RECON: A Reciprocal Recommender for Online Dating

Luiz Pizzato, Tomek Rej, Thomas Chung, Irena Koprinska and Judy Kay
School of Information Technologies
University of Sydney, NSW 2006, Australia
{forename.surname}@sydney.edu.au

ABSTRACT

The reciprocal recommender is a class of recommender system that is important for several tasks where people are both the subjects and objects of the recommendation. Some examples are: job recommendation, mentor-mentee matching, and online dating. Despite the importance of this type of recommender, our work is the first to distinguish it and define its properties. We have implemented RECON, a reciprocal recommender for online dating, and have evaluated it on a large dataset from a major Australian dating website. We investigated the predictive power gained by taking account of reciprocity, finding that it is substantial, for example it improved the success rate of the top ten recommendations from 26% to 45% and also improved the recall at the same time. We also found reciprocity to help with the cold start problem obtaining a success rate of 26% for the top ten recommendations for new users. We discuss the implications of these results for broader uses of our approach for other reciprocal recommenders.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Information filtering*; H.1.2 [**Models and Principles**]: User/Machine Systems—*Human information processing*

Keywords

Recommender systems, Online Dating, Reciprocity

1. INTRODUCTION

Recommender systems have been largely based on the goal of providing a user with a set of recommended items that the system predicts that the user will like. In this paper, we focus on a rather neglected, but important, class of recommenders which we call *reciprocal recommenders* because they recommend people to people and a successful recommendations only occurs when *both* people like each other, or reciprocate. For instance, in an employment recommender system, potential employees seek suitable positions while employers, who offer the positions, are looking for suitable potential employees. There are many other important domains that involve reciprocal recommendations, including mentoring systems, business partner identification and, the focus of our work, dating sites that aim to help a person meet a suitable partner.

Table 1 defines the characteristics of reciprocal recommenders by comparing them against more traditional recommender systems. We now explain these briefly, with examples from the domain of online dating. The first row shows the defining feature of a reciprocal recommender. For example, if an online dating recommender offers a user, Andrew, a recommendation for Beryl, the success of this recommendation relies on both Andrew, and Beryl, liking each other. So the reciprocal recommender must take account of the preferences of both Andrew (the subject of the recommendation) and Beryl (who is recommended). The second row of the table notes that both users are aware of this need for reciprocity for success. This is important as it drives other important aspects listed later in the table.

The next pair of characteristics relate to the different roles of *explicit* profiles, provided by the user directly giving information about themselves and their preferences, and *implicit* profiles that a system creates based on the user's behaviour. Since the very early recommender systems, the latter has been recognised as valuable [14] because it can build a useful profile without any extra work by the user. The nature of reciprocal recommenders means that *explicit* profiles have a particularly prominent place.

Continuing our example, in online dating, like in the traditional matchmaker or marriage agency, a new user, Andrew, is expected to provide detailed information before he can receive recommendations. Reflecting the reciprocal nature of the recommendations, he will provide information about himself and his ideal partner by answering closed questions about attributes such as gender, age, and location. Some of these questions are inherently reciprocal, e.g. the type of the relationship sought: short-term or long-term. Andrew will also provide more complex forms of information such as his photo and free text description of himself and his ideal partner.

The role of *implicit* profiles is also distinctive. After Andrew creates his profile, he would be provided with an initial set of recommendations, each in the form of a short profile. His actions on the site would then provide data for his implicit profile. For example, on seeing Beryl's short profile,

Table 1: Characterising reciprocal recommenders

Typical recommender	Reciprocal recommender
Success is determined solely by the user receiving the recommendation.	Success is determined by both parties in the recommendation.
User receiving recommendation is *aware* that they are the sole arbiter of its success.	User is *aware* that success depends on the other person involved.
Users are typically reluctant to complete detailed *explicit* profiles.	Users are willing to provide detailed profiles
Users often develop a long history within the recommender site, therefore defining a substantial *implicit* profile of preferences.	Users may leave the recommender site after a short time, and may never return after a successful transaction. This makes the cold-start problem particularly acute.
The same recommendation can typically be made to many users.	People have limited availability, so one person should not be recommended to too many others.
Users are usually *proactive*, engaging with the system to find items.	Users can be *proactive* or *reactive*: a user is *proactive* if they take the initiative, selecting a recommendation; a user is *reactive* when they respond to contact.
It may be fine if some items are never recommended.	It is important that users be part of recommendations; this is particularly true for *reactive* users

he may click to see her full profile. This suggests he likes her. He may then send her a pre-defined message indicating he likes her, or he may just go back to exploring other recommendations. The latter action provides evidence that some information on the full profile discouraged him.

