Recommender Systems for Online Dating

Eric Andrews

Master's Thesis
UNIVERSITY OF HELSINKI
Department of Computer Science

Helsinki, May 19, 2015

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty Faculty of Science	Laitos — Institution — Department Department of Computer Science

Tekijä — Författare — Author
Eric Andrews

Työn nimi — Arbetets titel — Title
Recommender Systems for Online Dating

Oppiaine — Läroämne — Subject
Computer Science

Työn laji — Arbetets art — Level Master's Thesis	Aika — Datum — Month and year May 19, 2015	Sivumäärä — Sidoantal — Number of pages 70

Tiivistelmä — Referat — Abstract

Users of large online dating sites are confronted with vast numbers of candidates to browse through and communicate with. To help them in their endeavor and to cope with information overload, recommender systems can be utilized.

This thesis introduces reciprocal recommender systems that are aimed towards the domain of online dating. An overview of previously developed methods is presented, and five methods are described in detail, one of which is a novel method developed in this thesis. The five methods are evaluated and compared on a historical data set collected from an online dating website operating in Finland. Additionally, factors influencing the design of online dating recommenders are described, and support for these characteristics are derived from our historical data set and previous research on other data sets.

The empirical comparison of the five methods on different recommendation quality criteria shows that no method is overwhelmingly better than the others and that a trade-off need be taken when choosing one for a live system. However, making that trade-off decision is something that warrants future research, as it is not clear how different criteria affect user experience and likelihood of finding a partner in a live online dating context.

ACM Computing Classification System (CCS):
Information systems → Recommender systems
Human-centered computing → Social recommendation

Avainsanat — Nyckelord — Keywords
Recommender Systems, Matchmaking, Online Dating, Reciprocity, Data Mining

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — Övriga uppgifter — Additional information

Contents

1 Introduction **1**

2 Background **2**

2.1 Recommender systems 3

2.2 Reciprocity 5

2.2.1 Online dating 5

2.2.2 Other reciprocal domains 8

3 Reciprocal recommenders **9**

3.1 Overview of approaches 10

3.2 Notation 12

3.3 RECON: content-based approach 13

3.4 RECON with negative preferences 18

3.5 CCR: hybrid approach 20

3.6 Multiple compatible subgroups 25

4 Thompson sampling approach **30**

4.1 Statistical models 31

4.2 First pass: cluster assignment 33

4.3 Second pass: candidate assignment 34

4.4 Considerations 38

5 Data set **39**

5.1 Preprocessing profile data 42

5.2 Exploratory analysis 48

6 Comparative analysis of methods **52**

6.1 Measures of success on historical data 53

6.2 Results 55

6.3 Discussion 57

7 Conclusion **62**

References **64**

1 Introduction

In 2015 in the United States alone, the revenue of the dating business was estimated to be at around $2 billion dollars with an annual growth rate of 5% and with over 3900 companies involved [24]. Every fifth American adult aged 25–34 has tried online dating and 2/3 of them have gone on a date with someone they met online [51]. The largest of online dating sites boast tens of millions of registered users [66], and dating site Match.com self-reportedly has over 2.39 million active subscribers in North America with a revenue of over $850 million in 2014 [41].

Given the massive user bases of these dating sites, there is need and value in helping users combat information overload by filtering the most relevant partner candidates of the abundant pool of choices. Otherwise users have a hard time finding a partner, as they have to browse through and communicate with potentially hundreds, if not even thousands, of users and profiles. While important attributes such as age, location, gender, and relationship preference can be used to cut down on the number of viable candidates, the leftover set may still be huge especially for users living in densely populated areas.

Recommender systems are a means to combat this information overload and provide users with candidate recommendations personalized to their preference and profile. Moreover, there is value in such recommender systems, even for smaller dating sites, because users' attention and time can be devoted to only a handful of choices at a time and poor matches may lead to repeated rejections and discouragement [52].

Recommender systems may also be used to reduce the burden of popular users, who may receive unruly amounts of messages, and relieve the anguish of unpopular users and users who continually get rejected by balancing the way users occur in recommendations.

From the viewpoint of a dating service provider, a recommender has the potential of becoming a part of the core business, if not the core business itself. It could be claimed, perhaps a bit hyperbolically, that Amazon.com is just as much in the retail business as it is in the recommendation business [19]. Many of the largest online services, such as Netflix, YouTube, Facebook, and Spotify, utilize recommender systems in their core products and derive measurable revenue or website traffic from those systems [18, 70].

In this thesis, we introduce recommendation algorithms tailored to the online dating domain and then empirically analyze and compare their performance on a real-world historical data set gathered from an online dating service operating in Finnish markets. The scope of the analysis is limited to offline data. Although much research has been published on evaluating individual recommenders for online dating, there is less published research evaluating and comparing methods across authors.

A further contribution of this thesis is the development of a novel recommendation method for online dating that utilizes multi-armed bandits, Bayesian statistics, and elements of previously published methods. The method attempts to alleviate some of the problems unresolved by previous methods; however, further real-world evaluation and research is required before any strong claims can be made about its effectiveness.

The structure of this thesis is as follows. In Section 2 we give a brief introduction to recommender systems and online dating. In particular, we highlight special characteristics of the dating domain that need be taken into consideration when developing recommenders.

In Section 3 we introduce a special class of recommenders, called reciprocal recommenders, that have been proposed for the online dating domain. We describe four of them in detail, explaining their inner workings, rationale, advantages and disadvantages.

Because the presented existing methods do not handle the cold-start problem satisfactorily, in Section 4, we describe our novel reciprocal recommender, which we refer to as the Thompson sampling approach. It handles cold-start users properly, while still maintaining support for truly personalized recommendations with high coverage.

Section 5 relates to the data set used in the evaluation and comparison of methods. In this section, we explain how the data set was arrived at, how it was divided into training and test sets for evaluation, and how profiles were obtained by means of data preprocessing. The section concludes with exploratory analysis of the data in an attempt to understand its characteristics and lend support to some of the claims made in Section 2.2.1.

Section 6 presents the evaluation and comparison of the reciprocal recommenders. To begin with, the evaluation set-up is described, after which different measures of success are introduced. Then, the performance of each of the methods on different measures are presented, after which a discussion ensues on which method one should choose in what situation.

Section 7 concludes the thesis and provides ideas for future research.

2 Background

In this section, a brief introduction to recommender systems is provided, followed by a discussion of the problem domain under consideration—that is online dating. An especially distinctive quality pertaining to online dating not usually encountered in recommender system literature is reciprocity, which is the need for both parties, the recommendee and the recommended, to share mutual interest in each other. We thus delve into this property, and finally end the section by listing some other domains were this property holds importance.

2.1 Recommender systems

Recommender systems are tools, algorithms, or software that provide users with suggestions, or *recommendations*, of items that may be of use or of interest to them. Depending on the application domain, the items may be anything from books, CDs, movies to services, vacations, or even drug treatments. Recommender systems are designed to help a user make a decision, whether it be buying a book, choosing a movie to watch, renting a car, booking a vacation, or deciding which drug to prescribe a patient [57].

One of the primary motivations for using a recommender system is to overcome *information overload*, which refers to the problem of making a decision in the presence of overwhelmingly many choices or too much information [3, 47, 57]. Recommender systems also have value in guiding inexperienced users to items that are of utility to them [57].

A recommender system can also provide value to a service provider. It can be used to to sell more items, diversify items sold, increase user loyalty, or learn more about customers' preferences [57].

Recommendations are often personalized to the user currently being recommended to [3, 47, 57], i.e., the active user or the recommendee. The emergence of e-commerce sites, for example, has introduced product catalogs consisting of hundreds of thousands or even hundreds of millions (Amazon.com) of items [59]. Because users have nor the time or expertise to sift through such a paramount selection, the need for personalization is apparent to help users find suitable products to purchase [57, 59].

To be able to provide personalized recommendations, a recommender system need understand the preferences of a user. These are collected either *explicitly* by allowing users to give ratings or thumps up and down signals, or *implicitly* by interpreting interest from user actions, e.g., what products they click on in a web store [57]. The more feedback data are amassed on a user, the more accurate a user model can be built for personalization.

To decide which items to recommend to a user, a recommender must be able to predict (or compare) the *utility* that different items have to a user [57]. Given user u and item i, the true utility $R(u, i)$ of the item to the user is estimated by $\hat{R}(u, i)$. In top-k recommendation, given a large set of items $i_1, \ldots, i_N$, the system computes $\hat{R}(u, i_1), \ldots, \hat{R}(u, i_N)$ and provides a list of recommendations $i_{j_1}, \ldots, i_{j_K}$ in descending order of utility, where $k > 0$ is an integer for which $K << N$ [3, 57].

Based on work by Burke [11], there are six recommendation algorithm approaches for predicting utility of items. Out of those six, three are topical to this thesis: content-based recommendation, collaborative filtering, and hybrid recommenders.

Content-based recommenders exploit the items a user has liked in the past to recommend new similar items. Similarity of items is based on comparing associated structured features, such as keyword vectors. A user model is aggregated from features of items a user has expressed interest in, and utility of new items is estimated by calculating similarity to said user model [57].

Collaborative filtering is based on the observation that people often rely on others' opinions when making everyday decisions. Collaborative filtering recommends to an active user items that users with similar taste (neighbors) have liked. Similarity in taste is based on the similarity of rating history, e.g., whether two users have liked and disliked similar items in the past. The assumption is that because two users shared similar taste in the past, they will continue to like and dislike items similarly in the future as well [57].

The advantages of content-based recommenders compared to collaborative filtering are as follows. Firstly, recommendations are based solely on feedback provided by the active user, and no feedback are needed from neighbors. Secondly, it is easy to explain to a user how his or her previous actions have lead to the presented recommendations by comparing features of user's model to recommended items. In collaborative filtering, explaining the causal link is harder because it relates to neighbors. Thirdly, content-based recommenders do not suffer from the *new-item problem* (collaborative filtering methods do), in which a new item cannot be recommended because no one has yet provided feedback on it [57].

One disadvantage of content-based methods is the need for a structured representation of items. Item representations are often limited and sometimes do not contain enough information to discriminate between items of use and no use. For example, a simple keyword vector cannot model the correlations between words. On the other hand, an overly complicated representation will require unruly amounts of data to work reliably. There is also the issue of whether enough information can even be obtained about the items given available resources and time. Collaborative filtering does not require any information about items [57].

Another disadvantage of content-based methods is *overspecialization.* As content-based recommenders are based on recommending items similar to what a user has liked in the past, they tend not to recommend unexpected items unless such considerations are taken into account when developing the methods. Collaborative filtering can recommend unexpected items, provided an active user's neighbors have expressed diverse interests [57].

An issue affecting both collaborative filtering and content-based recommenders is the *cold-start problem* [60], in which a new user cannot be provided accurate recommendations because enough feedback data has not been gathered on them. If a new user is asked to explicitly rate some items or define their user model, then the problem may be somewhat alleviated [20, 57].

Hybrid recommenders combine multiple recommendation approaches in hopes to use aspects of some to address the shortcomings of others, and vice versa. In the context of this thesis, the main focus is on combining collaborative filtering with content-based recommendation to alleviate the cold-start and new-item problems. There are several ways of combining recommenders [57].

Before moving on, it is worth noting that even though in this thesis recommender systems are viewed from an algorithmic point of view, recommender system research as a discipline lies on the crossroads of machine learning, information retrieval, and human-computer interaction. Specifically, the importance of good user experience and user interface design to the perceived trust and credibility of recommendations should not be underestimated [57].

2.2 Reciprocity

Reciprocal refers to something being "given or felt by each toward the other; mutual" [1]. In domains where people are looking for other people, the need to consider the preference of both parties emerges [52]. In traditional recommendation domains, items are inanimate products or services, whereas in people-to-people domains, they have preferences of their own and often limited availability (a person can actively communicate with only so many people at a time) [52].

In this section, we delve into online dating as an example of a reciprocal domain. We start off by describing the typical use case of an online dating service and identify special characteristics that should be taken into consideration when developing a recommender for the domain. One of the most important characteristics is reciprocity [52], for which special consideration need be taken when developing recommenders. The section is concluded by a brief introduction to other domains that exhibit reciprocity.

2.2.1 Online dating

Users sign up to online dating services (often) for the purpose of finding another person to form a relationship with. The relationship types seeked are numerous: short-term to long-term, friendship to romantic. Users are associated with profiles that contain basic information about them and their preferences. To communicate with others, users exchange messages consisting of text and pictures.

A typical (successful) interaction in an online dating website goes as follows.

1. User A registers on the site and fills in his or her profile with information such as: age, gender, height, interests, location, hair color, photo, self-description, preferred age, gender, and so forth.

2. User A searches for interesting profiles, views them, and sends a message to an interesting user B.

3. User B, if active, views the message and the profile of user A and decides whether to respond.

4. If interest is reciprocal, the discussion continues.
5. At some point users A and B may exchange contact details and meet offline, perhaps later on establishing a relationship.

This is just one example of a use case in online dating. Some users wait to receive initiatives instead of actively approaching users themselves, and some may just be interested in viewing profiles without sending any messages.

Typically, a user has to purchase tokens or a subscription in order to send and/or view unconstrained messages albeit some online dating services are free-of-charge to the user and are funded by other means, e.g., advertisement revenues. Many services allow users to send predefined messages without any payment, but in order to exchange contact details, tokens or membership must be acquired [52].

It is important to understand the unique challenges and limitations that online dating poses when designing a suitable recommender system. To that end, let us go through some of these characteristics as originally presented by Pizzato et al. [52, 56] to highlight the difference to traditional product recommendation.

The main differences can be summarized as: reciprocity, existence of rich detailed user profiles, profilic presence of cold-start users, limited availability of users, and the duality of roles a user can take.

Whilst in traditional recommender systems the success of a list of recommendations can be assessed by measuring the active user's response, in reciprocal domains, success can be viewed from three perspectives: the active user, the candidate, and whether interest was mutual for both parties, i.e., reciprocal.

Indeed, as users of online dating are (typically) looking for partners to connect with, satisfying the preferences of both parties is paramount. This implies that recommender systems need to take reciprocity into consideration when, for example, estimating the utility of a recommendation.

