 2 \cdot 10^{-4}$, and a single training iteration. We also found that a value of $k = 5$ for the dimensionality of the latent vectors offered a good trade-off between prediction accuracy and training time.

Once the training phase is complete, we use the learned parameters to compute rating predictions using the equation 1. In order to provide the target user with recommendations, the system first estimates her ratings for the unknown recipes, sorts them from the highest to the lowest predicted value, and presents the top ranked items.

5. EVALUATION

We have asked some users to test our system. The ratings and tags collected from these users were then used to conduct an offline 5-fold cross validation. When using our application, users also evaluated the usability of our developed prototype. The results from the offline evaluation and user study are reported in this section.

A total of 20 subjects have participated to the evaluation experiment. The subjects are composed of 60% male and 40% female in the range of 23-50 years old. The subjects' occupations vary from college students, researchers, company employees as well as nutritionists. They are from a variety of countries such as France, Germany, China, USA, UK, Morocco, Persia and Italy.

Each subject used a tablet to test the system. The system is running in the Android v.4.4.3 operating system. The tablet that we used is a mobile device manufactured by ASUS, model is "Fonepad 7 - ME372CG". Hardware specification of the device are as follows: 7 inch 1280x800 IPS display, Atom Z2560 Dual-Core 1.6 GHz, RAM 1Gb, network standard DC-HSPA, WCDMA, EDGE/GSM and WLAN802.11 a/b/g/n.

5.1 Offline Evaluation

In the offline evaluation, We have conducted a 5-fold cross validation to compare the proposed rating prediction algorithm, which we call MF-T, and the content-based approach (CBFB) presented in [11], which we have briefly described in the related work section. We have used a 5-fold cross validation, which is normally used in RSs research. In this validation procedure, the full rating data set is split into five disjoint subsets of the same size. Then, the system prediction model is trained using four of these five subsets and the predictions for the ratings in the fifth subset are compared with the true ratings found in that subset (test set). This training and test procedure is repeated 5 times, each time with a different test set. And finally the system error measured on the five iterations is averaged in order to produce a final measure.

In total we collected 113 recipes. All of them are real-world recipes listed in Total Wellbeing Diet [11]. We initially collected a total of 1888 tags derived from ingredients and Internet resources. Ingredient tags are simply the names of the ingredients. Other tags were crawled from the Instagram website, where we found tags to images of the considered recipes. Furthermore, we manually discarded tags that bring almost no information, such as *food* or *hmmmm*, asn other normalization steps were applied, such as stemming. As such, we obtained 850 distinct tags associated to the recipes, on average 7.5 tags per recipe. The most popular tags are ingredients such as chicken, salad and beef. From the users who tried the system, we obtained 137 ratings and 176 tags, on average 6.85 ratings and 8.8 tags per user.

Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) were used to evaluate the accuracy of the rating prediction algorithms. They measures the average absolute deviation between a predicted rating and the user's true rating. Lower MAE or RMSE means higher prediction accuracy. Their definitions are as follows:

$$MAE = \frac{1}{|T|} \sum_{(u,i)\in T} |\hat{r}_{ui} - r_{ui}| \quad (3)$$

$$RMSE = \sqrt{\frac{1}{|T|} \sum_{(u,i)\in T} (\hat{r}_{ui} - r_{ui})^2} \quad (4)$$

where T is a test set of user and item pairs. $\hat{r}_{ui}$ is the model predicted rating and r_{ui} is the true user ratings, which we obtained in the user study.

In Table 1, we can see that in each fold of the 5-fold cross validation procedure the MAE of MF-T, our matrix factorization algorithm extended with tags, is lower than the content based approach (CBFB) proposed in [11] ($p<0.01$). Similar results were obtained for RMSE (see Table 2).

Table 1: MAE of the considered rating prediction algorithm

MAE	MF-T	CBFB	Standard MF
Fold 1	0.702	1.559	1.122
Fold 2	0.678	1.503	1.200
Fold 3	0.733	1.393	1.052
Fold 4	0.546	1.464	0.854
Fold 5	0.772	1.025	0.860
Mean	0.686	1.389	1.018

Table 2: RMSE of the considered rating prediction algorithm

RMSE	MF-T	CBFB	Standard MF
Fold 1	0.882	2.241	1.617
Fold 2	0.855	2.052	1.566
Fold 3	0.885	1.957	1.505
Fold 4	0.717	2.001	1.272
Fold 5	0.899	1.624	1.123
Mean	0.848	1.975	1.416

In order to explain the observed good performance of MF-T, we compared it with standard Matrix Factorization, i.e., when no tags information is used in the prediction algorithm. In Table 1 and 2 we can see that the standard Matrix Factorization performs slightly better than CBFB. But we also observe that our extended matrix factorization algorithm with tags outperforms standard Matrix Factorization. This suggests that the good performance of our algorithm is based on the exploitation of the users' preferences brought by the tags data.