In an online dating system, it is likely that if Andrew finds a partner and establishes a long term relationship with her, he may never return to the site. This is unlike an online movie recommender where a user may continue to use the site if it works well. This difference is due to the people-to-people nature of reciprocal recommenders - when people are successful in finding a date, a job or a mentor, they may never return to the site. This is not an absolute characteristic; some people may need to find many jobs and others may seek many partners. However, the reciprocal recommender should be designed to cope with rather abruptly ended implicit profiles. These may indicate success as in the example of Andrew but also failure, e.g. Chris leaving the system dissatisfied because he did not receive satisfactory recommendations. The system may not have the data needed to distinguish between the cases of Andrew and Chris. Of course, the rich explicit profiles, can be used in conjunction with the available implicit profiles. But the design of a reciprocal recommender must take into account the cold start problem for implicit profiles.

We now consider the last set of defining characteristics. The first relates to the need to avoid overloading people. There is no problem if a book recommender system recommends the same book to a user and he/she buys it. However, in a people-to-people recommender system such as online dating, recommending the same person, say Beryl, to a large number of users should be avoided. First, Beryl can select only a small number of people to date (perhaps just one). Second, if she is extremely popular and is recommended to many people, she may become overwhelmed by the attention and stop responding. As a result other users trying to initiate contact with her may become discouraged as she is not replying or replies negatively.

The final row of Table 1 introduces another important aspect of reciprocal recommenders in terms of different roles that people may take. For example, Andrew may play an *proactive* role, initiating contact with people recommended by the system. By contrast, Beryl may play a *reactive* role, communicating mostly with those who initiated contact with her. Therefore, it is important that Beryl is present in people's recommendations, as she does not play an active role in the website. Similarly, on many job sites, the potential employee takes an proactive role while the employer plays a reactive role. Notably, it may be difficult for a system to quickly distinguish who are the reactive users so this principle could be applied to all users who are not yet known to be proactive. This highlights to the importance of solving the cold start problem for reciprocal recommenders.

We came to realise the distinctive nature of reciprocal recommenders in the context of our work with a major online dating website. Based on this realisation, we built RECON, a reciprocal recommender system that uses online dating historical data to make recommendations. To gain understanding of reciprocal recommenders and how to build them, we designed experiments to compare the performance of a recommender that makes use of reciprocity with one that does not. Then, given the importance of the cold-start problem on several levels, as described above, we designed experiments to assess the power of our recommender compared against the existing system.

The next Section 2 reviews the literature in reciprocal recommenders and social matching recommenders. Section 3 describes the three main parts of the RECON's algorithm: finding the user's preferences, calculating the compatibility scores, and producing the ranked list of reciprocal recommendations. We also provide details about our implementation of RECON. Section 4 presents the evaluation setup and the evaluation results, including the effect of reciprocity and the power of RECON for new users. Finally, Section 5 discusses the results and their implications for other reciprocal recommenders.

2. RELATED WORK

Given the range of domains that involve reciprocity, there have been several interesting works that have explored ways to build reciprocal systems. This section reviews some important examples of reciprocal systems in domains such as dating, job finding, expert finding, friend and colleague matching, matching students with people willing and able to help them. In addition, we briefly discuss some of the established techniques and their potential to create effective recommenders with the particular properties we have identified in Table 1, notably, related to the importance of: *explicit* profiles since they are particularly rich; *implicit* profiles when available; the need to avoid overloading or neglecting individuals; and the nature of the *cold-start problem* particularly for addressing the short stay of many users as well as the presence of *reactive* users.

We now describe some important work that involves reciprocity in a recommender. Notably, the broad area of social matching, recommending people to other people [24] has a clear link with reciprocal recommenders because the quality of a match is determined by those involved in the match. However, some existing work on social matching tailor only to the needs of one party [21]. While few papers discuss the concept of reciprocity and even fewer attempt to act on it, there has been some important work that informs the creation of reciprocal recommenders.

Beehive [6] is a social networking site within IBM. It helps users communicate, share information and befriend each other. Chen et al. explored the use of four algorithms, two mainly content-based and two mainly based on collaborative filtering. They evaluated these in terms of accuracy and the ability to give novel recommendations, concluding that all four approaches increased the number of connections compared to a control group that received no recommendations. The content-based technique provided the highest percentage of good recommendations among contacts who were not previously known; collaborative filtering worked well in finding previously known contacts.

The i-Help system [4] helped students find peers who were willing and able to help them with problems in computer science. The PHelpS system [10] applied a similar approach in a workplace so that an employee could find a person to help them with an unfamiliar task. Both systems relied on rich user models, or profiles, to take account of the expertise of potential helpers as well as the help requester, and their preferences for each other. This work involved small number of users as it was restricted to those electing to be involved in the system, and the only helpers considered were those logged into, and available, at the time of a request for help.

The bilateral recommendation approach for a job matching system [15] also addressed reciprocity. Apart from filtering candidates based on ability and aptitude, the "Job preference" part of the system made use of the ratings provided by the candidate for other job listings. The paper discusses the *bilateral matching problem*, focusing on finding a globally optimal solution when the preferences of both sides of a match is considered, such as a Pareto-optimal solution. The computational efficiency of finding this solution is considered, and some heuristics are suggested for addressing limitations. The applicability of this approach is rather limited for reciprocal recommenders on the scale of our online dating task. Notably, it is problematic to aim for global optimal solutions when the accuracy of the user preferences cannot be assured. This work takes an intuitive approach to deal with the preferences of both users in a recommendation, with two separate recommenders, each considering the preference of one side, and combining the output of both recommenders. More broadly, methods of combining different recommenders have been summarised in [5].

