Optimally Balancing Receiver and Recommended Users' Importance in Reciprocal Recommender Systems

Akiva Kleinerman
Bar-Ilan University
Ramat-Gan, Israel

Ariel Rosenfeld
Weizmann Institute of Science
Rehovot, Israel

Francesco Ricci
Free University of Bozen-Bolzano
Bolzano, Italy

Sarit Kraus
Bar-Ilan University
Ramat-Gan, Israel

ABSTRACT

Online platforms which assist people in finding a suitable partner or match, such as online dating and job recruiting environments, have become increasingly popular in the last decade. Many of these platforms include recommender systems which aim at helping users discover other people who will also be interested in them. These recommender systems benefit from contemplating the interest of both sides of the recommended match, however the question of how to optimally balance the interest and the response of both sides remains open. In this study we present a novel recommendation method for recommending people to people. For each user receiving a recommendation, our method finds the optimal balance of two criteria: a) the likelihood of the user accepting the recommendation; and b) the likelihood of the recommended user positively responding. We extensively evaluate our recommendation method in a group of active users of an operational online dating site. We find that our method is significantly more effective in increasing the number of successful interactions compared to a state-of-the-art recommendation method.

CCS CONCEPTS

• **Information systems → Recommender systems**;

KEYWORDS

Reciprocal Recommender Systems; Optimization; Machine Learning; Online-dating Application

1 INTRODUCTION

Reciprocal recommender systems (RRS) recommend people to people [19], as opposed to traditional recommender systems which recommend items to people. There are many potential applications for RRSs, such as online-dating platforms and automated job recruiting services.

RecSys '18, October 2–7, 2018, Vancouver, BC, Canada

ACM ISBN 978-1-4503-5901-6/18/10…$15.00
https://doi.org/10.1145/3240323.3240349

RRSs are inherently different than traditional recommender systems. In "item-to-people" recommendations, the success of the recommendation is determined by the acceptance of the recommendation by the receiver (also termed *service user*). In contrast, in RRSs, a successful recommendation is one that leads to a *successful interaction*, meaning that: 1) the service user accepts the recommendation and initiates an interaction with the recommended user; and 2) the recommended user replies positively [19]. Therefore, a beneficial recommendation in a RRS should account for both the service user's interest and the recommended user's interest, which makes recommendation generation more complex than in the case of a regular non-reciprocal recommender.

Several algorithms aimed at recommending people to people. have been proposed in the past. These algorithms have demonstrated that considering both the potential interest of the service user in the recommended user as well as the potential interest of the recommended user in the service user will bring about better results than merely considering the interest of one side (e.g. [18, 23]). However, we argue that these algorithms are limited in two major ways: first, existing algorithms give equal importance to the perceived interests of both sides when in fact users may vary in how they act upon their interests. For example, research focused on online-dating has demonstrated that users differ in the extent to which they consider the likelihood that the contacted partner will reply before sending a message [9, 23]. Secondly, to the best of our knowledge, prior RRS methods were only evaluated offline by using historical data, and were not integrated into an operational system in order to provide recommendations and investigate their real world effect on users' behavior. Recently there has been a growing recognition in the recommender systems community that conclusions from laboratory-based evaluations may not be confirmed in live-user studies [16, 22].

Therefore, in order to overcome these limitations, in this work we present a novel method for RRSs which optimally balances the interests of both sides in order to bring about successful interactions (i.e., positive replies of the recommended users to the service users who contacted them). We call our method Reciprocal Weighted Score (RWS). RWS relies on the the separate calculation of two scores: 1) the service user's presumed interest in the recommended user, which is estimated using a *collaborative filtering method*; and 2) the likelihood that the recommended user will reply positively to a message from the service user, which is estimated using a *machine learning model* built specifically for this purpose. Then, RWS balances these two scores into a single score which is a proxy

to the probability of forming a successful interaction. This optimal balance is tailored individually for each service user according to her interaction history, and thus RWS is personalized.

In order to demonstrate the importance of individual optimal balance, consider the following scenario: say Alice and Bob are users in an online-dating environment. Bob sends messages to a wide range of users without considering whether or not his messages will be accepted. In addition, Bob is unpopular (i.e., rarely receives messages) and the rate of positively replied messages to Bob is low. Therefore, in order to maximize successful interactions, an optimal RRS should generate recommendations for Bob that mainly focus on the chances for a reply. Alice, on the other hand, is very cautious about sending messages and only sends to a narrow range of users. In addition, she has a high rate of positive message replies. Therefore, in contrast to Bob, for Alice an RRS should generate recommendations that give more importance to the chances she will find a potential partner appealing.

The evaluation of RWS was conducted on an operational online-dating platform. We have compared this method to a previous RRS approach which was proven to be very effective [23]. In an experiment involving 398 subjects, we found that RWS significantly increases the number of successful interactions in comparison to the previous approach. On the other hand, as one would expect, RWS decreases the number of accepted recommendations. This is not surprising since our method prioritizes the probability of a reply rather than the probability that the service user will accept a recommendation.

This paper's contributions are threefold: 1) we present RWS, a novel recommendation method for RRSs which leverages an adaptive boosting (AdaBoost) machine learning model [7] and traditional collaborative filtering; 2) we construct a method for optimizing the successful interactions by optimally balancing the importance of the model's components that predict the service user's and the recommended user's behaviors; 3) we evaluate our methods online in an active dating application and find that our method significantly improves the number of successful interactions.