In traditional product recommendation, users are rather reluctant to explicitly supply information about their preferences, nor the less their personal information. Sometimes users are not even aware of what their preferences are [4, 9]. However, in online dating, because they want to find a partner and are aware that this depends not only on them, but also on the other party and in providing the other party with a reasonably accurate profile of themselves, users are willing to provide rich detailed profiles about themselves and their preferences. This information is valuable and should be utilized by a recommender to improve recommendation performance.

Regarding user models, compared to traditional product recommendation, where we perhaps have only an implicit knowledge source available, in online dating, several knowledge sources are combined to build a user model. A user *explicitly* provides information about themselves and their preferences [1]. As messaging behavior and/or profile browsing history data of the user is collected over time, an *implicit* source of preference data becomes available. All three sources (the explicit profile, the explicit preferences, and the implicit preferences) can be utilized in building a recommeder's user model. (As a side note, in traditional domains personal information about a user is rarely utilized in recommendation.)

One challenge of online dating has to do with the abundance of cold-start users. In traditional product recommendation, a satisfied user has a long history with the service, as they continually come back to it to receive quality recommendations. If an online dating recommender succeeds in its task, i.e., producing recommendations that lead to relationships, served users will leave the site and may never return. Also dissatisfied users may quit the service early on because of costs and the anguish of not finding a date. Thus for many users, use of the service is short-lived and detailed implicit preferences are not gathered over a long period of time. It is thus imperative that a recommender for online dating be able to handle cold-start users well.

Another unique aspect of online dating is the limitedness of the items (other users). A single user can be in serious contact with only so many different users at one point in time. If a user receives too many initiatives, they may stop replying entirely or start replying negatively to new initiatives. In the worst case, the popular user may become distressed by the amount of messages and quit the service.

On the other hand, if a recommendee receives too many negative responses when sending initiatives to his or her recommendations, it may discourage the user. It is thus vital not to overburden popular users by overrepresenting them in recommendations. This poses a challenge since it is known that, for example, collaborative filtering has a tendency to favor popular items [22, 25]. Limitedness is not such a large problem in traditional product recommendation because the same product can commonly be bought by several people.

Last but not least is the duality of roles a user can take in online dating. In traditional product recommendation, users often take a proactive role, engaging with the system to find new items of interest. In online dating, a user can either be proactive or reactive. Proactive users actively send initiatives to other (recommended) users. Reactive users wait for someone to contact them first. Stereotypically male are proactive and female reactive, but a recommender should take into account that the role of a user may change during the course of time.

When designing a recommender for the dating domain, it is important not to neglect reactive and unpopular users in the presented recommendations because otherwise they will not be found and initiated, leading to

[1]Explicit preferences are not utilized in methods presented in Section 3 and Section 4.

dissatisfaction with service. There may be merit in favoring reactive users over proactive users in recommendations since proactive users will engage in unprompted discussion anyway. Diversity of recommendations is important from the viewpoint of limited availability. In traditional domains, on the other hand, it may be just fine if some products are never present in recommendations.

Some of the points raised here have been empirically validated on an Australian data set by Pizzato et al [52]. They validated that users understand and are aware of reciprocity in their messaging behavior, i.e., they generally take into consideration a candidate's preference before initating paid contact. They also showed that profile completedness affects number of initiatives received, implying that users benefit from spending time on filling in their profile.

Pizzato et al. provide evidence that users' activity in online dating is short-lived; they noticed that 25% of users quit the service 4 weeks after signing up. Regarding proactiveness and reactiveness, Pizzato et al. [52] notice from data that men tend to be more proactive and women reactive, confirming that users take on different roles in online dating. Pizzato et al. also plot success rate as a function of user popularity and notice that indeed, popular users are less likely to respond positively.

In an earlier study by Akehurst et al. [6], it is shown that implicitly gathered preferences are a better predictor of reciprocated initiatives than explicit preferences. Therefore, when enough messaging behavior data are amassed on an individual, it would be beneficial to utilize that data [52]. This discrepancy between implict and explicit profiles may be due to the fact that users tend to have problems formulating their preferences in an accurate manner [7].

In addition to the above studies, in Section 5.2, we perform exploratory analysis on our data set and lend support to some of the characteristics presented in this section.

2.2.2 Other reciprocal domains

There are other domains where reciprocity holds importance although they may exhibit additional considerations of their own that must be taken into account when developing or choosing suitable recommenders. Domains in which reciprocal recommenders seem suitable exhibit the following two properties [52]:

1. The items being recommended are people.

2. Both sides of the recommendation must share interest in each other.

A third property that is not strictly required, but that often crops up when the term reciprocity is used in conjunction with recommender systems, is that there is an asymmetrical relationship between two groups of users.

Take the case of a recruitment service where job seekers are recommended employers or employers are recommended job seekers based on posted job ads and resumes. The two sides play different roles, but both sides are represented by a person. There is some published work [38, 40, 68] on developing reciprocal recommenders for such recruitment services.

Other reciprocal domains that have been mention in passing in previous research are mentor-mentee matching, business partner identification [53], as well as finding flatmates, selecting reviewers for papers, and finding experts and research collaborators [52]. Reciprocal recommenders can also be used to recommend friends on social websites in the likes of Facebook or LinkedIn.

3 Reciprocal recommenders

In reciprocal domains, such as online dating, where the items being recommended are other users with preferences of their own, a *reciprocal recommender* [54] is a recommender that utilizes the preferences of both the recommendee and the candidate when generating recommendations. Other terms that have been used to refer to this class of recommenders are *people-to-people recommenders* [30, 52] and the dating specific, *matchmaking algorithms* [10, 49]. The first of the two is perhaps more commonly used to refer to the type of recommenders found in traditional social network research, for example, for recommending friends on a social website like Facebook.

To start off this section, we present an overview of recommendation methods that have been proposed in the realm of online dating. We then proceed to build up formal notation for our online dating domain so as to avoid repetition and to accurately describe the methods in subsections that follow.

In the latter subsections, considered reciprocal recommendation algorithms are introduced and their implementations carefully explained step-by-step. Some rationale and background is provided as well, along with analysis of advantages and disadvantages on a qualitative level. The empirical comparison of these methods on historical real-world data is presented in Section 6. For a characterization of the data set itself, please refer to Section 5.

It should be noted that the methods presented may contain minor modifications and may not exactly represent those presented in the original research papers. The reason for this is that often specifics such as parameter values or algorithmic details are not covered thoroughly in the original papers, our data set has differences to the articles', or it was found out in our work that some modifications seemed to perform better on our data set. All deviations made consciously are pointed out in the subsections that follow.

The algorithms chosen for deeper study are ones found in literature that the author of this thesis found were well explained and that had some intuitive rationale behind them. As the byproduct of this thesis was an actual prototype recommender system for a commercial dating site, it was deemed important to start with the most established of methods that may yield at least some improvement and that the methods have been empirically validated on real-world data in previous studies.

3.1 Overview of approaches

Most matchmaking systems today deployed in the real-world are proprietary and a level of secrecy surrounds them as they are valued trade secrets. Online dating sites such as Match.com and eHarmony, both powered by recommendation systems, have neither publicly released the details of their algorithms with the exception of some high-level descriptions of their methodologies.

One of the first published studies on applying recommender systems to online dating was by Brozovsky et al. [10] in 2007. They reported on applying the typical user-user and item-item collaborative filtering on a ratings data set from a Czech online dating service where users rate the attractiveness of others on a scale from 1 to 10. They did not consider reciprocity, but did state it as a consideration for future research.

Based on articles searched and collected by the author of this thesis, at around 2010, a surge of publications from different authors appeared on applying recommender systems to online dating [13, 14, 34, 54]. After that, there have been tens of additional publications on the topic.

Kunegis et al. [36], for example, borrowed the concept of split-complex numbers from abstract algebra to model two orthogonal relationship types: like/dislike and similar/dissimilar. They then noticed that split-complex adjacency matrices go hand in hand with singular value decomposition in a way that allows generation of recommendations. The study was performed on ratings data and considered user-user interaction data only.

Diaz et al. [14] approached the problem of recommendation in online dating from an information retrieval perspective, proposing to learn a global reciprocal ranking function (learning to rank). Their method is quite involved containing hundreds of engineered features, gradient boosted decision trees supplied with data preprocessed by logistic regression, and using regular expressions on message bodies to capture whether contact information such as phone numbers or email addresses have been exchanged between users.

Multiple hybrid and content-boosted recommenders for online dating are presented by Kim et al. [28]. They especially focus on the acute problem of cold-start users and based on empirical evaluation conclude that two methods rise above the rest. The first one follows the same idea as CCR (Section 3.5), and the second utilizes compatible subgroup rules (Section 3.6) to seed and generate the input for collaborative filtering. The latter seems to perform slightly better on cold-start users but is conceptually and computationally rather complex.

Much work on recommenders for online dating has been authored by a research group from the University of Sydney led by Luiz Pizzato. For example, the methods RECON and CCR presented later in this section are by this group. Another interesting method they have proposed is a stochastic matching approach [56], which addresses the known bias that recommender systems have towards popular items [22, 25]. The idea is to match everyone with someone and avoid overburdening popular users by ensuring that a user receives the same number of recommendations as they have been recommended to others. Given a list of user pairs and probabilities that each will have a successful relationship, the task then becomes to optimize the total number of successful matches by selecting an appropriate combination of pairs.

Some authors have approached the problem of recommendation in online dating from an explicitly graph-centric viewpoint: mapping users to nodes and messages as edges in-between. BehvPred takes into consideration not only messages but also visits to user profiles to form such edges [63]. The resulting (gender-)bipartite graph is then partitioned into 8 loose groups, users are fuzzily assigned to them, and the collective behavior of the groups are used to generate recommendations.

Kutty et al. [37] also interpreted messaging behavior as a bipartite graph, but in addition attached user profile information to nodes. The resulting "attributed bipartite graph" was then used as the basis for developing a rather involved match-making procedure that utilizes concepts from social network research and graph theory.

Another graph-based method, MEET [38], models users as nodes but instead of messages, uses user profile data to form the weighted edges of a bipartite graph. Weights are calculated using two-way relevance based on similarity of user profiles. The graph is then partitioned into specialized subgraphs for computational scalability using co-clustering methods. Each subgraph is further refined by building a second set of edges based on messages and attaching user availability information to nodes. Recommendations are produced by performing graph inference on the local subgraphs.

Numerous variations of collaborative filtering to the reciprocal domain of online dating have been proposed as well [34, 67, 69]. An especially interesting discovery was made by Xia et al. [67], who noticed that the recommenders they tested performed differently depending on the gender of the recommendee. In particular, men benefited from a reciprocal recommender more than women hinting that women may be more conscientious towards the preferences of their candidates than men. Interestingly, they also noticed that collaborative filtering seemed to beat content-based recommendations by a long shot.

One recurrent trade-off that keeps coming up in the description of these methods is the degree to which user profiles versus interaction history (e.g., messages) should be used to calculate similarity between users or interest towards users [67]. Data sparsity is a concern when utilizing interactions, but user profiles may not always be accurate because they are manually input by users themselves. Lack of self-awareness or desire to have a more attractive profile affect accuracy of profiles [52]. Indeed, there are known behavioral factors that cause people to misrepresent themselves in online dating profiles, e.g., men overestimating their height and women underestimating their weight [23].

3.2 Notation

The notation presented is partly based on work by Pizzato et al. [53] with additional concepts to cater for the multitude of methods presented. The two important broad level concepts here are user profiles and messaging history which we formally define in that order.

Let A be the set of profile attributes such as age, gender, height, have children, want children, eye color, hair color, religion, etc. Each attribute is associated with either a discrete or a continuous set of values, e.g., $a_{\text{hair color}} = \{"blonde", "brown", "gray", "black"\}$ or $a_{\text{like arts}} \in [0, 1]$. Most of the attributes (described in detail later, in Table 5 and Table 6) are of discrete nature, i.e., countable and finite. These attributes are also referred to as *nominal* or *categorical* in statistical parlance.

Let x be a user. Then the profile of user x is represented as

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

where v_a is the value of attribute $a \in A$ for user x. When the situation demands, $U_{x,a}$ is used to refer to the value of attribute a for the profile of user x.

Alongside profiles we also have historical messaging data. The first message sent by a user to another user is called an *initiative*. There may exist a maximum of one initiative between two users. Hence one of them has to be the *initiator*.

Denote by $M_{x,*}$ the set of users user x has sent an initiative to. Similarly, denote by $M_{*,x}$ the set of users who have sent an initiative to user x.

An initiative will receive either a *positive*, a *negative*, or *no response* from the recipient. To differentiate between positive, negative, and no response messages, we use superscript symbols $+$, $-$, and *null*, respectively. For example, $M^{+}_{x,*}$ is the set of users that have responded positively (*reciprocated*) to the initiative of user x, and $M^{-}_{*,x}$ is the set of users that have sent an initiative to x but to whom user x has responded negatively (*rejected*).

A recommendation algorithm produces a list of *candidates* as output for each active user. That is an ordered list of candidate users R_x for each recommendee x. Terms *recommendee*, *active user*, and *target user* are used interchangeably to refer to the user currently being recommended to. Recommendation list size may vary, and as is shown in Section 6, the performance of methods do vary by list size. A recommendation list will

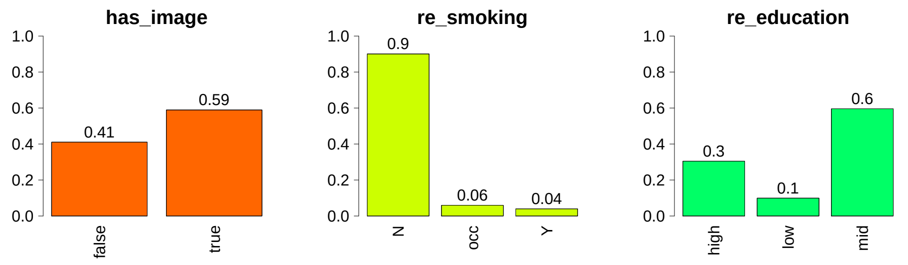


Figure 1: An example of a user preference. For convenience distributions for only 3 attributes are shown.

never contain the active user itself ($x \notin R_x$) nor will it contain users that the active user has had previous interactions with, in other words, it holds that $R_x \cap (M_{x,*} \cup M_{*,x}) = \emptyset$.