In fact, the prediction model can learn that a user tends to prefer items that are tagged with certain positive tags and can extrapolate that other items that are similarly tagged by the user will be also preferred. To illustrate this idea, we refer to one particular user's tagging behavior, that we have found in our data set. This user has added *beef* as a positive tag to one recipe. Then, in two other recipes that contain the beef ingredient the user again marked the recipes with the positive tag *beef*. This clearly signals a tendency of the user to like recipes containing beef.

We also note that our algorithm can be considered an hybrid method, since it uses both ratings and recipe features, in the form of tags. We conjecture that our algorithm has a better performance than the pure content-based model that we have considered because the last method uses indiscriminately all the recipe's features when estimating its ratings. Conversely, our algorithm identifies the tags that are more important for rating prediction, hence performing an implicit feature selection process. In the future work, we will conduct further experiments to determine the most informative content features and we will double check if a CB algorithm that only exploits these most informative features can compete with the tag-based solution proposed here.

5.2 User Study

In order to evaluate the usability of the developed prototype we adopted the System Usability Scale (SUS) from [2]. SUS is a widely accepted measurement for evaluating system usability. It is composed of 10 standard questions. Each question is answered with values in a Likert scale ranging from 1 to 5 (1 means Strongly Disagree and 5 means Strongly Agree). The replies are then converted from the Likert scale into scores ranging between 0 and 4, summed and normalised to 100. As [2] states, a SUS score of 68 can be considered as a benchmark of (average) system usability.

The 20 users, whose demographical information have been described in Section , have also evaluated the system usability by filling the SUS questionnaire. We found that for 70% of the users the system scored a SUS value larger than 68. This indicates that most of the users believe that the system has a large usability. The average SUS score among the 20 subjects is 75.5 (margin of error is 6.01), which is larger than the SUS benchmark score of 68, indicating that the usability of our system is perceived as high.

In addition we measured the perceived recommendation quality of our system. The questionnaire that we used to measure the perceived recommendation quality is adopted from [14], which has 7 questions and a highest score of 4 for each of them. Hence, the maximum value is 28. Calculating the overall score of the perceived recommendation quality for the system, we obtained 19 out of 28. On average the user assigned the "agree" level, which indicates that the users agreed that the recommendations were well-chosen and also suit well their preferences.

6. CONCLUSION

In this paper we have proposed a novel food recommender system that generates recipe recommendations using an extension of Matrix Factorization. The system in addition to ratings collects users' tags. The system was developed for a tablet-based platform, and implements a well designed human-computer interaction.

We have carried out an offline evaluation to compare the proposed recommendation algorithm with a state of the art content-based algorithm proposed in [11]. Our algorithm significantly outperforms the other algorithm. In order to better understand what causes the significant improvement, we further compared our solution with a standard Matrix Factorization algorithm and found that the usage of tags is the important factor that leads to the observed rating prediction improvement. Furthermore, our user study shows that our system prototype has a high utility, the recommendations are well-chosen and of high quality.

As future work, we plan to consider nutritional factors such as calories and proteins to really generate health-aware recommendations. We want to generate recommendations that enable the user to correctly balance the intake of nutritional elements. Also, in order to improve our recommender algorithm, we will incorporate the negative tags into the rating prediction model, we will consider the problem of generating sequentially coherent recommendations, so to avoid that the system offers the same recommendation two days in a row. In addition to only offering our application to individual user, we will develop a new recommendation algorithm and HCI design so that it can be used in the context of a group of users, e.g., a family.

7. REFERENCES

[1] N. Baghaei, S. Kimani, J. Freyne, E. Brindal, S. Berkovsky, and G. Smith. Engaging families in lifestyle changes through social networking. *International Journal of Human-Computer Interaction*, 27(10):971–990, 2011.

[2] J. Brooke. Sus: a quick and dirty usability scale. In Jordan, editor, *Usability Evaluation in Industry*. London: Taylor and Francis, 1996.

[3] R. Burke. Knowledge-based recommender systems. *Encyclopaedia of Library and Information Systems*, 69(32), 2000.

[4] C. Desrosiers and G. Karypis. A comprehensive survey of neighborhood-based recommendation methods. In F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors, *Recommender Systems Handbook*, pages 107–144. Springer, 2011.

[5] M. Elahi, M. Ge, F. Ricci, D. Massimo, and S. Berkovsky. Interactive food recommendation for groups. In *Poster Proceedings of the 8th ACM Conference on Recommender Systems, RecSys 2014, Foster City, Silicon Valley, CA, USA, October 6-10, 2014*. 2014.