2 BACKGROUND AND RELATED WORK

2.1 Reciprocal Recommender Systems

Applications that require RRSs have unique characteristics, which present opportunities and challenges for providing successful recommendations [19]. Perhaps the most significant difference between RRSs and traditional item-to-people recommender systems is that RRS's recommendations must satisfy both parties, the service user and the recommended user. Another important issue that should be addressed in RRSs is avoiding recommendations of "popular" users, meaning users who receive a lot of messages, regardless of the recommendations [15].

In the past decade, many researchers have investigated the field of RRSs and specifically the domain of online-dating. In typical online-dating environments, users can create a profile, browse other users' profiles and interact with users by sending messages. Some online-dating environments include an option for explicitly rating profiles or pictures. Brozovsky and Petricek [4] show that in such environments, collaborative filtering algorithms, which leverage similarity between users assessed from their explicit ratings, are significantly more effective in comparison to other algorithms which were commonly used in on-line dating sites. However, online-dating sites do not generally include explicit ratings. Therefore, the recommendation methods for online-dating environments commonly elicit the users' interests from their interaction history. Krzywicki et al. [14] show that a collaborative filtering method, which derives the similarities between users from their interactions, is applicable and effective in the domain of online-dating.

Later, Pizzato et al. [19] show the importance of taking into account the reciprocity of recommendations in online-dating environments. Namely, in order to generate successful recommendations, the recommender system must measure both the interest of the service user in the recommended user and the interest of the recommended user in the service user. In their study, the authors present a content-based algorithm (named RECON) which, for a potential match, calculates the compatibility of each side's attributes to the other side's presumed interest. Pizzato et al. also define a new evaluation metric to evaluate the performance of the recommender system in providing recommendations which lead to successful interactions. We use this evaluation metric in our evaluation process (Section 4). Xia et al. [23] show that a collaborative filtering method, which contemplates the interests of both sides of the match, outperforms the content-based algorithm described above. We will present this method in detail in Section 2.3 and we will use it as a baseline to benchmark our novel approach.

In [15], Krzywicki et al. present a different approach for recommendations in reciprocal environments. In their work, they present a two-staged recommendation algorithm which first generates recommendations using collaborative filtering and later re-ranks the recommendation with a decision tree "critic". They compare the algorithm with a baseline profile matching algorithm, which matches users according to common attributes and shows that their algorithm is superior. However, this method was not compared to the previous algorithms described above.

Another popular domain of RRSs is recommender systems for job recruitment sites. Similar to online-dating, a successful match requires mutual interests of both the employee and the employer. Hong et al. [10] introduce several algorithms for recommendations of jobs to employees. They conclude that a job recommender system should apply different recommendation approaches for different users according to their characteristics. The 2016 ACM Recommender Systems Challenge [1] focused on the problem of job recommendations. The participant teams were given a large dataset from XING[1], a career-oriented social network, that consisted of anonymized user profiles, job postings, and interactions between them. The goal of the teams was to predict job postings with which a user will interact. This problem is somewhat similar to the problem of predicting a user's reply to a message on an online-dating site, which we address in this work.

2.2 Collaborative Filtering RRS

Standard collaborative filtering utilizes similarity relations between users or items in order to generate recommendations. As mentioned above, the interests of a user in online-dating are commonly inferred

[1] https://www.xing.com

from her interaction history [14], as we assume an initial message from user x to user y indicates that y fits user x's interests. In [23], Xia et al. present a similarity measure for users in online-dating. The similarity between two users x and n is defined as follows:

$$Similarity_{x,n} = \frac{ReFrom_x \cap ReFrom_n}{ReFrom_x \cup ReFrom_n}$$

where:

$$ReFrom_x = \{y : y \text{ has received a message from } x\}$$

Similarly, the group of users to whom x has sent an initial message is defined as follows[2]:

$$SentTo_x = \{y : y \text{ has sent a message to } x\}$$

Using this similarity measure, Xia et al. [23] introduced a recommendation method for online-dating which utilizes collaborative filtering to measure both the interest of the service user in the recommended user and the interest of the recommended user in the service user. As mentioned above in Section 2.1, they found that this approach significantly outperforms RECON, a content-based method which also contemplates the interest of both sides. Algorithm 1 describes this method and Figure 1 illustrates an example for the calculation of the mutual interest between two users[3]. We call this method Reciprocal Collaborative Filtering (RCF) and we will use it as a baseline to benchmark our novel approach.

Algorithm 1 Reciprocal Collaborative Filtering Recommendation

Input: service user x
Output: top-k recommendations

1: $Recs \leftarrow \emptyset$
2: **for all** $y \in RecommendationCandidates$ **do**
3: $score_{x,y} \leftarrow 0, score_{y,x} \leftarrow 0$
4: **for all** $u \in SentTo_y$ **do** ▷ calculate x's interest in y
5: $score_{x,y} \leftarrow score_{x,y} + Similarity_{x,u}$
6: **for all** $v \in SentTo_x$ **do** ▷ calculate y's interest in x
7: $score_{y,x} \leftarrow score_{y,x} + Similarity_{y,v}$
8: $score_{x,y} \leftarrow \frac{score_{x,y}}{|SentTo_y|}$ ▷ normalize scores
9: $score_{y,x} \leftarrow \frac{score_{y,x}}{|SentTo_x|}$
10: **if** $score_{x,y} = 0$ or $score_{y,x} = 0$ **then**
11: $reciprocalScore_{x,y} \leftarrow 0$
12: **else**
13: $reciprocalScore_{x,y} \leftarrow \frac{2}{score_{y,x}^{-1} + score_{x,y}^{-1}}$ ▷ save the harmonic mean of both scores
14: $Recs \leftarrow Recs + (y, reciprocalScore_{x,y})$
15: **sort** $Recs$ and **return top-k**