3.3 RECON: content-based approach

RECON [53] is a reciprocal content-based recommender that utilizes messaging behavior and profile attributes to calculate recommendations. The main idea is that for each user we build a distribution per profile attribute in order to model the preferences of the user. These distributions are calculated and aggregated based on profiles the active user has indicated interest towards, based on his or her messaging behavior. Having the preferences of a user captured as profile attribute distributions, we can then easily calculate the compatibility between the user's preferences and unmet users' profiles. These compatibility scores then serve as the basis for making recommendations.

Being a content-based recommender, RECON utilizes the past behavior of a user to provide individual recommendations for them. Falling back to the traditional user-item terminology used in recommendation system research, the recommender analyzes the items (other users) the user has previously interacted with and then builds a structured model of the items' contents in a hope to capture the preferences of the user. The recommendation process then simply matches the contents of new items to the model. This results in a numerical score per new item that can be used to predict the level of interest a user may have in said item. When the built model is accurate, this method can work very well [57].

Taking a high-level view of the recommendation process of RECON, we can identify three separate steps: 1) build a preference model for each user; 2) use the preference model of each user to calculate similarity scores with unmet users; 3) finally, for each user, create a ranked recommendation list based on the compatibility scores calculated in the previous step.

The algorithm begins by building a *preference model* for each user based on their historical messaging behavior. The output is a set of categorical distributions, one for each attribute, as exemplified in the bar plots of Figure 1. The figure is essentially saying that out of previous messaging partners this user has expressed interest in, 59% had a profile image, a majority of 90% were not smokers, and that the candidates have tended towards people with a mid or high level of education. Another way of thinking about these distributions is that they capture the preferences of the user. In this case, the user is not that picky about whether a candidate has a profile image or not, but the user will likely not prefer smokers.

The technical procedure for building these user preferences is outlined in Algorithm 1. The first step is to find out which users the active user x has expressed interest in. These are not only users $M_{x,*}$ that x has sent initiatives to but also users $M^{+}_{*,x}$, the users whose initiatives x has responded positively to. The union of these two sets, denoted H, is then the base set from which the preference model is constructed.

Algorithm 1 RECON: Building a preference model for a user

```
Require: user $x$ has expressed interest in at least one user. ($H \neq \emptyset$)

1: function BuildPreferenceModel(User $x$)
2:     $H \leftarrow M_{x,*} \cup M^{+}_{*,x}$
3:     for each discrete attribute $a \in A$ do
4:         for each value $v \in a$ do
5:             $p_{a,v} \leftarrow$ count of users in $H$ with attribute $a = v$.
6:             $p_{a,v} \leftarrow \frac{p_{a,v}}{|H|}$
7:     for each continuous attribute $a \in A$ do
8:         $p_a \leftarrow$ mean of value of attribute $a$ for users in $H$.
9:     return $p$
```

Building up the preference model is a two-stage process. First of all, for discrete attributes, we enumerate through all attribute-value pairs calculating the occurrences of each in the profiles of users in H. We then divide each count with the number of users in H. This effectively normalizes the counts by attribute so that for each attribute $a \in A$: $\sum_{v \in a} v_a = 1$, meaning that counts becomes proportions.

The second stage involves continuous attributes in the range of $[0, 1]$. This is our own extension to RECON, as the original research by Pizzato et al. [53] addressed only discrete-valued attributes. For each continuous attribute, we simply take the mean value over all users H.

The result of this procedure is a set of categorical distributions as illustrated in Figure 1. In the case of continuous attributes, the distributions break down essentially to Bernoulli distributions because there is only one category with parameter $p \in [0, 1]$.

The next step of RECON involves calculating compatibility scores. Let us jump right in with an example. Suppose there were only the 3 attributes as illustrated in Figure 1 and that the presented example distributions were for a user named `Mike`. Say we also have user `Anjelica` with an image, who does not smoke, and who has a mid-level education. Calculating the compatibility of `Mike` with `Anjelica` involves looking at `Mike`'s distribution for each attribute and in each case, picking the value that matches with `Anjelica`'s profile. The proportions of these values are summed and divided by the number of attributes to get a value between $[0, 1]$. The calculation boils down to

$$\mathit{Compatibility}(\texttt{Mike}, \texttt{Anjelica}) = \frac{0.59 + 0.90 + 0.60}{3} \approx 0.697.$$

For the case of continuous attributes, which are assumed to be in $[0, 1]$, multiplication is used. So say that in addition to the 3 attributes presented in Figure 1, there is a fourth (continuous) attribute "like arts" that indicates the level of interest a user has towards arts. If the preference model ($p_{\mathit{like\ arts}}$) of user `Mike` has a value of 0.8 and the profile of `Anjelica` a value of 0.5, then the term 0.5×0.8 is added to the numerator of the equation above, and the denominator is incremented to be 4.

The actual method for calculating compatibility scores has a few additional corner cases and is presented in Algorithm 2. Firstly, if the active user has no preference model, we default to returning 0.001. The reasoning will become apparent in subsequent paragraphs. Suffice to say that in this situation, we do not have a user preference because the user has not sent or responded positively to any messages. Therefore we have nothing to work with to generate scores. Returning a small constant means, as we will soon see, that when recommending to the "preferenceless" user, we only consider the preferences of the candidates.

The second edge case on lines 11-12 is when a value has zero occurrences in an attribute's distribution. Such is the case, for example, when a compatibility score is calculated between two male users where the user of the first operand has not had historical messaging activity with males, in other words, his behavior has lead us to deduce that he is strictly heterosexual.[2]

The attentive reader may have noticed that the compatibility score algorithm just introduced calculates a one-way compatibility: the compatibility that `Mike` would like `Anjelica`. But as we are discussing reciprocal recommenders, the degree to which `Anjelica` likes `Mike` is also important. Indeed, RECON considers this by calculating compatibility both ways and combining them by their harmonic mean:

[2]As a side note, in a real-world production system one should probably rely on the explicitly stated sexual preference of a user, as automatically inferring this and generating recommendations based on this could be a bit intrusive from a user experience standpoint.

Algorithm 2 RECON: Calculating one-way compatibility (adapted) [53]

```
1: function COMPATIBILITY(User $x$, User $y$)
2:     if user $x$ has no preference model then
3:         return 0.001
4:     else
5:         $s \leftarrow 0$
6:         $p \leftarrow$ Fetch preference model of user $x$
7:         $U_y \leftarrow$ Extract profile of user $y$
8:         for each discrete attribute $a \in A$ do
9:             $v \leftarrow$ value of $a$ in profile $U_y$
10:            $q \leftarrow p_{a,v} \in [0,1]$
11:            if $q = 0$ then
12:                return 0
13:            else
14:                $s \leftarrow s + q$
15:        for each continuous attribute $a \in A$ do
16:            $r \leftarrow$ value of $a$ in profile $U_y$ (in $[0,1]$)
17:            $q \leftarrow p_a \in [0,1]$
18:            $s \leftarrow s + qr$
19:    return $s$ / $|A|$
```

$$\frac{2}{\mathit{Compatibility}\,(\texttt{Mike},\texttt{Anjelica})^{-1} + \mathit{Compatibility}\,(\texttt{Anjelica},\texttt{Mike})^{-1}}.$$

Why the harmonic mean? The harmonic mean is biased towards smaller values. If `Mike` does not match `Anjelica`'s preference well, for example, if Anjelica prefers highly educated men that have profile images set up but Mike has neither a profile image or is highly educated, the compatibility score may be, say 0.1. Then the harmonic mean between the users will be around 0.175 while the traditional arithmetic mean will be 0.399.

Pizzato et al. [52] point out that intuitively it makes sense not to assign large scores to user pairs whose preference towards each other differ considerably because both users' preferences matter when looking for a partner. They also briefly allude to their experience according to which best results were obtained when two-way scores were in-between but biased towards lower values. Empirical data for this claim, however, is not presented.

Having discussed how to build preference models and how to calculate two-way scores between pairs of users, we are now ready to explore RECON's main procedure, Algorithm 3. This algorithm produces the desired list of recommended candidates. It is presented from the viewpoint of a single user; for a real implementation, one may want to calculate the recommendations for all users in one-pass in order to avoid the overhead of building preference models multiple times.

Algorithm 3 RECON: The main procedure, adapted from [53]

```
function RECON(User $x$, Recommendation list size $N$)
    $p^x \leftarrow$ BuildPreferenceModel($x$)
    $R \leftarrow (M_{x,*} \cup M_{*,x})^\complement$                ▷ unmet users
    $s_y \leftarrow 0 \ \forall y \in R$                              ▷ two-way scores
    for each candidate $y \in R$ do
        $a \leftarrow$ Compatibility($x$, $y$)
        if $a > 0$ then
            $p^y \leftarrow$ BuildPreferenceModel($y$)
            $b \leftarrow$ Compatibility($y$, $x$)
            $s_y \leftarrow \frac{2}{a^{-1}+b^{-1}}$                  ▷ harmonic mean
    $R \leftarrow$ descending list $\langle y_1, y_2, ..., y_j \rangle$ where $s_{y_i} \geq s_{y_{i+1}} \forall i$
    return $N$ first elements of $R$
```

The algorithm starts off by building a preference model for active user x, finding all users who have not communicated with x, and initializing all two-way scores to be zero. It then iterates through the candidates, calculating a two-way score between each and user x. Finally, the candidates are sorted in descending order by their two-way score and the first N candidates are returned.

The advantages of RECON are that it is conceptually quite simple and it provides highly personalized recommendations based only on the interaction history of the active user. This means that it is able to adjust to individual tastes that may differ considerably from the typical preferences of a user's cohort. An additional advantage of RECON is that it supports homosexual and bisexual preferences out of the box. Even proportions between genders are modeled in the preference model.

Although the necessity to address cold-start users is conveyed multiple times in the article introducing RECON [53], a major drawback of the method has to do with them. RECON is able to provide new and reactive users with recommendations well enough using only the one-way preferences of the candidates. It does not, however, fair so well in recommending cold-start users to old users because the 0.001 score returned for them will result in them being situated at the back of recommendation lists. The case is even worse when recommending cold-start users to cold-start users because all two-way scores will be 0.001, providing no mechanism for discriminating between matches. A large portion of online dating users are cold-start users, making the problem prominent.

A second drawback has to do with attributes. Each of them contribute equally to the final score so their relative importance to each other should be more or less the same. Correlations between attributes may cause latent features to be scored multiple times. For example, attributes $a_{have\ children}$ and a_{age} are presumably quite correlated.

Another headache may be caused by attributes whose distributions are highly skewed. As will be discussed in Section 5.1, left-out attribute "country" had value "Finland" as its overwhelming majority. Because of sheer numbers, most users would attain a distribution for country that is also highly-skewed towards Finland. This does not, however, necessarily mean that they prefer people who live in Finland, and this could strip them from seeing some interesting candidates living abroad.

These attribute-related problems may be alleviated by careful feature engineering. The employed solution for solving the skewness problem in this thesis was to just leave out problematic attributes as is discussed in detail in Section 5.1. Some correlation was allowed between attributes as it did not seem to hurt performance, but this is something that could be studied more rigorously.

A weighting scheme for the attributes could also be considered. Park [49] studied a preference model for online dating similar to RECON where attributes are initially weighted by user-defined weights. After gathering messaging behavior, the weights are slowly tuned towards the implicit preferences of the user by learning a logistic regression model and transforming its weights. However, to the author of this thesis, it was left unclear on what theoretical basis those transformations were made. Moreover, they only tested their method on simulated artificial data.

Regarding the problem of cold-start users, it was suggested in later work that a weighted harmonic mean could be used in Algorithm 3 with zero weights for users without preferences [52]. We tested this strategy, but it did not lead to better success rate or recall (defined in Section 6.1).

3.4 RECON with negative preferences

We also implemented an extension of RECON that alongside positive preference also maintains a negative preference model [55]. This negative preference model aggregates the attribute values of senders whose initiatives the target user has rejected (responded negatively to). For example, if a user is very strict about preferring only non-smokers, this may be reflected in his or her negative preference which in turn reduces the compatibility score with smokers. But only if the user has behaved so in their messaging history.

Adaptation of negative preferences is quite straightforward. First of all, to build the negative preference model, we change line 2 of Algorithm 1 to use base set $H = M^-_{*,x}$. These are users whose initiatives user x has rejected. This change provides us with a negative preference model that we distinguish from the positive p with an overline $\overline{p}$. Moreover, we can make use of the same compatibility score calculation as presented in Algorithm 2 but instead use the negative preference model $\overline{p}$ in place of the positive one.

The main algorithm of RECON, Algorithm 3, needs to be modified so that the calculation of variables a and b combines the scores obtained with preference models p and $\overline{p}$. Let us abbreviate the original compatibility function COMPATIBILITY to C^+. When instead used with negative model $\overline{p}$, let us denote it as C^-. The following formula introduced by Pizzato et al. [55] is used to calculate the *combined compatibility score* between two users x and y:

$$C^{\pm}(x,y) = \frac{1}{2}\Big[C^+(x,y) + (1 - C^-(x,y))\Big] \in [0,1]$$

The above formula takes the average of two scores, the first of which is the same score used in basic RECON, which measures how well user y matches the positive preference of user x. The second score measures how much user y matches the negative preference of user x. The better the match the closer $C^-(x,y)$ is to one, hence bringing the term $1 - C^-(x,y)$ closer to zero. This of course, in turn, lowers the value of the combined score $C^{\pm}(x,y)$.

Broadly speaking, a combined score near 1 indicates that a candidate matches the positive preference of a user well, but differs quite a bit from the negative preference. Scores of around 0.5 indicate that a candidate matches the positive preference well, as well as the negative. Values close to zero indicate that positive preference is not matched at all, worse yet, the negative preference is matched very well.

As a side note and just for clarification, the combined scores are calculated before the two-way scores. That is to say the harmonic mean is applied to the combined compatibility scores $C^{\pm}$ in Algorithm 3.

The rationale of using RECON with combined preferences instead of just positive preferences is two-fold. First and foremost, in the empirical evaluation conducted by Pizzato et al. [55], it was found that extending RECON with negative preferences resulted in statistically significant improvements of success rate (Section 6.1) for recommendation lists of size 1 and 5. Secondly, accounting for negative preferences resulted in a sharp drop in failure rate (Section 6.1) for small list sizes below 40. In other words, the chance of receiving a rejection from sending an initiative to a candidate in the recommendation list was lowered.