Online dating has received little attention in recommender systems research. Brožovský and Petříček [3] report an evaluation of their recommender for matchmaking, using variations of user-user and item-item collaborative filtering algorithms and several different benchmarks using a dataset from an online dating service. They mention the need for reciprocation but do not explore it. Current research indicates that profile information currently recorded by online dating websites can provide interest insights into the likes and dislikes of people. Hitsch et al. [11] studied which personal characteristics are more important for certain groups of users; for example that women discriminate on occupation more than men do. Fiore and Donath [8] observed a positive correlation in the personal attributes of people who match well. While these studies point to some useful approaches for building recommenders, it is clear that they are not good enough to be standalone solutions. Rather, they point to the potential value of a content-based technique for online dating recommenders.

The importance of explicit profiles means that we need to take account of the considerable literature associated with these. Notably, user input may be unreliable [1, 2]. As part of the foundation for our work, we have analysed the consistency of user's explicit profile and an implicit one based on their use of the system [20]. Unsurprisingly, there was a difference and it varied across elements of the profile.

In broader recommender work, reports on many systems indicate that users are generally reluctant to provide explicit profile information and feedback, favouring implicit feedback based on their activity [12, 16]. Moreover, such implicit profiles have shown good results [13, 7]. It is clear that we should exploit implicit profile information when it becomes available.

The cold-start problem is particularly important for reciprocal recommenders, both to ensure that new users can be given useful recommendations early in their use of the site, and to deal effectively with *reactive* users, who rely on being recommended to others, so they can respond. The cold-start problem is well recognised for recommenders in general, and earlier work has been summarised in [23]. Recent ideas on the problem include integrating the rating of items into the sign-up process to increase engagement [9], and using hybrid approaches such as performing linear regression on all user-item attribute pairs [17]. Although hybrid approaches are currently receiving much attention, we are still far from having one approach which solves the entire problem.

The ability to recommend novel items is perceived as an important feature by users. This problem of spreading recommendations is a known problem for content based filtering approaches. Clustering on seldom selected items [18] has been suggested to find groups of similar items. Demographic information can be used as another measure of similarity between people [19]. Dimensionality reduction [22] may reduce the number of classes of items. All these approaches relate similar items or users by finding new associations between them.

In summary, we can draw upon considerable existing understanding of the various defining aspects of reciprocal recommenders. We have also taken account of the range of above systems that have a reciprocal nature. At one level, the concept of reciprocity in recommender systems is not new. At the same time, we have not found other work that recognised and explicitly explored the distinctive character of reciprocal recommenders and how these impact the effectiveness of the recommendations. This paper builds upon our definition of reciprocal recommenders to provide a systematic base for designing them. We further use the definition to drive the principled evaluation of approaches, in the context of a large dataset from a commercial online dating website.

3. RECON'S ALGORITHM

The primary way a user expresses interest in another user is by sending a short, pre-defined message, such as "I like you, do you want to talk?".[1] The receiver of a message can

[1]We will simply refer to this type of communication as *messages* despite other types of messages such as emails also being possible in dating websites.

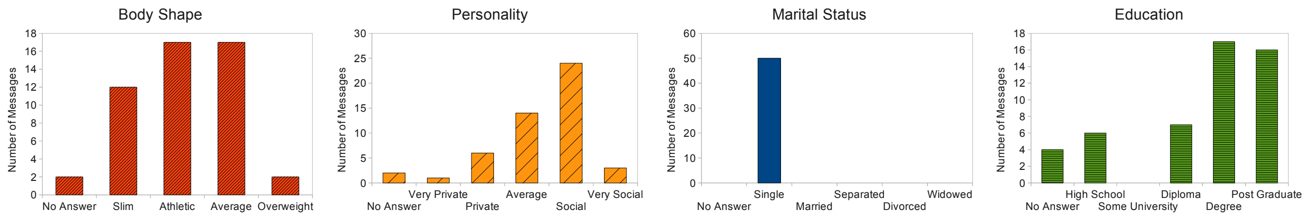

Figure 1: Example of distribution of messages sent by a user

reply either positively with another message such as “Loved the message, can you send me an email” or negatively with a message such as “Thanks, but I don’t think we are right for each other”. We observed that the positive responses to the messages directly correlates with the opening of other communication channels such as emails; these channels typically require paying a fee by the user.

Thus, the messages sent by a user and the responses to these messages contain important information about the user preferences. We use this information to build a preference model for user x by recording the attributes of all users to whom x sent a message and all users whose messages were positively replied to by x. We then compute the distribution of these attributes which gives a good indication of the type of user liked by x. For example, Figure 1 shows the distribution of all messages sent by a female user for four different attributes. It shows that this user does not have a strong preference for a particular body type as she messages the same number of users with *athletic* and *average* body and also a high number of users with *slim* body. However, this user has a strong preference for *singles* with a *social* personality who hold a university *degree* or *postgraduate* degree.