[2]In [23], $ReFrom_x$ is denoted as Se_x and $SentTo_x$ is denoted as Re_x.
[3] This method utilizes user-to-user similarities. Another option for finding the mutual interest is to use item-to-item similarities, meaning the attractiveness similarity of the recommended user to the group of users who received messages from the service user. This option was also examined in [23]. Both of these methods significantly outperformed RECON and there was no significant difference between them. We chose the first method because it performed slightly better than the second.

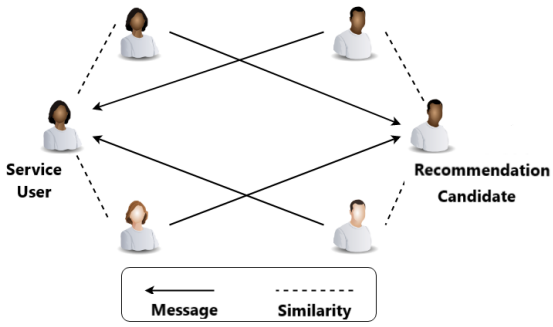

Figure 1: Reciprocal collaborative filtering visualization

2.3 The Online-dating Environment

In order to evaluate the recommendation method presented in this work, we collaborated with the owners of an Israeli active online-dating application, called *Doovdevan*. Doovdevan is a web and mobile application for Android and iOS devices. Similar to other online-dating applications, users of this environment can create a profile, search for possible matches and interact with other users by sending and receiving messages. Doovdevan currently serves about 32,000 users and is growing rapidly. We chose to perform our experiment in this environment because it is relatively new and the users had not yet received recommendations from the system prior to the experiment. This was important since previous recommendations can affect the trust of the users in the system and subsequently affect the acceptance of the recommendations [5, 13].

3 OPTIMAL WEIGHTING APPROACH

In this section, we introduce RWS, a novel algorithm for RRSs. In order to measure the compatibility of a recommendation of user y to user x we initially calculate two measurements separately: we use user-to-user similarity in order to estimate x's interest in y and an AdaBoost machine learning model for predicting whether y will respond positively to x. Moreover, in contrast to previous methods, which assign equal importance to the interest of both sides for all users, in RWS the relative importance of the two is tailored for each user individually. Namely, based on each user's previous interactions, we optimize the relative importance of the two scores for this specific user and tune the weights accordingly. Figure 2 is a diagrammatic representation of our recommendation method.

As mentioned above, RWS measures the interest of the service user in the recommended user using collaborative filtering. This score is calculated identically to the way in which the service user's interest is calculated in the RCF method described above (lines 4-5 in Algorithm 1). We denote the score of user x's interest in user y as $CF_{x,y}$. The second score is calculated by an AdaBoost machine learning prediction model (described in Section 3.2), which predicts the chances of a positive reply from user y to user x following an initial message from user x to user y. We denote this measure as $PR_{y,x}$.

We use two different prediction models since the prediction tasks are inherently different. The prediction of the service user's interest

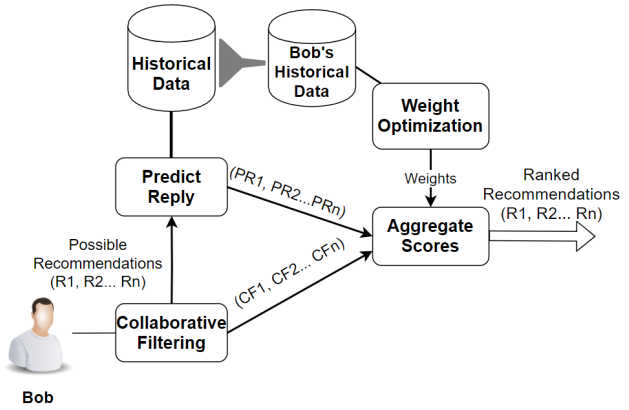

Figure 2: A diagrammatic representation of RWS. The "Predict Reply" component is described in Section 3.1 and the "Weight Optimization" component is described in Section 3.3

is similar to non-reciprocal recommendation tasks and therefore we use collaborative filtering methodology. On the other hand, the prediction of the reply of a recommended user can be better framed as a 2-class classification problem.

3.1 Predicting Replies of Recommended Users

Our reply prediction model was trained on 35,000 samples of messages contained in the dataset provided by Doovdevan, each including a list of features. We used the AdaBoost classifier, which we found to outperform other machine learning algorithms we tried on our dataset. The samples in the dataset are classified into two classes: 1) positive reply and 2) no reply or negative reply[4]. As our goal in this work is to increase positive interactions, we do not distinguish between a negative reply or no reply at all.

The features of a message sent to a recommended user can be divided into two main groups: 1) features describing the sender and 2) features describing the recipient. Each of these groups can be divided into two subgroups: 1) attributes of the user from her public profile, for example: age, gender, height, profession; and 2) features describing the activity and popularity of the user, such as the number of received messages, sent messages, views, logins. We first consulted a domain expert, who does not co-author this paper, in order to find potentially influential features and later we reduced the number of the features to 54 using the backward elimination feature-selection method [12]. In Table 1 we present the most prominent features, ordered by their information gain [8].