On a qualitative level, RECON with negatives serves to reduce, as put eloquently by Pizzato et al. [55], "the anguish of repeated rejection". Being repeatedly rejected or overburdening popular users with messages may lead to churn [55]. It is also valuable to discard undesirable recommendations from a recommendee's list of candidates, as this increases the utility of the recommendations. In some cases, there may be no good recommendations available for a preference; then at least reducing the risk of rejection may be valuable [55]. Lastly, making use of rejections means utilizing more available data, which might have rippling effects on success rate.

One idea for improving the recommendation performance of RECON with negatives would be to weight the terms C^+ and C^- of the combined compatibility score according to the amount of initiatives and replies available. In other words, if for a single user we had 25 positive replies and initiatives available, and 2 negative replies available, we would weight C^+ higher than C^- to account for the fact that the positive preference model is based on more data and is thus likely more accurate.

3.5 CCR: hybrid approach

Content-Collaborative Reciprocal (CCR) [5] is a hybrid approach to reciprocal recommendation that combines ideas of content-based recommenders and collaborative filtering to especially address challenges introduced by cold-start users. As discussed earlier in Section 2.2, cold-start users constitute a large portion of the user base in online dating.

The recommender is composed of three high-level steps:

1. Given a user x, find the set of similar users S_x, or *neighbors*, by comparing selected profile attributes.

2. For each neighbor s in S_x, find the users they have had reciprocal interest in (i.e., the set $M^+_{s,*} \cup M^+_{*,s}$). All these users form the candidate set C_x.

3. Rank the candidates in C_x by tallying their positive and negative interactions with users in S_x.

The content-based part of this hybrid recommender is Step 1 in which we find similar users. This involves comparing the profile of the recommendee to other users. Steps 2 and 3 are based on collaborative filtering: we find candidate users based on neighbors' interactions and then rank them based on the collective behavior of the neighbors. Conceptually, CCR is rather simple, but as usual, the technical details contain a few complexities.

There is one considerable assumption in this method that stems from its reliance on collective behavior. This assumption is that similar people like and dislike similar people and are liked and disliked by similar people. Take the case of user `Mike` who likes `Anjelica` and `Ruby` but dislikes `Jill`. On the other side, `Anjelica` dislikes `Mike` but `Ruby` and `Jill` both like him. If we deem new user `Bob` similar to `Mike` (e.g., same age range, location and body type), we could make an educated guess that the above relations may apply to new user `Bob` as well.

The aforementioned hypothesis was empirically validated by Akehurst et al. [5] on a real-world online dating data set. First of all, they established 5 conceptual interaction groups that relate to a user x. These are x likes, x is liked by, x dislikes, x is disliked by, and x is reciprocally liked by. They then performed a correlation analysis to determine the relationship between profile similarity and users' interaction groups given two users a and b. To make a

Attribute distance	Ordering
D_{height}	<155 cm, 156–160 cm, ..., >200 cm
$D_{education\ level}$	low, mid, high
$D_{smoking}$	no, occasionally, yes
$D_{have\ children}$	no, yes

Table 1: Ordering of (some of) the attributes considered in the distance function D.

long story short, they observed moderate correlations (around 0.5) between profile similarity and similarity of users' interaction groups. Coefficients of 0.5 are considered reasonably high in studies involving humans [5].

Having discussed the rationale, let us now delve into the technical details of CCR. The steps of the algorithm are presented in the logical order they were outlined earlier, and with that in mind, let us begin with Step 1.

The first step of the algorithm involves finding a set of similar users S_x to the active user x. To determine similarity between two users, we need a similarity function or its inverse, a distance function. The distance function D presented here is a variation of the one presented by Akehurst et al. [5], as their profile attributes were slightly different and they did not consider location in their study. The selected attributes reflect those used in the original study with the additional inclusion of the location attribute.

The profile attributes used by the distance metric are age, height, body type, education level, smoking, have children, and location. The distance function D is composed of a set of *attribute distance functions* and it is simply the sum of them (with a multiplier for age because of its importance):

$$\begin{aligned} D(x,y) = \ & 2D_{age}(x,y) + D_{height}(x,y) \\ & + D_{body\ type}(x,y) + D_{education\ level}(x,y) \\ & + D_{smoking}(x,y) + D_{have\ children}(x,y) + D_{location}(x,y) \end{aligned}$$

Each attribute distance returns 0, 1, or 2. If either x or y has missing value n/a then 2 is always returned. D_{age} is 0 when the absolute difference between ages is 0-5 units, 1 when the difference is 5-10 units, and 2 otherwise. $D_{location}$ is the minimum number of adjacent territories that need to be traversed to get from the location of user x to the location of user y. The territories are shown in the right-hand side of Figure 5, and the maximum value is 2 even if sometimes 3 regions need to be crossed.

For each of the nominal attributes listed in Table 1, an ordering is defined, and then distance is defined as distance between values in the list.

Again, the maximum distance is capped to 2. For example, one user may have height in the range of 156–160 cm while another user is >200 cm. Although their distance in the ordered list of values is greater than two, the attribute distance value used is 2.

Finally, we have $D_{body\ type}$. This is rather tricky to define because there is no inherent or naturally occurring ordering that one could resort to. The following order is used for the three values: 'slim', 'average', 'stocky OR full figured'. The missing value 'muscular OR athletic' is defined to have distance of 0 with itself, 1 with slim and average, and 2 otherwise.

With the attribute distance functions defined, it is now easy to see that the codomain of function D is $\{0, 1, 2, \ldots, 16\}$. The lower the value, the more similar two users are to each other.

Whether or not this is the best way to define similarity/distance is open to further investigation. Our implementation follows, quite closely given our data set, what was presented by Akehurst et al. [5]. They had conducted data analysis to figure out 7 attributes that should be used. Out of those 7, this study used 6, leaving out civil status which was found to be rather age correlated and skewed. Likewise the inclusion of location was another necessary deviation, as the original study considered only users living in the same area.

Given the distance function D, similarity between active user x and all other users of the same gender is calculated. The other users are grouped by their distance to x resulting in a maximum of 16 different groups. This more or less concludes Step 1.

Step 2, the generation of candidates, is outlined in Algorithm 4. The procedure is given current user x, neighbors S_x grouped by distance (as calculated in Step 1), and two integer parameter k and c. Parameter k controls the minimum number of neighbors to analyze, and c is the minimum desired number of unique candidates to collect. (Note, however, that this does not imply that c or more candidates will necessarily be returned. This is because the main loop may terminate prematurely if neighbors are run out of, i.e., $d > 16$.)

The procedure goes through the neighbors S_x in order of distance d such that at each iteration a random neighbor s with distance d is chosen and removed from S_x. If there are no such neighbors, d is incremented until there are. All users neighbor s has had positive interactions with, i.e., users of set $M^+_{s,*} \cup M^+_{*,s}$, are added to set of viable candidates C_x, provided active user x has not had previous interactions with them (hence the intersection with R on Line 12). This process in continued until k neighbors have been evaluated and c unique candidates have been gathered, or, there are no more neighbors left.

The double constraint involving integers k and c is in place to ensure two things. First of all, to make sure that enough candidates are accumulated, and secondly, to make sure that enough neighbors are considered because otherwise messaging behavior of highly-active users may dominate the generated candidate set [5].

The third and final step of the algorithm involves ranking the unmet candidates C_x according to their negative and positive interactions with

Algorithm 4 CCR: Candidate generation bounded by a double constraint

```
1:  function GenerateCandidates(user $x$, neighbors $S_x$, $k \in \mathbb{N}$, $c \in \mathbb{N}$)
2:      $R \leftarrow (M_{x,*} \cup M_{*,x})^{\complement}$                    ▷ unmet users
3:      $C_x \leftarrow \emptyset$                                          ▷ candidates
4:      $d \leftarrow 0$
5:      repeat
6:          if $S_x$ does not contain a neighbor with distance $d$ then
7:              increment $d$ until that is the case or $d > 16$.
8:              if $d > 16$ then
9:                  break loop
10:         $s \leftarrow$ choose a random neighbor from $S_x$ with distance $d$
11:         remove $s$ from $S_x$
12:         $C_x \leftarrow C_x \cup (R \cap (M^+_{s,*} \cup M^+_{*,s}))$
13:     until at least $k$ neighbors evaluated and $|C_x| \geq c$
14:     return candidates $C_x$
```

users L, which we denote to stand for the set of neighbors that were popped from S_x during Algorithm 4. For each candidate $c \in C_x$, we calculate the level of *support*:

$$Support(c, L) = \sum_{s \in L} \mathbb{I}(c \in M^+_{s,*} \cup M^+_{*,s}) - \sum_{s \in L} \mathbb{I}(c \in M^-_{s,*} \cup M^-_{*,s}),$$

$\mathbb{I}(A)$ is an indicator variable taking value 1 when A is true and 0 otherwise. The above formula, for a candidate c, counts the number of times c has responded positively to or has received a positive reply from a user in S_x. It then subtracts the number of times c has responded negatively to or has received a negative reply from a user in S_x. The resulting value indicates how reciprocally liked candidate c is with users S_x.

The final list of recommendations is obtained by sorting the candidates in descending order of support and cutting the list to the desired list size N—assuming there even are that many candidates. In our implementation, ties are broken arbitrarily.

The ranking method ensures that if a candidate c liked and was liked collectively by similar users to x, then that candidate should be put higher in the recommendation list for x. This intuitively seems to make sense. Another benefit of ranking by support is that it alleviates the burden of popular users who tend to respond negatively more often thereby pushing their support down [5]. This in turn also reduces the chance of rejection from the recommendee's perspective.

An alternative ranking measure for CCR was proposed in a later study by Akehurst et al. [6], in which they studied the discrepancy between explicitly stated preferences and implicit preferences gathered from user behavior data. They proposed that for each recommendee x, a NBTree model [31] (a decision tree with naive Bayes models on its leaves) be learned to model the preferences of the user—in a similar fashion as was done in RECON. The learned preference models would then be used to assign probabilities to candidates for the ranking phase, effectively replacing Step 3 with a content-based method, and thereby leaving Step 2 to be the only one to utilize collaborative filtering.

Yet in the results of their study, they found the NBTree ranking to be no better than the original support-based ranking [6]. As a matter of fact, their main contribution was to show that explicit preferences are not as good predictors of messaging success as implicit ones [6]. Based on these facts, time was not invested during this thesis on implementing the alternative ranking procedure. However, it does seem like an interesting idea for further study.

Regarding Algorithm 4, the parameters used in the original study were $k = 100$ and $c = 250$. We found $k = 400$ and $c = 250$ to work better for our data. Different choices of sensible values did not impact performance in any major way, but an extensive study of the effect of combinations of parameter values was not conducted.

A disadvantage of CCR compared to RECON is that it does not support other gender preferences than heterosexual. Modifications to choosing neighbors and candidates would have to be made to support homosexual or bisexual preferences.

Like RECON, CCR is able to immediately provide recommendations to a cold-start user based on their profile. However, it uses collective behavior of neighbors instead of one-way profile preference to accomplish this which may yield more reciprocal recommendations. This ensures that new (proactive) users receive quality recommendations from the get-go.

That being said, CCR cannot (similar to RECON) effectively recommend a cold-start user to other users because of lack of interactions. Specifically speaking, because a cold-start user y has had no or few interactions, when gathering neighbors for a recommendee x, the probability of y having had an interaction with a neighbor of x is small, and in the case of user y having no interaction history, the probability is zero implying that there is no way user y could appear in the recommendations for active user x.

As personal messaging data are amassed on an active user, the method still provides recommendations based solely on neighbors' interactions. Therefore the method is unable to provide personalized recommendations on an individual level.

An advantage of CCR stems from its ability to utilize messaging data across similar users to generate recommendations. Moreover, the reliance on profile attributes is weaker than in RECON because they are only taken advantage of when calculating distances whereas RECON extensively utilizes profile attributes.

Another method called SIM-CF developed independently by Kim et al. [28] follows the same basic idea as CCR. Interestingly, it was found out in their study that this simple hybrid approach works pretty well despite its relative simplicity.

3.6 Multiple compatible subgroups

Multiple compatible subgroups [30] (earlier work [29]) is based on the idea of using profile attributes to partition the user base into subgroups. Given an attribute, it is possible to divide the male and female users into subgroups by attribute value and then measure the reciprocal interest each subgroup of one gender has towards the subgroups of the other. These subgroup relations are then encoded as rules which are attached to users for the eventual generation of recommendations.

The method presented here is based on the premise that similar people have similar preferences. In sociological studies concerning homophily, it has been recorded that people with similar traits, forming subgroups, tend to have similar preferences as well [43]. An empirical correlation study by Akehurst et al. [5] summarized in Section 3.5 on CCR also lends support to this theory.

To begin the exposition, we define notation pertaining to multiple compatible subgroups, after which we will go through the technical details of the actual recommendation process, which is rather involved.

A *subgroup* is defined as a set of users of same gender with identical values for a set of attributes. Take for example subgroup A that contains men aged 28–32 that are highly educated and occasionally smoke. Subgroup B containing females aged 23–27 who are also highly educated and occasional smokers may be a *compatible subgroup*, which loosely means that A is more interested in B compared to other subgroups.

A (multiple) *compatible subgroup rule*,

$$a_1 = v_1 \wedge a_2 = v_2 \wedge \cdots \wedge a_m = v_m \Rightarrow a_1 \in W_1 \wedge a_2 \in W_2 \wedge \cdots \wedge a_m \in W_m,$$

embodies the relationship between a subgroup and its compatible subgroup(s). The rule consists of attributes $a_1, \ldots, a_m$, where $1 \leq m \leq |A|$, and respective values v_i and sets of values W_i. The conjunction of clauses before the arrow is the *condition* and what comes after is the *conclusion*. Specifically, the former defines a *sender subgroup* and the latter defines its (multiple) compatible subgroups. When the profile of an active user matches the condition (values $v_1, \ldots, v_m$ match with profile), then the conclusion of the rule can be used to choose candidates that are members of compatible subgroups.

Extending a rule r with attribute $a \in A$, value v, and value set W results in a new rule r' that has clause $a = v$ conjuncted to the condition and $a \in W$

conjuncted to the conclusion. Attribute a must not have been previously included in r. The new rule r' is more *specific* than previous rule r.