Finding the user’s preferences

The profile of a user is made of two components: free text information and a pre-defined list of attributes, A. Because of the complexity involved in working with natural language text, we will, at this stage, ignore the existence of the free text component and will describe RECON using only the pre-defined list of attributes. The list consists of nominal and numeric attributes; numeric attributes are discretised; for example, age is discretised into groups, such as 18-20 years and 20-22 years.

The profile U_x of a user x is represented as

$$U_x = \{v_a : \textit{for all attributes } a \in A\}$$

where v_a is a value of an attribute a.

Given a user x we first find the list of users M_x to whom user x has sent a message:

$$M_x = \{m : m \text{ is a user messaged by } x\}$$

The distribution of discrete values of an attribute a in a list of users M_x is represented by a list of values and counts. This distribution of values represents the preference $p_{x,a}$ of a user x for the values of an attribute a:

$$p_{x,a} = \{(v, n) : \textit{for all unique discrete values } v \textit{ of } a\}$$

such that n is the number of times v occurred in M_x

The preference P_x of a user x can be represented by a list of distributions $p_{x,a}$ for all attributes A:

$$P_x = \{p_{x,a} : \text{for all } a \in A\}$$

The preferences of a user are represented as a group of histograms showing the number of times each value has occurred for each profile attribute. Figure 2 shows the profile information and preferences of a group of example users: Alice, Amy, Bob and Ben.

Calculating compatibility scores

After obtaining the preferences P_x, we can generate a simple recommender system that finds all people who contain the attributes sought by the user. The preferences vary for different people and attributes, which will result in different number of recommendations. A user who has messaged a wide variety of people in a highly populated area is likely to receive a large number of recommendations; conversely, a user who has messaged only a few users, all containing similar characteristics in a sparsely populated area will receive very few, if any, recommendations.

For the first class of users who have messaged a wide range of users, because the number of recommendations may be larger than what the user is capable of judging or is willing to verify, it is necessary to sort the list of recommendations. Different strategies can be used for ranking the recommendations, which might involve several aspects of the website including the users’ previous behaviour (e.g. ranking users based on their engagement on the website) or other decisions motivated by a business model such as presenting premium customers before others, or favouring potential customers such as those who are engaged to the website but are neglected by other users. Nevertheless, one way of ranking the recommendation list includes calculating how well each recommendation matches the user’s preferences.

The RECON algorithm measures how much a user y matches the preferences of another user x using the preferences p_x and all the attributes of user y. Algorithm 1 is used to calculate this compatibility score.

Taking the example users of Figure 2, the compatibility score between the Alice’s preferences and Bob is calculated as:

$$\begin{aligned} Compat.(P_{Alice}, Bob) &= \frac{1}{\text{No. attr.}} \sum \frac{\text{(Bob's attr. in Alice's Pref.)}}{\text{(No. Alice's Mess.)}} \\ &= \frac{1}{3} \times \frac{9^{(Male)} + 6^{(20-24)} + 5^{(Athl.)}}{10} \\ &= 0.67 \end{aligned}$$

The compatibility scores between all pairs of users is shown in Table 2. Notice that because Ben never messaged male users, the compatibility score between Ben’s profile and Bob is 0. The same situation is observed between Amy’s profile and Alice.

Profile	U_{Alice}	U_{Amy}	U_{Bob}	U_{Ben}
Gender	Female	Female	Male	Male
Age	23	30	26	33
Body	Slim	Average	Athletic	Average

Preferences	P_{Alice}	P_{Amy}	P_{Bob}	P_{Ben}
$p_{x,Gender}$	(Male, 9) (Female, 1)	(Male, 20)	(Female, 95) (Male, 5)	(Female, 50)
$p_{x,Age}$	(20-24, 3) (25-29, 6) (30-34, 1)	(25-29, 5) (30-34, 10) (35-40, 5)	(20-24, 70) (25-29, 20) (30-34, 10)	(20-24, 5) (25-29, 20) (30-34, 20) (35-39, 5)
$p_{x,Body}$	(Athletic, 5) (Average, 4) (Slim, 1)	(Athletic, 18) (Average, 2)	(Slim, 50) (Average, 30) (Overweight, 20)	(Slim, 5) (Average, 20) (Overweight, 25)