We denote the vector of feature values for a specific service user x as:

$$\mathbf{x} = (x_1, x_2, x_3 \ldots x_s)$$

where s is the number of the sender's features.

Similarly, we denote the vector of feature values for recommendation candidate y as:

[4]We manually classified all the samples which included a response into two classes: 1) positive response; and 2) negative response.

	Feature
Features of recipient	1) Percent of positively replied messages before current message. 2) Log-ins to the environment in the week before the message. 3) Number of profiles he/she viewed.
Features of sender	4) Number of users who viewed him/her. 5) Number of messages he/she received.

Table 1: Prominent features used in the reply prediction model, ordered by their information gain.

$$\mathbf{y} = (y_1, y_2, y_3 \ldots y_r)$$

where r is the number of the recipient's features.

For any given service user x and potentially recommended user y, we denote the probability for a positive response of y to a message from x as:

$$PR_{y,x} = h(\mathbf{x}, \mathbf{y})$$

where h is the function learned by the AdaBoost model, which returns the probability (value between 0 and 1) for a positive reply.

In our dataset, only about 7 percent of the initial messages were classified as positively replied, and therefore in order to improve our model we used the random oversampling class-balancing technique [2]. The area under the curve (AUC) of the model is 0.833[5].

In the next section we describe how RWS leverages this prediction model in order to generate recommendations.

3.2 Optimally Balancing Receiver and Recommended Users' Importance

In Algorithm 2 we give the general scheme for our recommendation algorithm, where *ServiceUserFeatures* (row 10) is a function which obtains the service user x's feature vector $\mathbf{x}$, as denoted above, and *RecommendedUserFeatures* (row 11) obtains the recommended user's feature vector $\mathbf{y}$. The *PredictReply* (row 12) function returns the probability of a positive reply according to our predictive model function h. The *OptimizedWeight* function (row 13) retrieves a weight, optimized specifically for the service user, as described in Section 3.3.1. Later (row 14), our method utilizes these weights to aggregate the *CF* and *PR* scores into a single score that resembles the reciprocal interests of the match.

Notice that for a given user x, the algorithm only predicts the probability of a reply for potentially recommended users y where $CF_{x,y}$ is not null (rows 7-9). In this way we reduce the size of possible candidates to a smaller subset of users. The importance of this reduction will be discussed in detail in Section 5.

In the following section we will describe the method we use to optimize these weights.

[5]For comparison, following are the best AUC scores received by other prediction models which we tested: 1) random forest classifier: 0.798; 2) logistic regression: 0.795; 3) multi-layer-perceptron classifier: 0.791; 4) Gaussian naïve Bayes classifier: 0.672.

Algorithm 2 Reciprocal Weighted Score Recommendations Scheme

Input: service user x
Output: top-k recommendations

$Recs \leftarrow \emptyset$
for all $y \in RecommendationCandidates$ **do**
  $CF_{x,y} \leftarrow 0$
  for all $n \in SentTo_y$ **do** ▷ calculate x's interest in y
    $CF_{x,y} \leftarrow CF_{x,y} + Similarity_{x,n}$
  $CF_{x,y} \leftarrow \frac{CF_{x,y}}{|SentTo_y|}$ ▷ normalize score
  if $CF_{x,y} = 0$ **then**
    $reciprocalScore_{x,y} \leftarrow 0$
  else
    $\mathbf{x} \leftarrow ServiceUserFeatures(x)$
    $\mathbf{y} \leftarrow RecommendedUserFeatures(y)$
    $PR_{y,x} \leftarrow PredictReply(\mathbf{x}, \mathbf{y})$ ▷ predict y's response to x
    $\alpha \leftarrow OptimizedWeight(x)$
    $reciprocalScore_{x,y} \leftarrow (\alpha \cdot CF_{x,y} + (1-\alpha) \cdot PR_{y,x})$ ▷ aggregate scores
  $Recs \leftarrow Recs + (y, reciprocalScore_{x,y})$
sort $Recs$ and **return top-k**

3.3 Weight Optimization

We aim at finding, for each service user x, a weight α_x which balances $CF_{x,y}$ and $PR_{y,x}$ (for each recommendation candidate y) so that it will optimize x's successful interactions. We denote the weighted score of user x for the recommended user y as $RWS_{x,y}$, and it is calculated as follows:

$$RWS_{x,y}(\alpha_x) = \alpha_x \cdot CF_{x,y} + (1-\alpha_x) \cdot PR_{y,x}$$

Note that for both the CF and PR scores we use the standard score [6] rather than the original score.

In order to find a specific weight optimized for x, we observe from the user's *interaction history* the influence of each score (*CF* and *PR*) on her successful interactions. We first define the following sets:

$$SuccInter_x = \{y : x \text{ has sent an initial message to } y \text{ and } y \text{ replied positively}\}$$

$$V_x = \{y : x \text{ has viewed } y\}$$

In addition, we denote with $RWS_{x,*}(\alpha_x)$ the scoring function of user x when a particular α_x is employed. Moreover, we denote with $Rank_y\left(RWS_{x,*}(\alpha_x)\right)$ the rank position of y in the list of the viewed users $v \in V_x$, sorted by decreasing value of the scoring function of user x.