Function $n(r)$ returns the number of initiatives sent by the subgroup of the condition of rule r (sender subgroup) to the (compatible) subgroups of the conclusion. Likewise $n^+(r)$ returns the number of reciprocated initiatives — those that were positively replied to. The *success rate* of a rule is then defined as $\mathit{success}(r) = n^+(r)/n(r)$.

The multiple compatible subgroups recommendation process involves two stages. The first stage involves finding compatible subgroup rules for each user x, and the second stage involves evaluating these rules to generate and rank candidates for the final recommendation lists R_x.

Algorithm 5 delineates the steps of rule set construction. It takes as input the active user x and base success rate s, which is the proportion of initiatives sent by all users of same gender as x that were reciprocated (positively replied to).

The algorithm starts off by initializing a cosmetic rule for gender to reflect the fact that the user set is already considered bipartite (male-female). List *rules* is initialized as a singleton consisting solely of this rule.

Algorithm 5 Multiple compatible subgroups: rule set construction

```
function ConstructRules(user $x$, success rate $s \in [0,1]$)
    $A' \leftarrow A \setminus \{a_{gender}\}$
    if $x$ is male then
        $r \leftarrow (a_{gender} = \text{"male"} \Rightarrow a_{gender} \in \{\text{"female"}\})$
    else
        $r \leftarrow (a_{gender} = \text{"female"} \Rightarrow a_{gender} \in \{\text{"male"}\})$
    $rules \leftarrow$ list $\langle r \rangle$
    loop
        $r_a, s_a \leftarrow$ ExtractRule($r, a, U_{x,a}$) for all $a \in A'$
        $\alpha \leftarrow \arg\max_{a \in A'} s_a$
        if success rate $s_\alpha$ improvement over $s$ insignificant then
            break loop
        $A' \leftarrow A' \setminus \{\alpha\}$
        $r, s \leftarrow r_\alpha, s_\alpha$
        append $r$ to $rules$
    return $rules$
```

On each iteration, a rule r_a and its success rate s_a is extracted for each attribute a not yet included in current rule r. The extracted rules r_a are extensions of the rule r. The rule r_α with largest success rate is chosen as new base rule r, but only if it increases the success rate and the increase is statistically significant. The final lines of the loop body involve straightforward bookkeeping to prepare for the next iteration.

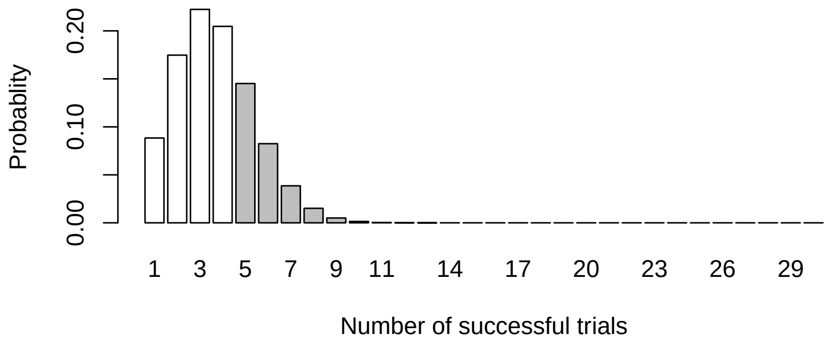


Figure 2: Given a previous success rate of $s = 0.12$, and with $n = 30$ initiatives of which $n^+ = 5$ have received positive reply, a p-value of around 0.2882 is obtained by summing the probability mass of (gray) bars n^+ to n, which is equivalent to Equation 1. Hence H_0 is not rejected.

To asses whether the increase in success rate has statistical significance, a one-way binomial test is employed on previous round's success rate s, the number of initiatives $n(r_\alpha)$ sent by the sender subgroup of the new rule r_α, and the number of positive replies $n^+(r_\alpha)$ by the compatible subgroups of rule r_α.

The null hypothesis H_0 is that the success rate of new rule r_α is no better than the success rate s of current rule [65]. Under the null hypothesis, given evidence $n = n(r_\alpha)$ and $n^+ = n^+(r_\alpha)$, we obtain a p-value of

$$p_{r_\alpha} = \sum_{k=n^+}^{n} Bin(k; s, n) = \sum_{k=n^+}^{n} \binom{n}{k} s^k (1-s)^{n-k}. \tag{1}$$

Operating on a significance level of 5%, we reject the null hypothesis when $p_{r_\alpha} < 0.05$ [65], and proceed to add rule r_α to collection *rules*. Otherwise, we terminate iteration because success rate increase was deemed insignificant. An example of not rejecting H_0 is given in Figure 2.

The precise statistical test used in the original study was not mentioned. In previous work by Kim et al. [28], in which they consider rules where only a single compatible subgroup is allowed, binomial tests were used. These tests make sense in our context as well because we can think of an initiative as a Bernoulli trial with two possible outcomes: positive reply (successful), or negative or no reply (unsuccessful). Given a series of independent and identically distributed Bernoulli trials, the number of successful outcomes is modeled by the binomial distribution [65].

Going back to Algorithm 5, sub-procedure ExtractRule is invoked to find a rule and its success rate were attribute a used to expand the current rule r. Let us go through this sub-procedure as presented in Algorithm 6.

First off, sets X and Y are initialized so that both conform to the previous

Algorithm 6 Multiple compatible subgroups: rule extraction

Require: v is value of attribute a for active user

```
 1: function ExtractRule(previous rule $r$, attribute $a$, value $v$)
 2:     $X \leftarrow$ all users defined by condition of $r$ and for whom $a = v$
 3:     $Y \leftarrow$ all users defined by conclusion of $r$
 4:     for each value $w$ of attribute $a$ do
 5:         $Y_w \leftarrow$ users of $Y$ for whom $a = w$
 6:         $s_x \leftarrow$ positive reply rate of initiatives sent from $X$ to $Y_w$
 7:         $s_y \leftarrow$ positive reply rate of initiatives sent from $Y_w$ to $X$
 8:         $\beta_w \leftarrow \frac{2}{s_x^{-1}+s_y^{-1}}$                ▷ harmonic mean
 9:     $\mu, \sigma \leftarrow$ mean and standard deviation of values $\beta_w$
10:     $W \leftarrow \{w : \beta_w \geq \mu + \sigma/2\}$
11:     $r_a \leftarrow$ expand $r$ with attribute $a$, value $v$ and value set $W$
12:     $s_a \leftarrow SG(r_a)$
13:     return rule $r_a$ and success rate $s_a$
```

rule r. The former contains all users that adhere to the condition and the latter those that adhere to the conclusion. Additionally, users of X are filtered by condition $a = v$ according to would-be rule r_a.

To determine which compatible subgroups to include in the conclusion of would-be rule r_a, attribute values w corresponding to compatible subgroups are iterated through. On each iteration, the harmonic mean (denoted β_w) of positive reply rates between users of X and users of Y_w (filtered from Y by $a = w$) is calculated. The *positive reply rate of initiatives sent* from users of set A to users B is calculated as

$$\frac{\sum_{a \in A} |M^+_{a,*} \cap B|}{\sum_{a \in A} |M_{a,*} \cap B|} \in [0, 1].$$

Set W, representing the selected compatible subgroups, is then composed of attribute values w whose score β_w is at least 0.5 standard deviations above the mean. Lastly, candidate rule r_a is constructed from a, v, and W, and it alongside its success rate are returned. An example of this algorithm in action is illustrated in Figure 3.

The motivation for using harmonic mean was discussed in Section 3.3. The use of threshold $\mu + \sigma/2$ is a heuristic. This heuristic is based on (unshown) preliminary analysis done by Kim et al. [30] and in this thesis no reason was found to change it, as it seemed to work well enough.

That completes our exposition to the rule set construction step of the recommender. After assigning rules to users, candidates must be generated and ranked for recommendations.

Given user x with a sequence of rules $r_1, \ldots, r_n$ ordered by increasing

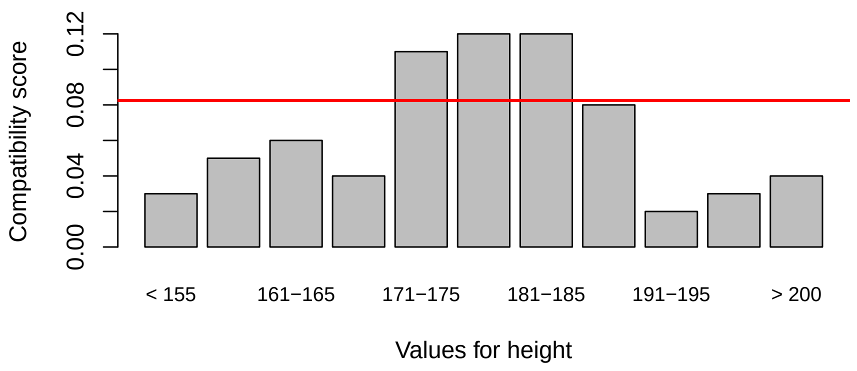


Figure 3: An illustration of rule extraction given attribute "height". Threshold $\mu + \sigma/2$ is drawn as a horizontal red line. Subgroups whose bar rises above threshold are chosen for value set W. Adapted from Kim et al. [30].

specificity (i.e., r_{i+1} is more specific than r_i, as is output by Algorithm 5), candidates are chosen by collecting all unmet users belonging to the compatible subgroups of rules $r_1, \ldots, r_n$. Candidates are then ranked using the following criteria, with ties broken at each level by using subsequent criteria:

1. In descending order by the most specific rule that applies. For example, if r_k is the most specific rule that applies to user `Jim` and r_l, where $k < l$, is the most specific rule that applies to `Bob`, then Bob is ranked above Jim.

2. In descending order of individual positive reply rate rounded to two digits. If rule r_k is the most specific that applies to `Bob` and `Jim` then the candidate with higher positive reply rate is ranked above the other. If candidate is a cold-start user, the base success rate of all users is used in comparison.

3. In descending order of number of positive replies.

4. In descending order of number of received initiatives.

5. Remaining ties are randomly broken.

The last step then is to cut the recommendation lists so that N or less candidates are left.

To the author of this thesis, implementing multiple compatible subgroups required the most interpretation of all methods presented thus far. Details on how exactly statistical significance was calculated and whether the comparison was made to the success rate of the previous round or always to the base rate was left unclear in the method description. Furthermore, regarding ranking, there was no mention on how to handle cold-start users for whom we have no individual positive reply rates. These missing details were addressed as given above.

An advantage of the multiple compatible subgroups method is that it is able to recommend cold-start users to old users, vice versa, as well as recommend cold-start users to other cold-start users. As candidates are picked from compatible subgroups based on profile attributes, the actual messaging behavior of a candidate is downplayed (although it still impacts secondary, tertiary and quaternary ranking).

This *global* property of the method also has its downside. Recommendations are not personalized for the individual user but are rather based on the sender subgroup that the current user happens to belong to by virtue of their profile. In particular, if the user has preferences that are unlike those of his or her peers, the user will be presented with recommendations that may not be to his or her liking. Any messaging behavior gathered later on will not impact the recommendations received (except for removing met users from recommendations), but only the position of the user in others' recommendations will be impacted.

The method is heavily based on male-female preferences and would require some adaptation and modification for non-heterosexual dating domains. Notably, the notion of a bipartite graph should be extended to allow for male-male, female-female, and bisexual preferences.

Further work on the method could involve better secondary ranking of cold-start users, who are currently assigned the base success rate effectively pushing them down in recommendations. Kim et al. [28] combined subgroup rules with collaborative filtering but did so only for compatible subgroup rules with singe value conclusions [29]. As a further line of research, they suggested hybridizing multiple compatible subgroup rules as well.

Lastly, the generated rules might have utility in and by themselves. In an unsupervised setting, for example, one could look for co-occurring attribute values in conclusions, and use the discovered patterns to gain knowledge about the data set.

4 Thompson sampling approach

Each of the methods presented in detail in Section 3 suffer from drawbacks that may render their recommendation performance subpar. RECON and CCR have problems dealing with cold-start users, while multiple compatible subgroups method cannot offer truly personalized recommendations. We decided to develop a novel method that incorporates the strengths of each of the previous methods, while attempting to avoid their downsides.

Our method casts the problem of recommendation as a multi-armed bandit problem[3] and is inspired by a Bayesian approach to multi-armed bandits

[3]In the K-armed bandit problem [8], a player is faced by K slot machines, each having

called Thompson sampling [12]—hence the name of our approach. Our method combines some of the best elements of earlier algorithms: RECON's preference model but in a probabilistic formulation, CCR's profile distance function for similarity, and ranking criteria as well as success rates between groups of users from the multiple compatible subgroups method.

Using these components, users are first presented recommendations according to the demography they happen to belong to. As messaging behavior data is amassed on an individual, their preference model can be steered towards their demonstrated preferences. Hence cold-start users are handled appropriately, whilst still being able to provide personalized recommendations when data become available.

This procedure could be implemented without bandits. Bandits introduce a degree of randomness to the recommendation process that allows for occasional surprises and perhaps even *serendipitous* recommendations. These are recommendations that are both surprising and useful to (or liked by) the user [22, 42, 57]. It has been argued that focusing solely on the accuracy of recommenders is harmful and that alternative metrics such as serendipity should be considered in conjunction when optimizing for user experience [42]. Another motivation for adding randomization components is to increase the (catalog) coverage, i.e., the number of distinct candidates across all recommendations [22]. These metrics are discussed further in Section 6.1.

Our method consists of two major steps—two phases during which each a pass is made through the entire recommendation list of an active user. On the first pass, slots of the recommendation list are allocated for specific clusters of users. Not until the second pass is the list substantiated with actual candidates to stand in place of the allocated clusters.

4.1 Statistical models

Before describing the two phases of the recommendation process, we briefly describe the Bayesian statistics we will be utilizing along with relevant statistical models.

At the heart of Bayesian statistics is *Bayes' theorem.* Given data D and a hypothesis space $\mathcal{H}$, the probability of hypothesis $H \in \mathcal{H}$ when having observed data D is given by

$$p(H \mid D) = \frac{p(D \mid H)\, p(H)}{p(D)} = \frac{p(D \mid H)\, p(H)}{\int_{\mathcal{H}} p(D \mid H')p(H')dH'}.$$

Above, we treat the hypothesis H as a random variable. Term $p(H)$ implies a prior distribution over hypothesis space $\mathcal{H}$ which models our subjective belief in the different alternatives before observing any data.

a reward distribution unknown to the player. Given a limited number of pulls, the task of the player is to come up with an optimal strategy of pulling the K different levers so as to maxmize the sum of received rewards.