Figure 2: Example of the Profile and Preferences of Alice, Amy, Bob and Ben

Algorithm 1: *Compatibility*(P_x, y)

```
Input: Preferences $P_x$, user $y$
Output: Compatibly score $s$
begin
    /* Return a small non-zero value if $x$ has no preference
       (e.g. $x$ did not sent any message)                      */
    if $P_x = \emptyset$ then
        return 0.001
    else
        Extract user $y$'s profile $U_y$
        foreach Attribute $a \in A$ list of attributes do
            Obtain the value $v_a$ of attribute $a$ in $U_y$
            Get $p_{x,a}$ in $P_x$ Find $(v, n) : (v, n) \in p_{x,a}, v = v_a$
            if $n = 0$ then
                /* e.g. male user $y$ for a strict
                   heterosexual male user $x$)                  */
                return 0
            else
                /* $s_a$ is the percentage of the distribution
                   that is equal to $v$                         */
                $s_a \leftarrow \frac{n}{\sum n \in p_{x,a}}$
        return $\frac{\sum s_a}{|A|}$          // Arithmetic mean of all $s_a$
end
```

$P_x \backslash y$	Alice	Amy	Bob	Ben
Alice	×	0.20	0.67	0.47
Amy	0.0	×	0.72	0.53
Bob	0.72	0.45	×	0.15
Ben	0.40	0.60	0.0	×

Table 2: Calculated compatibility scores between users

Creating a ranked list of reciprocal recommendations

After the user preferences are computed, we compute the list of recommended users for user x as shown in Algorithm 2. Every user y who x has not communicated with will receive a score that reflects how much the preferences of x match the attributes of y and how much the preferences of y match the attributes of x. We call this score a *reciprocal score*. The reciprocal score between x and y is the harmonic mean of the compatibility scores of these users. Then, the reciprocal scores are ranked and the top-N recommendations are presented to x. Table 3 shows the recommendations for Ben for our example, assuming that Ben has not communicated with the other users. Therefore, if we need to provide only one recommendation to Ben, we would recommend Amy as the reciprocal score with her is the highest.

Algorithm 2: ReciprocalRecommender(x,N)

```
Input: User Alice, $N$ number of recommendations to provide
Output: List of Recommendations $R$, List of Scores $S$
begin
    Find $P_x$
    $R \leftarrow M_x^c$                    // all users not messaged by $x$
    $S \leftarrow \{s_y : \forall y \in R\}$
    foreach $y \in R$ do
        $s_y \leftarrow Compat.(P_x, y)$
        if $s_y > 0$ then
            Find $P_y$
            /* Calculate the reciprocal score as the
               harmonic mean of the two reciprocal
               compatibility values                             */
            $s_y \leftarrow \frac{2}{((s_y)^{-1} + (Compat.(P^y, x))^{-1})}$
    /* sort by reciprocal score                                 */
    $R \leftarrow \{y_1, y_2, \ldots y_j \ : s_{y_i} \geq s_{y_{i+1}} \forall i\}$
    if $\exists N$ then
        $R \leftarrow R \setminus \{y_k : k > N\}$
    return $(R, W)$
end
```

x	*Comp.*(P_{Ben}, x)	*Comp.*(P_x, Ben)	*Reciprocal Score (Harm. Mean)*
Amy	0.60	0.53	*0.56*
Alice	0.40	0.47	*0.43*
Bob	0.00	N/A	*0.00*

Table 3: List of recommendations for Ben

Note that the reciprocal score is symmetric as the name suggests, i.e. y's score in the recommendation list for x is the same as x's score in the list for y. However, as the lists contain only the top-N recommendations, y may be in the top-N recommendations for x but the opposite may not be true.

RECON implementation

The RECON implementation slightly differs from the one described in the Algorithms 1 and 2. Despite the linear performance of the recommendation algorithm, making an assumption about the users that someone might like to communicate with is necessary to improve the runtime performance. This is particularly important if we want to find recommendations for all users in the database (i.e. a $O(N^2)$ process). Because of this, we initialise the recommendation list with a reduced set of users. This set is made of users of the dominant gender, who are within one standard deviation of mean age, and who live as far as one standard deviation from the mean location. This drastically reduces the number of users to be evaluated, and although we will possibly miss some good recommendations, because gender, age and location are the most important in online dating communication, the speed benefits are likely to be more important than the loss in recall.

One important aspect of any recommender system is the way it deals with the cold-start problem. In our recommender, cold-start means that we have a new user, who has not messaged or replied to anyone. In these cases the set of preferences P_x does not exist and the compatibility function will return a very small value (0.001); therefore, the recommender will create a list of users y ranked by the compatibility between P_y and the new user x. This means that, because we cannot tell whom the new user x will like to communicate with, we recommend users who will like to communicate with the new user.

As previously mentioned, because of runtime performance issues, we reduced the number of possible users for new users in a similar way as for users whose preferences are known. However, because we do not have the new user's preferences we assume that the new user will hold similar preferences to his group of peers. We define two users as peers if they are of the same gender, similar age (i.e. maximum of 5 years difference) and live in similar location (i.e. 5 km apart).

If further reductions are needed on the number of users who will be considered for compatibility score calculations, then we can find the set of attributes that are particularly important for each user, and then filter the list using these important attributes. One way of finding important attributes is by comparing the distribution of messages sent by a user x (preferences P_x) with the distribution of messages sent by all users in x's peer group. For instance, if the peers of user Bob mostly send messages to slim women and Bob mostly sends messages to overweight woman, then we can assume that body type is an important attribute in Bob's preferences.