We now define our target optimization problem for a specific user x, which we denote as *IndividualOptimization*:

$$\begin{aligned} \underset{\alpha_x}{\text{minimize}} \quad & \sum_{v \in V_x} \mathbb{1}_{v \in SuccInter_x} Rank_y\left(RWS_{x,*}(\alpha_x)\right) \\ \text{subject to} \quad & 0 \geq \alpha_x \geq 1 \end{aligned}$$

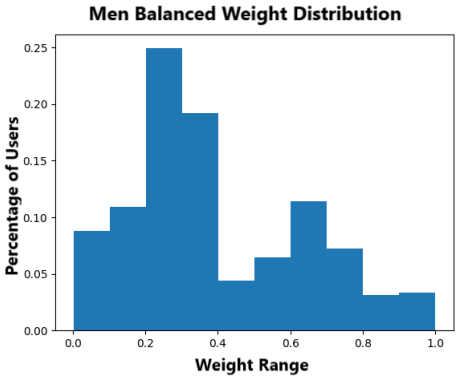

Figure 3: Distribution of the alpha weight for men in the Doovdevan environment

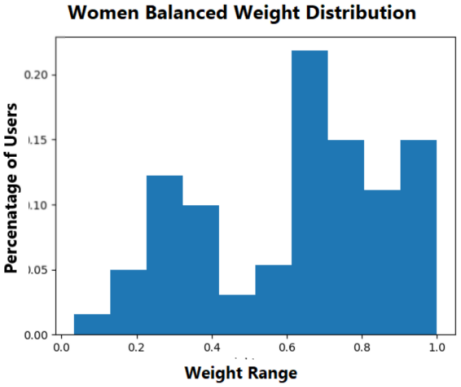

Figure 4: Distribution of the alpha weight for women in the Doovdevan environment

By solving this optimization problem we find the weight α_x which will rank the users with whom x had successful interactions higher than all other users that x has viewed. For the implementation of the optimization, we used Brent's (numerical analysis) method [3], which finds a local minimum in a given interval.

In Figures 3 and 4 we show the distribution of the alpha weight for 765 male and 566 female users randomly chosen from the Doovdevan environment. These figures demonstrate the importance of individual weighting: it is clear that in order to maximize the successful interactions, the balance of the *CF* and *PR* scores must vary substantially among different users. We can also observe that for most of the women the *CF* score is a stronger indicator for a successful interaction than *PR*, while for most of the men *PR* is a better indicator.

Notice that the optimization problem described above is only effective for users who had at least one successful interaction. In order to also recommend to users who did not have previous successful interactions, we define a similar optimization problem *over all users*, which we denote as *GlobalOptimization*:

$$\begin{aligned} \underset{\alpha}{\text{minimize}} \quad & \sum_{u \in U} \sum_{v \in V_u} \mathbb{1}_{v \in SuccInter_u} Rank_v \left(RWS_{u,*}(\alpha) \right) \\ \text{subject to} \quad & 0 \geq \alpha \geq 1 \end{aligned}$$

where U is the set of all users and $Rank_v \left(RWS_{u,*}(\alpha) \right)$ is the rank position of v in the ranked list of the users in U sorted by decreasing value of the $RWS_{u,*}(\alpha)$ score. In our environment, the calculated value of α for this global optimization problem is 0.3978. This result shows that, as expected, the PR score is a better predictor of a successful interaction than the CF score, for the average user.

4 EVALUATION

4.1 Experimental Setup

Our original intention was to compare several competing methods with RWS. However, due to the constraints imposed on us by our collaborators from the online-dating site, we were limited to studying only two conditions. Hence, we compared RWS to RCF (described above in Section 2.3), which has been shown to outperform previous approaches.

Our experiment involved a group of 398 randomly selected active users from Doovdevan, ranging in age from 18 to 70 (mean = 34.9, s.d.= 12.9), of which 24% (n=97) were female. We randomly divided the subjects into two conditions. Both condition groups received recommendations. The dependent variable was the recommendation method: the first group received recommendations based on the RCF method, and the second received recommendations based on RWS, our proposed method.

All subjects received the top three recommendations generated by the recommendation method of their respective condition. The recommendations were proposed once a day for three days.

4.1.1 The Optimized Weight of the RWS Condition. Before generating the recommendations, we calculated the optimized weight for all users in the RWS condition. About 89% of the subjects received individual optimized weights (the remaining subjects had no successful interactions). The average optimized weight was 0.411 and the median weight was 0.349. This indicates that to most of the users in our proposed method condition, the method gave higher importance to the score that measures the probability that a recommended user reply to a service user, while RCF method gives equal importance to the interests of the service user and the recommended user.

4.1.2 The Service User's Interface. The service user's interface supports three interaction stages:

(1) The system generates a recommendation and sends it to the service user's inbox. In addition, the service user receives a notification on her smartphone. The recommendation has a unique label that distinguishes it from other incoming messages in the inbox (left snapshot in Figure 5). The recommendation includes a brief description of the recommended user: low-resolution picture, name, age, location, marital status. The service user may decide to click on the recommendation.
(2) If the user clicks on the recommendation, she moves to a new screen showing a higher quality picture of the recommended user and a text stating that the recommendation was made based on her personal characteristics (right snapshot in Figure 5). The service user can then decide whether or not to send a message to the recommended user.
(3) If the service user opts to send a message to the recommended user, the recommended user can reply with a message or ignore the chat request.