This is set by us before observing any data. Likelihood $p(D \mid H)$ is the probability of observing data D given hypothesis H were true. It is often a straightforward calculation based on the probability mass/density function of a statistical model(s). Finally, $P(D)$ is the probability of observing data D.

Concretely speaking, the hypothesis space $\mathcal{H}$ is often represented as a parameter or vector of parameters (as will become apparent in the next paragraphs). Secondly, when simply comparing hypotheses, we can drop the computationally challenging denominator $P(D)$ from the equation, as it only serves to normalize the result and is constant with respect to hypotheses $H \in \mathcal{H}$.

Given a Bernoulli trial, or an event with two possible outcomes—success (1) and failure (0)—in which the probability of success stays constant over subsequent trials, the probability of success is modeled by the simple Bernoulli distribution: $p(X = 1) = p$ and $p(X = 0) = 1 - p$, where $p \in \mathbb{R}$, $0 < p < 1$ is the success probability parameter.

Given some historical trial data D, we may estimate p to be the ratio of successes occurring in D. This is called maximum likelihood estimation. However, because we are working with Bayesian statistics, we treat p as a random variable, and instead update the distribution of p to reflect our newly gained knowledge.

Before any data however, the hypothesis space $]0, 1[$ for p is modeled using the Beta distribution: $p \sim Beta(\alpha, \beta)$. Hyperparameters $\alpha > 0$ and $\beta > 0$ control the shape of the distribution. Increasing α shifts the density peak closer to 1 and β shifts it closer to 0. Their sum controls the steepness of the peak. This reflects the prior distribution $p(H)$, or, our subjective belief on how p is distributed.

After observing data D, we can calculate an "updated" Beta distribution $p \mid D \sim Beta(\alpha + a, \beta + b)$ where a is the number of successes and b the number of failures observed in data D. This is the *posterior distribution* $p(D \mid H)$ and accounts for the data witnessed. The statistics work out nicely because the Beta distribution is a *conjugate prior* of the Bernoulli distribution. Otherwise, $p(D \mid H)$ may not have a closed-form solution.

The above scenario can be repeated multiple times, i.e., the posterior distribution updated with new data to gain a new posterior distribution. After each update, the density peak for p slowly converges towards the true success rate probability because more data are utilized.

If instead of just success and failure, there are several outcomes or *categories*, e.g., red, black, blue, or green, the Categorical distribution can be used to calculate the probability of a certain category occurring. Given K categories, each having an event probability p_i of occurring, where $i \in 1, \ldots, K$ and $\sum p_i = 1$, the probability mass function is quite simply given by $f(i) = p(x = i) = p_i$.

The multivariate generalization of the Beta distribution is called the

Dirichlet distribution, and it serves as a conjugate prior to the Categorical distribution. Its shape is controlled by positive hyperparameters $\alpha_1, \dots, \alpha_K$. It models our belief in different event probability vectors $(p_1, \dots, p_K) \sim \mathit{Dirichlet}(K, \alpha_1, \dots \alpha_K)$ that serve as parameters to the Categorical distribution. Given data D of occurrences of categories, the calculation of the posterior distribution is done in similar fashion to the Bernoulli-Beta model, i.e., $(p_1, \dots, p_K) \mid D \sim \mathit{Dirichlet}(K, \alpha_1 + a_1, \dots, \alpha_K + a_K)$ where a_i the number of times the i:th category was observed in data D.

4.2 First pass: cluster assignment

The first phase begins with the use of the distance function D of CCR (presented in Section 3.5) to calculate the distance between users of same gender. Both male and female users are then hierarchically clustered based on the distances using an agglomerative (bottom-up) approach based on Ward's minimum variance criterion [64]. (The exact variant of the method is `Ward1` without squaring the input distance matrix [45].) The resulting hierarchies are then cut (or merged) so that k clusters remain per gender.

Given clusters $I_1, \dots, I_k$ of male and $J_1, \dots, J_k$ of female users, the messaging relationships between them are modeled using multi-armed bandits represented as Beta distributions.

Take the example of user `Mike` in cluster I_i sending a message to `Ruby` in J_j. This process can be modeled as a Bernoulli trial in which the reply of `Ruby` determines whether the outcome is 1 (positive reply) or 0 (otherwise). Given all initiatives and replies between users of cluster I_i and J_j, and assuming naively that there is one true probability s_{ij} governing whether any user of I_i and of J_j would have a reciprocal interaction, using maximum likelihood estimation, the probability estimate becomes $s_{ij} = \mathit{\#positive\ replies\ /\#initiatives}$.

However, instead of a single point estimate, to be Bayesian, a distribution is put over s_{ij} to model the uncertainty of the estimate and to allow for randomness in the lever pulling phase of the bandit algorithm. The logical model for the distribution here is the Beta distribution $s_{ij} \sim \mathit{Beta}(\alpha, \beta)$, where $\alpha > 0$ is the number of positive replies (reciprocations) occurring between clusters I_i, and J_j, and $\beta > 0$ is the number of initiatives that were not replied to or negatively replied to.

Historical messaging data are used to calculate parameters α_{ij} and β_{ij} for Beta distributions $s_{ij} \sim \mathit{Beta}(\alpha_{ij} + \alpha_0, \beta_{ij} + \beta_0)$ for each pair of clusters $i, j \in [1, k]$. The exact calculations are delineated below.

$$\alpha_{ij} = \sum_{x \in I_i} |M^+_{x,*} \cap J_j| + \sum_{x \in J_j} |M^+_{x,*} \cap I_i|$$

$$\beta_{ij} = \sum_{x \in I_i} |M_{x,*} \cap J_j| + \sum_{x \in J_j} |M_{x,*} \cap I_i| - \alpha_{ij}$$

Pseudocounts α_0 and β_0 are used to set prior distributions over s_{ij}. Our strategy is to set them so as to have the mode of prior distributions be equal to the positive reply rate across all messages (disregarding clusters). Let $0 < c < 1$ be this reply rate (desired mode). Then from the equation of mode for the Beta distribution, the relationship between variables α_0 and β_0 is derived:

$$c = \frac{\alpha_0 - 1}{\alpha_0 + \beta_0 - 2} \implies \alpha_0 = \frac{c(\beta_0 - 2) + 1}{1 - c} \quad (\alpha_0, \beta_0 > 1)$$

Now we are left with one parameter β_0 from which we can infer α_0. The exact value for β_0 depends on how certain we are of the prior, and it is left as a tunable parameter.

Given user $x \in I_i$ and candidate $c \in J_j$, to calculate the probability of a reciprocal interaction occurring between them, in a Bayesian context, the posterior predictive distribution can be used,

$$\begin{aligned} p(I_i \leftrightarrow J_j = 1 \mid M, \alpha_0, \beta_0) &= \int_0^1 p(I_i \leftrightarrow J_j = 1 \mid s_{ij}) p(s_{ij} \mid M, \alpha_0, \beta_0) ds_{i,j} \\ &= \int_0^1 s_{ij} p(s_{ij} \mid \alpha_{ij}, \beta_{ij}, \alpha_0, \beta_0) ds_{i,j} \\ &= \mathbb{E}[s_{ij} \mid \alpha_{ij}, \beta_{ij}, \alpha_0, \beta_0] \\ &= \frac{\alpha_{ij} + \alpha_0}{\alpha_{ij} + \alpha_0 + \beta_{ij} + \beta_0}, \end{aligned}$$

where M represents all available messaging data.

However, in order to induce randomness into the recommendation procedure, instead of using the predictive distribution, the posterior distribution $Beta(\alpha_{ij} + \alpha_0, \beta_{ij} + \beta_0)$ is *sampled* for a reply rate. In the sampling process, the number of historical initiatives observed beforehand accounts for the variability of the reply rates drawn. Specifically: more historical data leads to more certainty, ergo, less variability.

The first pass of the recommendation process is outlined in Algorithm 7. The first lines involve clustering of users and counting the α_{ij}, β_{ij} parameter pairs, as was described in detail earlier. After preliminary work is done, the recommendation list is pre-allocated with clusters of the opposite sex by pulling the levers of respective bandits. In other words, given cluster ω of the active user x, the bandit for each pair (I_ω, J_j) $j \in [1, k]$ is sampled and the resulting draws compared. Whichever j had the largest sample is given the slot of the current iteration n, assuming there are enough users left in the cluster to fill in the slot.

4.3 Second pass: candidate assignment

After the first pass, each element of the recommendation list is assigned to a cluster. But exactly which user should be chosen in each case, is still unre-

Algorithm 7 Thompson sampling approach: 1st pass

```
1:  function FirstPass(user $x$, recommendation list size $N$, $k \in \mathbb{N}$, $\beta_0 > 1$)
2:      Split users by gender into sets $I$ (same gender as $x$) and $J$
3:      Split users of each gender into $k$ clusters: $I_1, \dots, I_k$ and $J_1, \dots, J_k$
4:      $\alpha_{ij} \leftarrow$ Count positive replies between clusters $I_i$ and $J_j$
5:      $\beta_{ij} \leftarrow$ Count negative and no replies between clusters $I_i$ and $J_j$
6:      $\omega \leftarrow$ cluster of user $x$
7:      $slots \leftarrow \{\}$
8:      $m_j \leftarrow 0 \; \forall j$
9:      for each slot $n \in 1, \dots, N$ do                              ▷ assign slots
10:         for each cluster $j \in 1, \dots, k$ do                       ▷ pull levers
11:             if $m_j < |J_j|$ then
12:                 $s_{\omega j} \leftarrow$ draw sample from $Beta(\alpha_{\omega j} + \alpha_0, \beta_{\omega j} + \beta_0)$
13:         $\widehat{j} \leftarrow \arg\max_j s_{\omega j}$
14:         $slots[n] \leftarrow \widehat{j}$
15:         $m_{\widehat{j}} \leftarrow m_{\widehat{j}} + 1$
16:     return SecondPass($x$, $slots$)
```

solved. To make that final selection, the second pass of the recommendation process is invoked, in which the personal preferences of the active user is considered.

The preferences of a user are modeled using a series of independent Categorical-Dirichlet models, one per attribute. Given attribute $a \in A$ with K distinct values $v_1, \dots, v_K$, the probability of an active user x preferring a user with some value v is distributed according to the Categorical distribution $v \sim \mathit{Categorical}(K, \boldsymbol{p}_a)$, where $\boldsymbol{p}_a = (p_1, \dots, p_K)$ is a vector of proportions for each distinct attribute value and for which conditions $p_i \in [0, 1]$ and $\sum p_i = 1$ hold. For any value v_k, the probability is given by the probability mass function: $p(a = v_k) = f_a(k\,;\boldsymbol{p}_a) = p_k$.

Being Bayesian, parameter $\boldsymbol{p}_a$ of the Categorical distribution is interpreted as a random vector which is distributed according to its conjugate prior, the Dirichlet distribution: $\boldsymbol{p}_a \sim \mathit{Dirichlet}(K, \boldsymbol{\alpha}_a)$, where $\boldsymbol{\alpha}_a = (\alpha_{a,1}, \dots, \alpha_{a,K})$ is a vector of positive pseudocounts for each distinct value $v_1, \dots, v_K$.

Having historical messaging data of user x available, the posterior Dirichlet distribution for attribute a is described by:

$$\boldsymbol{c_a} = (c_{a,1}, \dots, c_{a,K})$$
$$\boldsymbol{p} \mid (M_{x,*} \cup M^{+}_{*,x}), \boldsymbol{\alpha}_a \sim \mathit{Dirichlet}(K, \boldsymbol{c}_a + \boldsymbol{\alpha}_a)$$

The set $M_{x,*} \cup M^{+}_{*,x}$ consists of profiles user x has sent initiatives to or has positively replied to. From these profiles a vector of counts $\boldsymbol{c}_a \in \mathbb{N}^K$ for values $v_1, \dots, v_K$ of attribute a are aggregated. The counts are used to update the Dirichlet distribution to reflect the supposed preferences of the active user x.

For cold-start users for whom no messaging data are available, the prior distribution can be used as such, with pseudocounts $\boldsymbol{\alpha}_a = (1, \ldots, 1)$, making the prior distribution effectively uniform. The resulting model, however, is not very useful because it cannot differentiate between and rank the users within the clusters—the very problem we are out to solve. For this reason, in addition to counts $\boldsymbol{c}_a$ and $\boldsymbol{\alpha}_a$, the positive interactions of l nearest neighbors w.r.t. distance D of user x are used to form a second count vector $\boldsymbol{d}_a \in \mathbb{N}^K$. As the magnitude of counts obtained in this manner tend to be considerably higher than the counts $\boldsymbol{c}_a$ (because we are counting the interactions of l neighbors), we divide the vector by the sum of its elements (number of interactions).

Given user x and user y of opposite gender, the probability of a match, or both users liking each other, given messaging data M, is modeled as:

$$p(x \leftrightarrow y \mid M) = p(x \to y, y \to x \mid M) \overset{\perp\!\!\!\perp}{=} p(x \to y \mid M)\, p(y \to x \mid M)$$

Preferences of both users are considered for reciprocity and it is assumed they are independent. A one-way probability is then calculated as:

$$\begin{aligned} p(x \to y \mid M) = p(y \mid \boldsymbol{c}^{(x)}, \boldsymbol{d}^{(x)}, \boldsymbol{\alpha}) &= p(A = U_y \mid \boldsymbol{c}^{(x)}, \boldsymbol{d}^{(x)}, \boldsymbol{\alpha}) \\ &\overset{\perp\!\!\!\perp}{=} \prod_{a \in A} p(a = U_{y,a} \mid \boldsymbol{c}_a^{(x)}, \boldsymbol{d}_a^{(x)}, \boldsymbol{\alpha}_a) \\ &= \prod_{a \in A} f_a(U_{y,a}; \boldsymbol{p}_a^{(x)}) \end{aligned}$$

Above U_y is the profile of user y, f_a is the probability mass function for attribute a, $\boldsymbol{c}^{(x)}$ and $\boldsymbol{d}^{(x)}$ refer to counts related to user x across all attributes $a \in A$, whilst $\boldsymbol{\alpha}$ are user-independent pseudocounts across all attributes. Vector $\boldsymbol{p}_a^{(x)}$ refers to an actual parameterization of a categorical distribution and is obtained as an affine transformation of $\boldsymbol{c}_a^{(x)}$, $\boldsymbol{d}_a^{(x)}$, and $\boldsymbol{\alpha}_a$ (Shown soon, in Line 11 of Algorithm 8).