4. EVALUATION

We designed the evaluation of RECON with two main goals. Given the lack of exploration of the notion of reciprocity in reciprocal recommenders, one core goal was to gain understanding of the role of reciprocity in the effectiveness of the algorithm described above. It was also clear that the cold-start problem is so important for this domain that it deserved special attention and analysis. Our evaluation takes into account the fact that users have limited patience when presented with recommendations. So, it is important to know how well RECON performs when it is restricted in the number of recommendations it can offer (or that the user is willing to consider). Accordingly, we report success rates in our evaluation experiments for a range of limits in the number of recommendations.

The data for the evaluation of our algorithm consists of the users' contact history over a six week period. Of those six weeks, the first four are used as training data and the remaining two are used for testing. In addition the users' profile information is used to train the content based recommender and to find matching users. The training set consists of 1.4 million messages sent by over 90,000 users. Note that both the user we are recommending to and any user we are recommending must be in this set of users active during the training period. We report the results for a particular six week period but we repeated the experiment over other periods, with near identical results.

We evaluate our system by measuring the *success rate* of the *known* interactions made during the test period, in the list of top-N recommendations. Success rate of a list of recommendation R given to a user X is defined as the proportion of the *known* interactions (users messaged by x) that were positive (users who replied to x positively:

$$Success(x, R) = \frac{|\{pr : pr \in R, pr\ replied\ positively\ to\ x\}|}{|\{kr : kr \in R, kr \in M_x\}|}$$

We also use *recall*, which in our experiments is defined as the proportion of known positive interactions that were present in the list of recommendations over all positive interactions made by a particular user.

$$Recall(x, R) = \frac{|\{pr : pr \in R, pr\ replied\ positively\ to\ x\}|}{|\{py : py \in M_x, py\ replied\ positively\ to\ x\}|}$$

Success and recall are analogous to the common definition of precision and recall, with the difference that we are only evaluating those recommendations for known interactions between two users. For instance, if we provide a list of five recommendations (top-5) for a user x, and two of these recommendations were users to whom x sent a message, then we evaluate success and recall using those two interactions only. If user x had ten positive replies to his messages in the test period, and only one of those two known interactions was positive, this gives a success of 50% and a recall of 10%.

We understand that evaluating a recommender system based on previous user interactions does not give us the ideal measure of success. A better measure can only be achieved in an evaluation comparing success where users receive recommendations from our recommender against that for a baseline system. However, the evaluation with previous data allows comparison of different techniques using a large sample of users to explore and compare those methods that produce better recommendations. This is particularly true when we want to understand what improvements are gained when different numbers of recommendations are provided.

The results in Figure 3 compare the success rate and recall for a reciprocal recommender (RECON) and for a non reciprocal recommender (RECON without the reciprocal score). This evaluation is also compared to a baseline success measure (17.3%) which reflects the proportion of messages sent by users that get positive replies in the testing set. The baseline measure reflects how well users do, unaided by a recommender, in their search for people who might like them.

Our results indicate that even when we only consider the automatically built intrinsic preferences of the users, we can outperform the users themselves in their search for a successful interaction. We also observe that, for all numbers of top-N recommendations, there is a large improvement in both success rate and recall. For the list of top-10 recommendations, the reciprocal recommender has a success rate of 45.20% while the non-reciprocal recommender has 26.30% success: an improvement of 71.86% due to the reciprocity. Similarly for the top-100 recommendations the addition of reciprocity improved recall by 75.36% (from 6.90% to 12.10%). This supports our hypothesis that reciprocity is important for recommender systems that involve people in both sides of the recommendations (i.e. reciprocal recommenders).

Because we compute a profile model for each individual

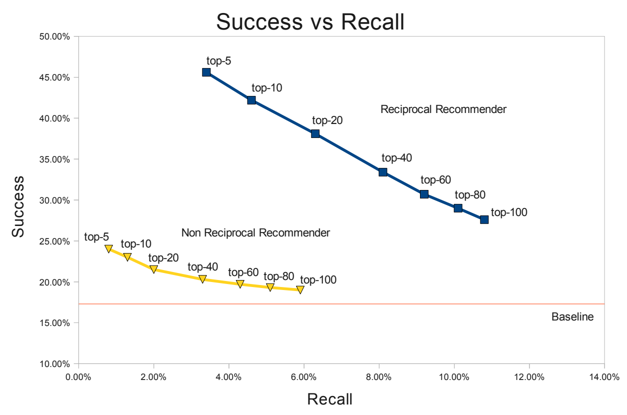

Figure 3: Success rate and Recall comparison between reciprocal recommender, non reciprocal recommender and the current search made by users (baseline)