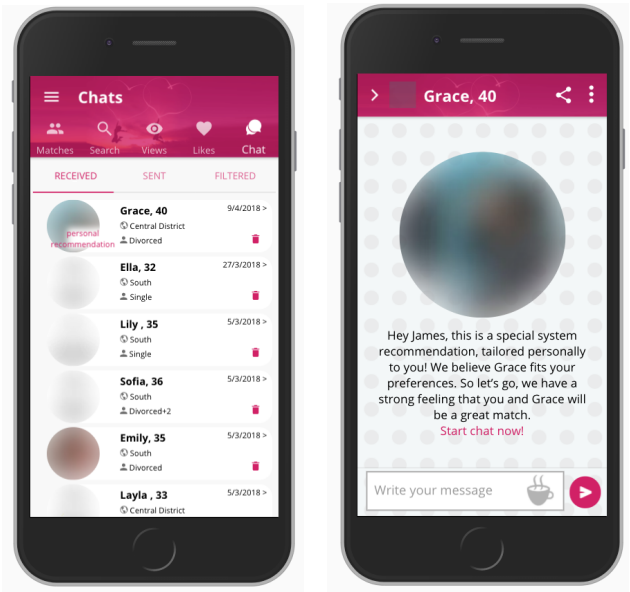

Figure 5: Screen shots of the recommendation's user interface. The left image is a screen shot of the inbox of a user who received a recommendation. In this case, the recommendation appears at the top. The right image is what the user sees after clicking on the recommendations. The pictures are blurred for reasons of privacy.

4.2 Evaluation Metrics

We have measured three important indicators of the success of a reciprocal recommender system: 1) The number of recommendations that were clicked on by the service user; 2) The number of recommendations that the service user accepted, meaning that she/he initiated a chat with the recommended user; and 3) The number of messages sent by the service user to which the recommended user replied positively. As our objective in this work is to increase the amount of successful interactions, our main focus is on the third indicator. For a given service user x we define the following four sets of users:

(1) RO_x is the set of recommended users who have been recommended to x and viewed by x in her inbox[6].
(2) RV_x is the set of recommended users whom x has viewed in her inbox and then clicked on in order to browse more detailed information.

[6]Some users did not view all of the recommendations, either because they did not log-in in the week following the recommendations or because they did not view their inbox.

Measure \ Condition	RCF	RWS
RO	320	356
RV	174	147
RM	171	138
M	889	1945
RI	1	8
I	99	322

Table 2: The summation of the results for all users in both conditions, evaluated a week after provision of the recommendations

(3) RM_x is the set of users who were recommended to x and received a message from x during the evaluation period.
(4) RI_x is the set of users recommended to x and their recommendations were followed by a successful interaction, i.e., x sent a message to the recommended user and the recommended user replied positively.

We first measured the number of messages that were clicked on by the user after viewing the messages in the inbox:

$$VPrecision_x = \frac{|RV_x|}{|RO_x|}$$

This metric evaluates the performance of the methods in recommending users who seem interesting enough to the service user so that she clicks on them in order to receive more information. The metric is only applicable for users who have viewed at least one recommendation in their inbox.

In addition we used four metrics for evaluation of RRSs defined in [19, 23]:

$$MPrecision_x = \frac{|RM_x|}{|RV_x|} \quad MRecall_x = \frac{|RM_x|}{|M_x|}$$

where M_x is the group of users who received messages from x in the evaluation period. We measure the successful interactions as follows:

$$RPrecision_x = \frac{|RI_x|}{|RM_x|} \quad RRecall_x = \frac{|RI_x|}{|I_x|}$$

where I_x is the total number of successful interactions of user x in the evaluation period.

4.3 Online Study Results

We examined the results a week after the provision of the recommendations. In Table 2, we show the summation of all of the results for all users in each condition.

We evaluated the performance of the recommendation methods by comparing the mean of the metrics defined above using a standard t-test. All of the results for both conditions were found to be distributed normally according to the Anderson-Darling normality test [20].

We first evaluated the average *VPrecision*. Our results show that the RCF method obtained significantly higher results (RCF: mean=0.57, s.d.= 0.43 vs. RWS: mean= 0.43, s.d.= 0.42), meaning that the recommendations that were provided by the RCF method looked more interesting to the users when they were scanned in their inbox.

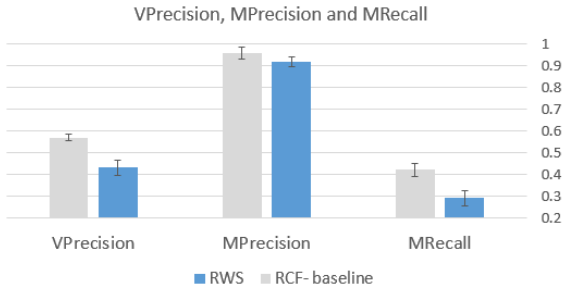

Figure 6: VPrecision, MPrecision and MRecall Metrics. Error bars represent the standard error

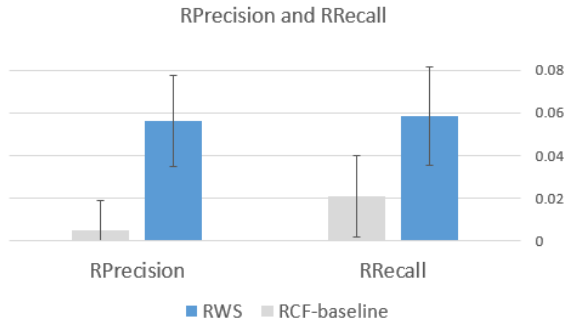

Figure 7: RRecall and RPrecision Metrics. Error bars represent the standard error

Regarding the *MRecall* metric, we found that the RCF approach significantly outperformed RWS (RCF: mean=0.42, s.d.= 0.38 v.s RWS: mean=0.29, s.d. =0.35). The *MPrecision* metric of both conditions is similar,with no statistically significant difference (mean=0.96, s.d.= 0.28 vs. mean= 0.92 , s.d.= 0.23). The mean and the standard error of *VRecall*, *MRecall* and *MPrecision* are presented in Figure 6.