The probability of user x preferring user y thus reduces to the probability of user x preferring the candidate's profile attributes U_y (compared to all other possible combinations of profile attributes). A naive assumption is made that attributes are independent of each other, which is not really true since, for example, hair color and age are most likely correlated. Nevertheless, the model can still prove useful even in the presence of such correlations since Naive Bayes classifiers, making similar independence assumptions, often work surprising well in the presence of feature correlation [58].

Algorithm 8 describes the second pass in whole. Lines 3 to 9 are the steps necessary for calculating the above counts for a single user. After they have been calculated, the next step is to arrange the candidates in each considered cluster, after which the final recommendation list assignments can be made.

Algorithm 8 Thompson sampling approach: 2nd pass

```
Require: Counts $c^{(y)}$ and $d^{(y)}$ are known in advance for each candidate $y$.

1: function SecondPass(user $x$, $slots$, $l \in \mathbb{N}$ , pseudocounts $\boldsymbol{\alpha}$)
2:     $R \leftarrow (M_{x,*} \cup M_{*,x})^{\complement}$                          ▷ unmet users
3:     $Z \leftarrow l$ nearest neighbors of $x$ (same gender)
4:     for attribute $a \in A$ do                                  ▷ Initialize counts
5:         for value $v$ of attribute $a$ do
6:             $c_{a,v} \leftarrow |\left\{y \in M_{x,*} \cup M^{+}_{*,x} : U_{y,a} = v\right\}|$
7:             $d_{a,v} \leftarrow 0$
8:             for neighbor $z \in Z$ do
9:                 $d_{a,v} \leftarrow d_{a,v} + |\left\{y \in M_{z,*} \cup M^{+}_{*,z} : U_{y,a} = v\right\}|$
10:    for attribute $a \in A$ do
11:        $\boldsymbol{p}_a^{(x)} \leftarrow$ draw sample from $Dirichlet(K, \boldsymbol{c}_a + \frac{\boldsymbol{d}_a}{\sum_v d_{a,v}} + \boldsymbol{\alpha}_a)$
12:    for each unique cluster index $j \in slots$ do           ▷ arrange candidates
13:        $n_j \leftarrow$ number of occurrences of $j$ in $slots$
14:        for each user $y \in J_j \cap R$ do
15:            $p_{yx} \leftarrow p(y \rightarrow x \mid M)$
16:            $p_{xy} \leftarrow \prod_a f_a(U_{y,a}; \boldsymbol{p}_a^{(x)})$
17:            $s_y \leftarrow p_{yx} p_{xy}$
18:            $r_y \leftarrow$ positive reply rate of $y$
19:        $C_j \leftarrow$ keep $n_j$ users from $J_j \cap R$ with largest $s_y$
20:        Rank users of $C_j$ by $s_y$ and $r_y$; re-arrange by mean rank
21:    $recs \leftarrow$ empty list                           ▷ build recommendation list
22:    for $i \in 1, \ldots, length(slots)$ do
23:        $recs[i] \leftarrow$ pop head of $C_{slots[i]}$
24:    return recommendation list $recs$
```

Unmet candidates are arranged within clusters by probability of two-way preference between the active user x and candidate y. However, instead of directly evaluating $p(x \leftrightarrow y \mid M)$, the preferences of user x, i.e., $\boldsymbol{p}_a^{(x)}\ \forall a \in A$, are sampled from the respective Dirichlet posterior distributions. This is done in furtherance of randomness. Variation of samples depends, again, on the amount of messaging data available on active user x. In contrast to sampling, for the candidates, the posterior predictive distributions are used as such without any sampling.

After the candidates of a cluster J_j have been ordered, the n_j largest are chosen, where n_j is the number of cluster slot assignments. Next, these candidates are assigned two distinct ordinal ranks: one by preference probability s_y and the other by positive reply rate r_y. If a candidate has received zero initiatives, the base positive reply rate of their gender is used instead. For each user, the ranks are summed and divided by two resulting in the mean rank. Users are then re-arranged based on this score with ties broken arbitrarily.

Assembling the recommendation list is the final, trivial step. Iterating through the cluster slot assignments, candidates are picked from within the clusters in order of mean rank.

4.4 Considerations

Our method consists of four separate parameters. Number of clusters $k \in \mathbb{N}$, number of pseudo negative-or-no replies $\beta_0 > 1$, number of nearest neighbors to use $l \in \mathbb{N}$, and attribute pseudocounts $\boldsymbol{\alpha}$. Attribute pseudocounts were set to induce uniform priors ($\alpha_{a,v} = 1$ for all a, v). The effect of number of nearest neighbors l used seemed to have little effect on key measures (Section 6.1)—a value of 40 worked well. We set $\beta_0 = 30$; a larger value leads to a sharper distribution, and vice versa.

The most important attribute with regards to key measures seemed to be the number of clusters k. A systematic study of the parameter was performed with other parameters fixed. Testing number of clusters k at 5, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 150, 200, we noticed that increasing k improves coverage while reducing success rate and recall. A sweet spot seemed to be $k = 40$ at which improvement of coverage plateaued, but success rate and recall did not suffer too much. (See Section 6.1 for measures.)

Further development of our approach could include ideas like employing alternative clustering methods or distance functions, finding a way to utilize negative replies separately from no replies, using a probabilistically sound model for sorting and ranking candidates during the second pass, and instead of relying on the naive assumption that attributes are independent, model dependencies using tree-augmented Naive Bayes [21] or learn the conditional independencies via Bayesian networks [50].

One idea that we were unable to get working well is utilizing individual messaging behavior during FirstPass by assigning a bandit to each user that is initialized according to the cluster he or she belongs to. This bandit could then be refined according to the messaging behavior we have amassed on said user. The problem here is that the number of initiatives sent between clusters are magnitudes larger than between individual users. Some sort of multiplier should be used, but how exactly should one come up with one is not obvious. We tried different multipliers, but were unable to see any major improvements in key measures.

Another adjustment that could be made is normalizing $\boldsymbol{d}_a$ less aggressively. In the above presented method, the impact of $\boldsymbol{d}_a$ is small and mostly affects

Profiles	
Total number	26538
Male	15117 ($\approx$ 57%)
Female	11421 ($\approx$ 43%)

Table 2: Summary of profiles data.

Initiatives		
Total number	294740	
Training set	174636	($\approx$ 59%)
Test set	120104	($\approx$ 41%)
Positively replied to (reciprocated)	25831	($\approx$ 8.8%)
Negatively replied to (rejected)	189101	($\approx$ 64%)

Table 3: Summary of initiatives data.

users with little to none messaging behavior data available.

5 Data set

In this section, the historical real-world data set used for evaluating and comparing the algorithms of Section 3 and Section 4 is presented. In next Section 6, the evaluation set-up alongside its results are presented.

To start off this section, a brief summary of the considered data set is given, followed up by a walk-through of how it was obtained by means of data wrangling and transformation and how it was divided into training and test sets. In the subsection that follows, profile attributes as well as selecting and engineering them are described, and in the last subsection, a series of exploratory analyses are presented that characterize the data.

A snapshot of user profiles and messaging history data was sampled from an online dating website operating in Finnish markets. The data set consisted of messages exchanged and profiles that had logged in during the 4-month period from October 2014 to January 2015. The final data set, after filtering and transformation, is summarized in Table 2 and Table 3.

Only users with undeleted profiles that had not been flagged as scammers, who had not blocked communications, and who had finished creating their profiles, were included in the data. Messages between two users were included only if both participants satisfied the above requirements. As a final step, users who were not involved in at least one message, as either sender or receiver, were left out. (Table 2 and Table 3 describe the state of affairs after all these steps.)

We decided to focus on self-reported heterosexual users because this sexual orientation was overwhelmingly represented in our data set and it eased the workload of implementing the recommendation algorithms. The applicability of each method to non-traditional gender preferences is discussed in Section 3. In particular, RECON (Section 3.3) provides support for non-traditional gender preferences out-of-the-box.

Regarding historical messaging data, the metadata of message chains between users were analyzed to determine the initiator and the number of messages sent in each direction. Whether any of the initiator's messages were read and if any were deleted, was also recorded.

In notation presented in Section 3.2, it is defined that an initiative may receive a positive (reciprocated), negative (rejected), or no reply at all. The presented recommender algorithms take advantage of these labels to differentiate between successful and unsuccessful interactions. However, in the context of the online dating site that we studied, replies were not accompanied by explicit positive or negative labels.

To deduce such labels, two basic approaches came to mind: 1) use number of messages exchanged and their direction to determine whether the initiative was reciprocated, or "positively replied to" so to speak; 2) use a natural language processing technique called sentiment analysis [48] on reply message bodies to determine the mood of the text. The former was settled on because of its simplicity. Finnish is notoriously hard to analyze because of its numerous grammatical cases, complex inflections and liberal use of compound words [33]. Moreover, we were unable to find a Finnish sentiment analyzer with a permissive license and even if we had, a new question of whether or not the produced results would have been precise enough may have arisen.

An initiative begins with the first message in a chain. An initiative is reciprocated (or positively replied to, or successful) when the chain consists of at least three messages sent from sender to receiver, and three messages sent from receiver to sender.

Take the hypothetical discussion below:

A Hello. You seem like an interesting person. Would you like to have a chat?

B Hi there. I actually just started seeing someone. I wish you the best of luck in future endeavors!

A Oh, I see. Well, thank you for replying.

Clearly just responding to a message is not strong enough evidence that two users have initiated a meaningful contact. Even a couple back and forths may still be considered plain good manners. In a video presentation[4] by Co-Founder Christian Rudder of dating site OkCupid, it was mentioned in passing that their analysts have defined two messages in both directions to

[4]`https://www.youtube.com/watch?v=YX1gTVa1N78&t=16m21s`

indicate a match. To err on the side of caution, three messages was used in this study. Admittedly, the choice of value is rather ad hoc and arbitrary.

An initiative is defined as rejected (or negatively replied to) when the number of messages sent in each direction is 2 or less *and* the recipient has read or deleted some of the messages. The thinking here is that sometimes initiatives are sent to users who no longer use the service. We do not wish to label such initiatives as rejected so we require that at least one of the received messages be acted upon. On the other hand, if a recipient has read or deleted a message but has not replied or has replied with one or two messages (to politely decline?), it provides evidence that the interest was not mutual. Finally, whether or not the messages were deleted, only a couple back and forths imply that the discussion did not sustain for long enough for a "real connection" to occur between users.

Notice that we equate rejecting with negatively replying. The latter implies sending a message, which the recipient does not have to do under our definition. Reading or deleting a message from the initiator is enough. In this thesis, negative replies are primarily used to model the negative preferences of a recipient so whether an actual reply occurred or not is irrelevant. Finally, it should be kept in mind that in addition to positive and negative replies, there are initiatives that receive no reply. In Section 3.2, these were labeled as *null*.

A caveat concerning these two definitions is that non-predefined messaging costs money. In some cases, the recipient may not be willing to pay for membership to respond. However in such a situation, the receiver is unable to even read the message so false positives are avoided. Furthermore, users of the service may delete old messages from their inbox to keep things in check. In the later case, we still consider initiatives to be reciprocated if the number of messages in both directions is 3 or more.

With the aforementioned interpretations of reciprocation and rejection, it could be said that our task has changed from that of matchmaking to predicting messaging partners. This is true, but we are still operating within the realms of a dating site where people get into discussions to, presumably, find a partner or friend. Therefore, if we predict two users to have lengthy discussion potential, we are also likely finding good partner matches. Moreover, from a business viewpoint, it is desirable to help users find messaging partners because that will increase their overall engagement with the dating service.

As our goal is to be able to evaluate and compare the recommender algorithms on this data, some of the data need be withheld from the algorithms in the learning phase. The initiatives were split into training and test sets. All initiatives initiated between October 1st and December 1st became the training data that was available for the recommender algorithms to learn from. Initiatives from December 14th to January 19th became the test set against which each recommender would be evaluated and compared. The

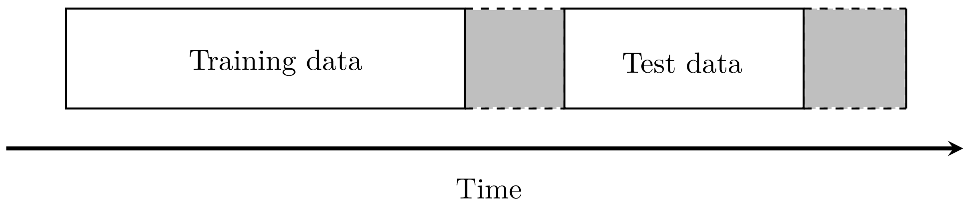


Figure 4: An illustration on how initiatives and their replies were collected and divided into training and test sets. Gray blocks indicate pause periods during which only replies were collected.

splitting was chosen so that approximately 60% of the initiatives were in the training and the remaining 40% in the test set.

In addition to the training and test periods, two week pauses were held after each period to collect any remaining replies (positive and negative) to initiatives of the previous period and to ensure that the training and test periods were properly insulated from each other so as to minimize temporal coupling. Any replies that occurred after pauses (to initiatives occurring before the respective pause) were simply ignored, as otherwise the recommenders might have had access to information regarding who is active during the test period. Initiatives sent during pause periods were ignored. A broad view of the periods and pauses is given in Figure 4.

Profile data were not separated into training and test periods because they would be readily available in a real world situation (and the task is to predict reciprocated initiatives not profile features). It should be noted that we did not have access to changes made to profiles during the 4-month period. The data collected are the state of the profiles on February 2$^{\text{nd}}$. This means that if a user, for example, moved to another region during the train-test period, we only had a record of their latest location and assume they lived there during the entire period. Ideally we would have access to the profile attributes in their exact state, whenever a user participated in messages.

5.1 Preprocessing profile data

For user profiles to be useful, an analysis of the raw attributes must be performed and then acted upon. Some attributes we may do without, for they may serve only to worsen the results of the recommenders. RECON (Section 3.3) in particular is sensitive to the distributions and correlations of attributes.