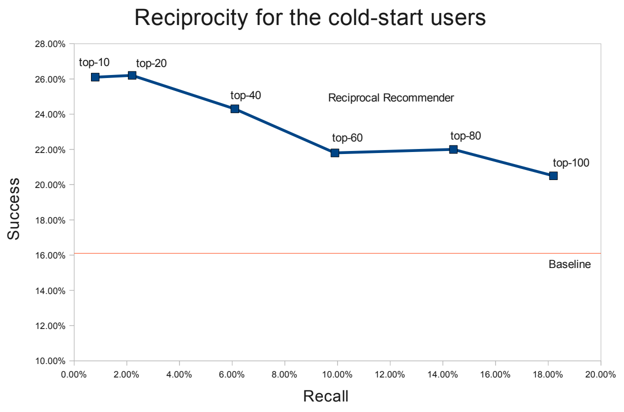

Figure 4: Success comparison between reciprocal recommender and the current search made by new users (baseline)

user, our approach allows us to provide personalised recommendations for each individual user. Nevertheless when we have no information about the user preferences (cold start problem), we can still personalise the user recommendation by assuming the user will equally like all users and finding those who will like him/her back.

We have evaluated the success rate of the communication of new users and we observed that 16.10% of messages sent by them receive positive replies. Therefore, new users have a slightly lower success rate than the other users. We have compared this baseline success rate with the success rate of RECON for the same group of new users, and we have observed that RECON can predict with higher accuracy whether a communication will be successful. In Figure 4, we observe that for the top-10 and top-20, one in four known messages are successful. This performance is achieved by pre-filtering the list of possible recommendations using the preferences of the new user's peer group. This is an assumption that has to be made, otherwise too many users would have to be ranked and the runtime performance of the system would be harmed considerably.

The time required to run RECON depends on the different phases of the algorithm. RECON calculates the preferences for more than 90,000 users (using 1,4 million messages) in about 10 minutes using a computer with two processing cores and 2GB of RAM. Using these preferences, it creates a list of recommendations for all users in about 2 hours. The initial size of the recommendation list varies depending on the peer group of each user: the average size of the recommendations list is 550, with the median 180.

5. CONCLUDING REMARKS

This paper presented the definition and the properties of reciprocal recommender, an important class of recommender system that has received little attention, in spite of many important recommender tasks that require reciprocity. We have developed RECON: a content-based reciprocal recommender for a major online dating website. RECON is able to efficiently provide good recommendations for a large number of users using a real dataset of user interactions. We have evaluated RECON, demonstrating the effect of taking account of reciprocity for a recommender system in a domain such as online dating. Our evaluations have also demonstrated that RECON deals with the important cold-start problem, providing a substantial improvement of more than 60% in success rates for new users, and this effect is particularly strong under a restricted number of recommendations.

We have shown that RECON can effectively select recommendations that have higher success rate than the user themself in the existing system: RECON improves success in 2.6 times given five recommendations. This encouraging result is an excellent foundation for our exploration of approaches to reciprocal recommendation approaches. Our use of historic data has been invaluable for experimentation with approaches. The next important stage is to assess the live performance.

One of the defining features of reciprocal recommenders is the need to ensure that there is a suitable distribution of recommendations across users. For the online dating domain, this is particularly important in avoiding overloading popular users as well as neglecting those who are not popular. And both these are related since ensuring more unpopular people are recommended fits well with making fewer recommendations with the popular. Although such recommender may well reduce the classic recommender success measures, it may be critical to broader success. In this study, we have neither favoured popular nor unpopular users, we believe that reciprocation can inherently distribute the load. The attributes of popular users skew the distribution of preferences towards them (meaning that most users will like someone popular - the essence of their popularity); however, the reciprocation component aims to ensure that the person who is recommended to a user will reciprocate (and this person may well not be popular). Future research will explore this aspect of distribution load for reciprocal recommenders and their effect in the quality of the recommendations.

Reciprocal recommenders represent an important class of recommender systems. RECON represents an exploration of this space with strong results in terms of the contribution of reciprocity and its power to tackle the cold-start problem. There are many potential people-people recommenders, some clearly matching individuals, as in the case of online dating, matching mentors and mentees, providing a useful helper, in contexts such as the workplace or educational settings. There are many other, somewhat less direct cases of reciprocal recommenders, such as job finding where the potential employee is to be matched with a job, or more precisely, an employer. While a job is not a person, it is under the control of the employer, and that involves people judging the suitability of potential employees. While the roles in this case are less symmetric than in online dating, the properties identified for reciprocal recommenders apply. In some systems, the job seeker has the active role, with the potential employer being reactive, as they wait for applications in response to their advertised jobs. In head-hunting systems, the roles are reversed, with the employer actively approaching registered people. This active-reactive aspect of reciprocal recommenders has been significant in our analysis, design and evaluation.

A distinctive feature of our online dating testbed is the richness of information about the users, including substantial useful data from user's answers to closed questions. As we have observed, the nature of reciprocal recommenders makes such explicit profiles an invaluable base. In other reciprocal recommender domains, such as job finding, the user profiles are more diverse as are the job requirements. Therefore, further experiments in reciprocal recommenders involve studies on other reciprocal domains such as employment recommenders and friend search in social networks.