These results indicate that the users recommended by RCF were evaluated as more appealing compared with those recommended by RWS. This finding was not surprising, as our method aims at optimizing successful interactions. In addition, as described above in Section 4.1.1, for most of the subjects our proposed method gives relatively less importance to the service user's interest.

Considering the *RRecall* and *RPrecision*, which measure the effectiveness of the algorithms in providing recommendations that lead to a successful interaction, we found that RWS significantly outperforms RCF with respect to *RPrecision* (RCF: mean=0.01, s.d.= 0.05 vs. RWS: mean= 0.06, s.d.= 0.21). Also, with respect to *RRecall*, RWS gave better results, but the difference is not significant (RCF: mean=0.02, s.d.= 0.14 vs. RWS: mean= 0.06, s.d.= 0.21). Note that *RRecall* is less important for our evaluation, as our optimization is based on precision rather than recall. The mean and the standard error of *RRecall* and *RPrecision* are presented in Figure 7.

In addition, we found that the average weight (α) assigned to a subject who had a successful interaction following a recommendation was 0.194, while the average weight of all subjects was 0.411, as mentioned above. This means that for these users, our method gave a substantially higher importance to the reply prediction model in comparison to the remaining subjects.

4.3.1 Popularity of the recommended users. We have also analyzed the popularity of the users who were recommended by both methods. The popularity of the users is commonly estimated by the number of messages received during a specific time period [15]. We measured the total number of messages received in the thirty days before the recommendation provision. We found that the popularity of the active recommended users[7] in the RCF condition was significantly higher than the RWS condition (RCF: mean= 59.49, s.d.= 45.14 vs. mean= 32.72, s.d.= 35.06, $p < 0.01$). This result indicates that our method recommends less popular users. In fact, it is very important to avoid recommending popular users, especially in online dating applications, where popular users are typically overwhelmed by incoming messages [17].

4.3.2 Runtime. We measured the average runtime for generating recommendations for a single user by both methods. The average runtime of the RCF method was 1.47 seconds, while in our proposed method the average runtime was 6.97 seconds, including an average of 2.61 seconds for the optimization calculation. However, in practice, the runtime has no impact on the user experience since the recommendations in our application are delivered as messages and are not pulled by the users.

5 DISCUSSION AND FUTURE WORK

The main outcome of this work is that predicting which recommended user will reply to a service user's message and combining this prediction with a standard CF method significantly improves the number of successful interactions in comparison to the RCF method. However, as we expected, it also reduces the amount of accepted recommendations. An additional benefit of focusing the recommendation on the prediction of the chances for a reply is that it reduces the recommendation of popular users.

We note that in this work we have performed an online comparison of a baseline CF method with our novel approach called RWS. We have preferred this type of evaluation to a simpler offline experiment (on historical data) due to the limitations of offline evaluations. Specifically, in order to rely on an offline evaluation, one must assume that behavioral data collected before the introduction of the recommender, which are used to test the performance of the recommender itself, are similar to what one could have observed after the recommender was made available. Unfortunately this is an incorrect assumption [16, 22]. Nevertheless, offline experiments are useful for making a first screening of candidate approaches and discarding totally inappropriate ones [22]. Therefore, we are currently implementing an offline experimental procedure to test the application of alternative solutions in an offline setting that simulates the online analysis. This will enable us to further optimize the proposed solution and better understand the importance of various components. In addition, we intend to compare our recommendation method with other, previously proposed, recommendation methods for RRS, in an offline experiment.

Although we evaluated our method in an online-dating environment, we speculate that its general scheme would be effective in other domains too, specifically in domains where partially conflicting interests arise [21]. For example, in the domain of online job-recruiting, we would use two prediction models for generating recommendations: 1) a prediction model for predicting job-offers which will fit a job-seeker's interest; and 2) a different prediction model which will predict whether the company would accept an application. Both components will be balanced with an optimized weight for each individual job-seeker according to her history. We are currently working on extending our recommendation approach to other domains, such as recommender systems which assist users in finding roommates and online job-recruiting platforms.

In addition to the decline in the recommendation acceptance rate discussed above, a limitation of the proposed method is that it requires a longer runtime for generating recommendations, as mentioned in Section 4.3.2. Although the runtime did not influence the user's experience, it is possible that in a large-scale deployment it will have an effect on the performance of the system. The latency of our proposed method is mainly caused by the reply prediction component and the computation of the scores' balancing weight optimization. Specifically, each reply prediction requires the extraction of several features, which is time consuming. For this reason, it was important to reduce the number of recommendation candidates, as described above in Section 3.3. In order to further improve the system's performance, we suggest calculating the optimized scores balancing weights offline, before providing the recommendation. The optimal weight can be saved and used later for several recommendations.