[6] M. Elahi, F. Ricci, and N. Rubens. Active learning strategies for rating elicitation in collaborative filtering: A system-wide perspective. *ACM Transactions on Intelligent Systems and Technology*, 5(1):13, 2013.

[7] M. Elahi, F. Ricci, and N. Rubens. Active learning in collaborative filtering recommender systems. In *Proceedings of the 14th International Conference on E-Commerce and Web Technologies*, pages 113–124. Springer, 2014.

[8] M. Enrich, M. Braunhofer, and F. Ricci. Cold-start management with cross-domain collaborative filtering and tags. In *Proceedings of the 13th International Conference on E-Commerce and Web Technologies*, pages 101–112. Springer, 2013.

[9] I. Fernández-Tobías and I. Cantador. Exploiting social tags in matrix factorization models for cross-domain collaborative filtering. In *Proceedings of the 1st Workshop on New Trends in Content-based Recommender Systems, Foster City, California, USA*, pages 34–41, 2014.

[10] J. Freyne and S. Berkovsky. Intelligent food planning: personalized recipe recommendation. In *Proceedings of the 15th International Conference on Intelligent User Interfaces*, pages 321–324. ACM, 2010.

[11] J. Freyne and S. Berkovsky. Evaluating recommender systems for supportive technologies. In *User Modeling and Adaptation for Daily Routines*, pages 195–217. Springer, 2013.

[12] G. Geleijnse, P. Nachtigall, P. van Kaam, and L. Wijgergangs. A personalized recipe advice system to promote healthful choices. In *Proceedings of the 16th International Conference on Intelligent User Interfaces*, pages 437–438. ACM, 2011.

[13] D. Jannach, M. Zanker, A. Felfernig, and G. Friedrich. *Recommender Systems: An Introduction*. Cambridge University Press, 2010.

[14] B. P. Knijnenburg, M. C. Willemsen, Z. Gantner, H. Soncu, and C. Newell. Explaining the user experience of recommender systems. *User Modeling and User-Adapted Interaction*, 22(4-5):441–504, 2012.

[15] J. A. Konstan and J. Riedl. Recommender systems: From algorithms to user experience. *User Modeling and User-Adapted Interaction*, 22(1-2):101–123, Apr. 2012.

[16] Y. Koren and R. Bell. Advances in collaborative filtering. In F. Ricci, L. Rokach, B. Shapira, and P. Kantor, editors, *Recommender Systems Handbook*, pages 145–186. Springer Verlag, 2011.

[17] F.-F. Kuo, C.-T. Li, M.-K. Shan, and S.-Y. Lee. Intelligent menu planning: Recommending set of recipes by ingredients. In *Proceedings of the ACM Multimedia 2012 Workshop on Multimedia for Cooking and Eating Activities*, CEA '12, pages 1–6, New York, NY, USA, 2012. ACM.

[18] P. Lops, M. de Gemmis, and G. Semeraro. Content-based recommender systems: State of the art and trends. In F. Ricci, L. Rokach, B. Shapira, and P. Kantor, editors, *Recommender Systems Handbook*, pages 73–105. Springer, 2011.

[19] M. G. Manzato. gsvd++: supporting implicit feedback on recommender systems with metadata awareness. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing, SAC '13, Coimbra, Portugal*, pages 908–913, 2013.

[20] A. M. Rashid, I. Albert, D. Cosley, S. K. Lam, S. M. Mcnee, J. A. Konstan, and J. Riedl. Getting to know you: Learning new user preferences in recommender systems. In *Proceedings of the 2002 International Conference on Intelligent User Interfaces, IUI 2002*, pages 127–134. ACM Press, 2002.

[21] F. Ricci. Recommender systems: Models and techniques. In R. Alhajj and J. G. Rokne, editors, *Encyclopedia of Social Network Analysis and Mining*, pages 1511–1522. Springer, 2014.

[22] F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor. *Recommender Systems Handbook*. Springer, 2011.

[23] J. Sobecki, E. Babiak, and M. Słanina. Application of hybrid recommendation in web-based cooking assistant. In *Knowledge-Based Intelligent Information and Engineering Systems*, pages 797–804, 2006.

[24] M. Svensson, J. Laaksolahti, K. Höök, and A. Waern. A recipe based on-line food store. In *Proceedings of the 5th International Conference on Intelligent User Interfaces*, pages 260–263. ACM, 2000.

[25] C.-Y. Teng, Y.-R. Lin, and L. A. Adamic. Recipe recommendation using ingredient networks. In *Proceedings of the 3rd Annual ACM Web Science Conference*, pages 298–307. ACM, 2012.

[26] Y. van Pinxteren, G. Geleijnse, and P. Kamsteeg. Deriving a recipe similarity measure for recommending healthful meals. In *Proceedings of the 16th International Conference on Intelligent User Interfaces*, pages 105–114. ACM, 2011.