Acknowledgements

This research was funded by the Smart Services Co-operative Research Centre.

6. REFERENCES

[1] X. Amatriain, J. M. Pujol, and N. Oliver. I like it... i like it not: Evaluating user ratings noise in recommender systems. In *UMAP '09*, pages 247–258, Berlin, Heidelberg, 2009. Springer-Verlag.

[2] X. Amatriain, J. M. Pujol, N. Tintarev, and N. Oliver. Rate it again: increasing recommendation accuracy by user re-rating. In *RecSys '09: Proceedings of the third ACM conference on Recommender systems*, pages 173–180, New York, 2009. ACM.

[3] L. Brožovský and V. Petříček. Recommender system for online dating service. *CoRR*, abs/cs/0703042, 2007.

[4] S. Bull, J. E. Greer, G. I. McCalla, L. Kettel, and J. Bowes. User modelling in i-help: What, why, when and how. In M. Bauer, P. J. Gmytrasiewicz, and J. Vassileva, editors, *User Modeling*, volume 2109 of *Lecture Notes in Computer Science*, pages 117–126. Springer, 2001.

[5] R. Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370, 2002.

[6] J. Chen, W. Geyer, C. Dugan, M. Muller, and I. Guy. Make new friends, but keep the old: recommending people on social networking sites. In *CHI '09*, pages 201–210, New York, 2009. ACM.

[7] M. Claypool, P. Le, M. Wased, and D. Brown. Implicit interest indicators. In *IUI '01*, pages 33–40, New York, 2001. ACM.

[8] A. T. Fiore and J. S. Donath. Homophily in online dating: when do you like someone like yourself? In *CHI '05*, pages 1371–1374, New York, 2005. ACM.

[9] J. Freyne, M. Jacovi, I. Guy, and W. Geyer. Increasing engagement through early recommender intervention. In *RecSys '09: Proceedings of the third ACM conference on Recommender systems*, pages 85–92, New York, 2009. ACM.

[10] J. Greer, G. McCalla, J. Collins, V. Kumar, P. Meagher, and J. Vassileva. Supporting peer help and collaboration in distributed workplace environments. *International Journal of Artificial Intelligence in Education*, 9(1998):159–177, 1998.

[11] G. J. Hitsch, A. Hortaçsu, and D. Ariely. What makes you click? - mate preferences in online dating. Quantitative Marketing and Economics (forthcoming), 2009.

[12] D. Kelly and J. Teevan. Implicit feedback for inferring user preference: a bibliography. *SIGIR Forum*, 37(2):18–28, 2003.

[13] Y. S. Kim, B.-J. Yum, J. Song, and S. M. Kim. Development of a recommender system based on navigational and behavioral patterns of customers in e-commerce sites. *Expert Syst. Appl.*, 28(2):381–393, 2005.

[14] J. A. Konstan, B. N. Miller, D. Maltz, J. L. Herlocker, L. R. Gordon, and J. Riedl. Grouplens: applying collaborative filtering to usenet news. *Commun. ACM*, 40(3):77–87, 1997.

[15] J. Malinowski, T. Keim, O. Wendt, and T. Weitzel. Matching people and jobs: A bilateral recommendation approach. In *HICSS*, 2006.

[16] D. W. Oard and J. Kim. Implicit feedback for recommender systems. In *AAAI Workshop on Recommender Systems*, pages 81–83, 1998.

[17] S.-T. Park and W. Chu. Pairwise preference regression for cold-start recommendation. In *RecSys '09: Proceedings of the third ACM conference onRecommender systems*, pages 21–28, New York, 2009. ACM.

[18] Y.-J. Park and A. Tuzhilin. The long tail of recommender systems and how to leverage it. In *RecSys '08: Proceedings of the 2008 ACM conference on Recommender systems*, pages 11–18, New York, 2008. ACM.

[19] M. J. Pazzani. A framework for collaborative, content-based and demographic filtering. *Artif. Intell. Rev.*, 13(5-6):393–408, 1999.

[20] L. Pizzato, T. Rej, T. Chung, K. Yacef, I. Koprinska, and J. Kay. Reciprocal recommenders. Technical Report 651, School of IT, University of Sydney, 2010.

[21] D. Richards, M. Taylor, and P. Busch. Expertise recommendation: A two-way knowledge communication channel. In *ICAS '08*, pages 35–40, Washington, DC, 2008. IEEE Computer Society.

[22] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. Riedl. Applications of dimensionality reduction in recommender systems - - a case study. 2000.

[23] A. I. Schein, A. Popescul, L. H. Ungar, and D. M. Pennock. Methods and metrics for cold-start recommendations. In *SIGIR '02*, pages 253–260, New York, 2002. ACM.

[24] L. Terveen and D. W. McDonald. Social matching: A framework and research agenda. *ACM Trans. Comput.-Hum. Interact.*, 12(3):401–434, 2005.