Moreover, in the Doovdevan environment, users were *not* charged for sending messages. We would like to extend our work to online-dating sites which have a different business model (e.g. pay per message), since the acceptance of a recommendation and the successful interaction may be influenced by that. As shown in [11], the introduction of explicit cost can bear a significant influence over users' behavior in online dating environments. In addition, we would like to assess the influence of the recommendations on the willingness to pay for the service.

[7] We only focus on the active recommended users, since non-active users receive fewer messages regardless of their popularity and attractiveness.

REFERENCES

[1] Fabian Abel, András Benczúr, Daniel Kohlsdorf, Martha Larson, and Róbert Pálovics. 2016. Recsys challenge 2016: Job recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, 425–426.

[2] Gustavo EAPA Batista, Ronaldo C Prati, and Maria Carolina Monard. 2004. A study of the behavior of several methods for balancing machine learning training data. *ACM SIGKDD explorations newsletter* 6, 1 (2004), 20–29.

[3] Richard P. Brent. 1971. An algorithm with guaranteed convergence for finding a zero of a function. *Comput. J.* 14, 4 (1971), 422–425.

[4] Lukas Brozovsky and Vaclav Petricek. 2007. Recommender system for online dating service. *arXiv preprint cs/0703042* (2007).

[5] Henriette Cramer, Vanessa Evers, Satyan Ramlal, Maarten Van Someren, Lloyd Rutledge, Natalia Stash, Lora Aroyo, and Bob Wielinga. 2008. The effects of transparency on trust in and acceptance of a content-based art recommender. *User Modeling and User-Adapted Interaction* 18, 5 (2008), 455–496.

[6] Wilerid J Dixon and J Massey Frank. 1950. *Introduction To Statistical Analsis*. McGraw-Hill Book Company, Inc; New York.

[7] Yoav Freund, Robert Schapire, and Naoki Abe. 1999. A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence* 14, 771-780 (1999), 1612.

[8] Mark A Hall. 1999. Feature selection for discrete and numeric class machine learning. (1999).

[9] Günter J Hitsch, Ali Hortaçsu, and Dan Ariely. 2010. What makes you click?–Mate preferences in online dating. *Quantitative marketing and Economics* 8, 4 (2010), 393–427.

[10] Wenxing Hong, Siting Zheng, Huan Wang, and Jianchao Shi. 2013. A Job Recommender System Based on User Clustering. *JCP* 8, 8 (2013), 1960–1967.

[11] Akiva Kleinerman, Ariel Rosenfeld, and Sarit Kraus. 2018. Providing Explanations for Recommendations in Reciprocal Environments. In *Proceedings of the 12th ACM Conference on Recommender Systems*. ACM.

[12] Daphne Koller and Mehran Sahami. 1996. *Toward optimal feature selection*. Technical Report. Stanford InfoLab.

[13] Sherrie YX Komiak and Izak Benbasat. 2006. The effects of personalization and familiarity on trust and adoption of recommendation agents. *MIS quarterly* (2006), 941–960.

[14] Alfred Krzywicki, Wayne Wobcke, Xiongcai Cai, Ashesh Mahidadia, Michael Bain, Paul Compton, and Yang Sok Kim. 2010. Interaction-based collaborative filtering methods for recommendation in online dating. In *International Conference on Web Information Systems Engineering*. Springer, 342–356.

[15] Alfred Krzywicki, Wayne Wobcke, Yang Sok Kim, Xiongcai Cai, Michael Bain, Ashesh Mahidadia, and Paul Compton. 2015. Collaborative Filtering for people-to-people recommendation in online dating: Data analysis and user trial. *International Journal of Human-Computer Studies* 76 (2015), 50–66.

[16] Sean M McNee, John Riedl, and Joseph A Konstan. 2006. Being accurate is not enough: how accuracy metrics have hurt recommender systems. In *CHI'06 extended abstracts on Human factors in computing systems*. ACM, 1097–1101.

[17] OkCupid. 2015. Okcupid blog: a women advantage. https://theblog.okcupid.com/a-womans-advantage-82d5074dde2d. (2015).

[18] Luiz Pizzato, Tomasz Rej, Joshua Akehurst, Irena Koprinska, Kalina Yacef, and Judy Kay. 2013. Recommending people to people: the nature of reciprocal recommenders with a case study in online dating. *User Modeling and User-Adapted Interaction* 23, 5 (2013), 447–488.

[19] Luiz Pizzato, Tomek Rej, Thomas Chung, Irena Koprinska, and Judy Kay. 2010. RECON: a reciprocal recommender for online dating. In *Proceedings of the fourth ACM conference on Recommender systems*. ACM, 207–214.

[20] Nornadiah Mohd Razali, Yap Bee Wah, et al. 2011. Power comparisons of shapiro-wilk, kolmogorov-smirnov, lilliefors and anderson-darling tests. *Journal of statistical modeling and analytics* 2, 1 (2011), 21–33.

[21] Ariel Rosenfeld and Sarit Kraus. 2018. Predicting Human Decision-Making: From Prediction to Action. *Synthesis Lectures on Artificial Intelligence and Machine Learning* 12, 1 (2018), 1–150.

[22] Guy Shani and Asela Gunawardana. 2011. Evaluating recommendation systems. In *Recommender systems handbook*. Springer, 257–297.

[23] Peng Xia, Benyuan Liu, Yizhou Sun, and Cindy Chen. 2015. Reciprocal recommendation system for online dating. In *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015*. ACM, 234–241.