On the contrary to disclusion, new attributes may be created to capture hidden features or to fix the granularity of the data by zooming in our out to an appropriate level of detail. Combining (or collapsing) values of an attribute may serve the aforestated goals as well. It is also imperative to

registration date	last login date	has image
birthday	age preference	gender preference
relationship preferences	zodiac sign	self-introduction
partner description	gender	civil status
have children	want children	smoking
drinking	eye color	hair color
height	body	ethnicity
religion	industry	education
1st language	2nd language	country
region	locality	general interests
music genres	sports	#albums
#images	user id	profile image

Table 4: Raw profile features extracted from original database before performing preprocessing and feature engineering.

have the data in a format that the algorithms can handle. For example, most methods presented in Section 3 work on categorical data but not on ordinal or interval type data.

The process described here is often referred to as *feature engineering* [15], or more generally, *data preprocessing* [62]. The two goals are 1) to ensure that the data are compatible with the algorithms; 2) to squeeze extra performance from the data. The process often involves hours of manual labor in order to get the data into a format that is performant [15].

The raw attributes extracted from our profiles database are listed in Table 4. The attributes that were used without any modification are: has (profile) image, gender, have children, want children, smoking, drinking, hair color, height, body, and region. Each of these are rather self-explanatory and seemingly important in online dating.

Some attributes were used under the hoods but were not that interesting from the algorithms' point of view. User ids, gender preferences, registration dates, and last login dates are used to match profiles to initiatives and in selection of users for training and test periods. Having said that, registration dates could be used to identify new users and last login dates actively participating users, both which may be interesting qualities to people looking for dates. Gender preferences are of course crucial to users, however, in this study we focused on users with heterosexual preferences.

Many attributes were discarded because they did not have any apparent use or predictive capability, their distributions were too skewed, or analyzing them would have required significant attention and time.

Zodiac signs were left out because there is no reason to believe that they would have much significance. Ethnicity was left out because it was very skewed (91% European). Also 1st language and country were very skewed

(93% Finnish, 98% Finland) and thus dropped. We did not see an immediate need to include 2nd language although it could be used to detect whether a person is a native speaker. Age (range) preference was not used (because of lack of time), but it could provide good prior knowledge for cold-start users of the age of people they are searching for.

Number of albums, #albums, was also discarded, as a more accurate representation of how many images a user has is given by attribute #images, and #albums is just a technicality of how the user has divided those images.

The problem with locality and industry was that there were too many values for them to be useful and hence they were not included. There were over 300 localities some of which only had reportedly one person living in them. It was thus felt that coarser attribute region would work better. Although for some larger localities, it could prove useful to distinguish them from the region they reside in, e.g., capital Helsinki from surrounding region Uusimaa. However, this consideration was not implemented into our preprocessing.

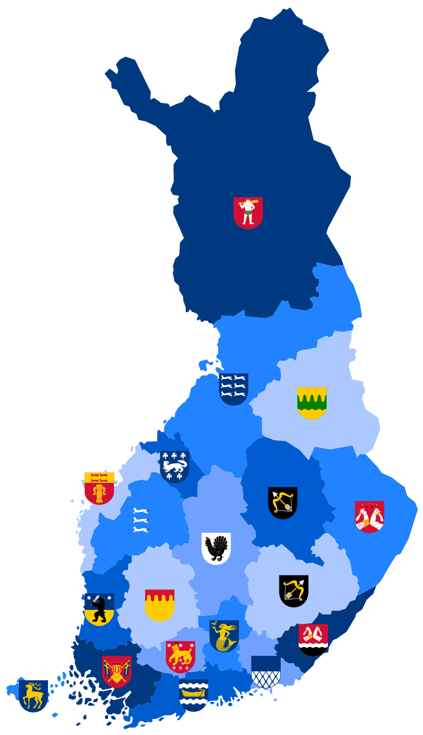
Industry (of employment) had over 26 options with a miscellaneous category having a majority of 18% of users. It was not immediately apparent how the industries should be grouped together for the attribute to be more useful.

Self-description and (ideal) partner description are free-text attributes that were not used because they would have required some heavy feature engineering to be useful. Studies have shown that free-text attributes are the second most important factor affecting the perceived attractiveness of a profile in online dating [17]. Finnish is a challenging language for text mining [33]. That being said, recently emerged free-to-use tools can lemmatize Finnish words into their base form enabling the use of bag of words and tf-idf models [39]. Given more time, this would definitely be one aspect we would have looked further into.

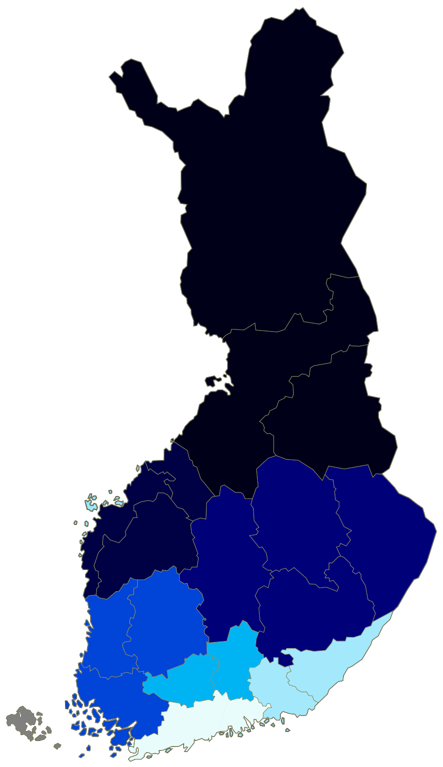
The most important predictor of profile attractiveness, profile images, were not tackled either [17]. This is a hard problem most likely warranting an entire body of research of its own, not only regarding how to get semantic information out of an image, but also on how to utilize this information in online dating. The author of this thesis is not aware of such work in the context of online dating matchmaking. Eigenfaces, as studied by Sirovich et al. [61], could be used to model positive and negative "average" faces as was done in a recent project [5]. Deep learning, a topic very much in vogue, is currently being harnessed for the task of generating sentences from images [27]. Perhaps future research on applying this work in online dating could yield interesting results.

Several attributes were either transformed or their values combined to create new categories. Birthdays were transformed into ages, and the numeric

[5]http://crockpotveggies.com/2015/02/09/automating-tinder-with-eigenfaces.html



(a) Regional map of Finland.[6]



(b) Attribute "location" created by merging adjacent regions (color-coded).[7]

Figure 5: Geographical attributes

ages were turned into age groups by intervals of five. Religions with few users were clumped into a miscellaneous group. Education backgrounds were divided into 'low', 'mid', and 'high' depending on whether they have attended upper-secondary and/or college-level studies. Eye colors of different shades but same base color were combined as well. For example, "dark brown" was clumped together with "brown". Number of images, #images, was transformed into a discrete attribute "total images" of three intervals.

Civil status values were very age skewed with older users preferring labels "divorced" or "widowed" and younger users preferring "single" to refer to the same inherent idea. Also reserved older people were usually married while younger were in a relationship. Values 'widowed', 'divorced', and 'single' were combined, 'married' and 'in relationship' were combined. Value 'living separately' was left as is. Arguably some semantic distinctions are lost, but on the other hand, we alleviate the strong correlations with age while still keeping the most important distinction intact.

[6]Derivative work based on work by user SokoWiki licensed under CC BY-SA 3.0.

[7]Derivative work based on work by Fallschirmjäger and licensed under public domain.

Some attributes were tailored for specific recommenders. The distance function of CCR (Section 3.5) requires calculating the physical distance between two users. Adjacent regions were collapsed to form a new coarser attribute "location". The final result along with original regions is shown in Figure 5. The combining of regions was based on hierarchical clustering of messaging data between users of different regions, historical provinces of Finland from 1997, and making sure that each location had an adequately sized representation in the user pool.

For RECON, continuous attributes were extracted from original variables 'general interests', 'music genres', and 'sports'. For each of these attributes, a user may select several values (or none). For example, "basketball", "hockey", and "gym" could be the choices for sports; "rock" and "pop" for music genres; and "fishing", "politics", and "traveling" as general interests for a single user. As there were several options (more than 20 for general interests and sports, and over 10 for music), some collapsing of the choices was deemed necessary.

For each of the three original variables, an incidence matrix was formed in which users represent rows and columns represent the different choices. The values in the matrices are binary: 1 when user has chosen the value and 0 otherwise. Correlations between choices were then calculated and these were turned into dissimilarities that were used to hierarchically cluster the choices for each of the three original variables.

The formed clusters are tabulated in Table 6. The names for these clusters were decided based on an intuitive grasp of what sort of original values were included each. For example, cluster masculine of general interests contained choices like "automobile", "motorbiking", "hunting", "spectator sports", and so forth. A user is assigned a value between 0 and 1 for each cluster based on the proportion of their choices that fall under each. Thus these values can be interpreted as the degree of interest a user has towards a category of choices. An additional requirement is that for a group, say "general interests", the values for clusters of a single user sum up to be equal to one.

In addition to the above attributes, attributes "does sports" and "likes music" were engineered by counting the number of choices a user has made on the original multivalued attributes. These counts were discretized so as to be available for all methods.

The final list of attributes alongside their values after data preprocessing are presented in Table 5 and Table 6. Many attributes have a "n/a" (not answered) value indicating that the value was not filled in by the user. In some cases, such as religion, this may signal that the user does not feel the attribute is relevant to them. In our methods, we mostly consider "n/a" as a value of its own without any special meaning albeit there may be merit in treating them specially, for example, by ignoring them in some cases.

As a final note, an observation about the data set in question. Finland is known to be ethnically a homogeneous society [2]. This was reflected in many of the attributes that we studied such as ethnicity, 1st language, and religion, each of which turned out to be too skewed to be useful. More indirectly though, this could been seen in attributes such as eye color and hair color that had certain values above the others. The challenge that might be introduced by such attributes is the ability to differentiate between candidates when making recommendations. Thankfully in this data set, there were other, less ethnicity-related attributes.

Discrete	
Attribute	Values
age group	18-22, 23-27, 28-32, . . . , 68-72, 73->
gender	male, female
has image	true, false
have children	yes, no
want children	yes, maybe, not relevant at the moment, no, n/a
civil status	single, living separately, reserved
smoking	yes, occasionally, no, n/a
drinking	yes, occasionally, no, n/a
eye color	blue, green, brown, gray, multicolored, n/a
hair color	brown, blond, dark, dyed, grayish, gray, bald, black, red, ash blond, white, n/a
height	$<$ 155, 156-160, 161-165, . . . , 196-200, $>$ 200, n/a (cm)
body type	slim, average, muscular, full-figured, stocky, athletic, n/a
religion	Lutheran, Orthodox, Catholic, Protestant, Pentecoastal, Muslim, agnostic, atheist, other, n/a
education level	low, mid, high, n/a
region	*19 regions of Finland (2015)*, other
location[8]	*See Figure 5.*
does sports	none, some, passionate
likes music	none, some, passionate
total images	0, 1-3, 4 or more

Table 5: Final discrete attributes after preprocessing and feature engineering. "n/a", not answered, indicates missing value.

[8]Used exclusively by CCR

Continous
General interests
arts
culture
entertainment
feminine
masculine
Music interests
africanamerican
classical
indie
modern
popular
rock
Sport interests
feminine
masculine
intense
relaxed

Table 6: Final continuous attributes after preprocessing and feature engineering. Each takes a value in the interval $[0, 1]$. For a single user, the sum of each group (general, music and, spot) must add up to 1. These attributes are used exclusively by RECON.

5.2 Exploratory analysis

To get a sense of the data set we are working with, in this section, some key aspects of the data are characterized by showing the results of exploratory analysis conducted on said data. Key characteristics are summarized as tables and figures.

The value of conducting this analysis is two-fold. First of all, we are able to empirically verify some of the claims made about online dating that were presented in Section 2.2.1. Secondly, we get a sense of the data we are working with, thereby enabling us to better develop and utilize methods that suite the data in question.

In Table 7, the number of unique users active during the training and the test period are tabulated. Also tabulated is the portion of users active during the training period that were no longer active during the test period, and the number of users active during the test period that were not yet active during the training period.

A few interesting insights are worth noting. Around 36% of users in the test set are cold-start users, which lends support to the claim that cold-start users constitute a large portion of the user base in online dating.

Users in training set	20296	Users in test set	17454
not in test set	9084 ($\approx$ 45%)	not in training set	6242 ($\approx$ 36%)

Table 7: Users involved in initiatives during training and test periods.

No messaging behavior data are available for cold-start users meaning that the recommender algorithms have little personal data to work with when providing such users with recommendations. This is especially problematic for RECON, and for CCR to some extent.

It should also be taken into account that 45% of users involved in the training set are no longer active during the test period. In Section 6, in which we introduce our set-up for the comparison of methods, it is noted that we can safely skip generating recommendations for these users, as they are not involved in interactions in the test set.

Taking a broader view of the two points presented above, it seems that user involvement with the dating site is sometimes short-lived. Thus the ability to provide cold-start users with quality recommendations is imperative. This is one of the motivations for developing our Thompson sampling approach (Section 4).

In Figure 6, users of the training period are distributed according to their participation activity in (training) initiatives as both senders and receivers.

To the very left we can see that almost 40% of users have not sent a single initiative. These users are either inactive or reactive. Reactive users tend to send no or hardly any initiatives and wait for others to contact them. To the very right we see that almost 20% of users are very active, i.e., sent 9 initiatives or more. The gray bars indicate popularity, with unpopular users to the left, and highly popular users to the right. Because of limited availability per user, it is crucial for the methods not only to recommend popular users, as they may be easily overburdened with initiatives and choose not to respond at all.

On the left side of Figure 7, the effect of gender on messaging proactiveness is shown. Given a user x, their *proactiveness* is defined as

$$\mathit{proactiveness}(x) = \frac{|M_{x,*}| - |M_{*,x}|}{|M_{x,*}| + |M_{*,x}|} \in [-1, 1].$$

A value of -1 indicates that a user only receives messages and 1 indicates that they only send messages. A value close to zero indicates that a user sends and receives initiatives equally.

From the left-hand side of Figure 7 we see that in our data set, female users tend to be reactive and male users active. Interestingly, there is a substantial portion of men that are highly reactive, which differs from the results of a similar study presented for an Australian online dating data set [52]. It is important to recommend reactive users alongside proactive

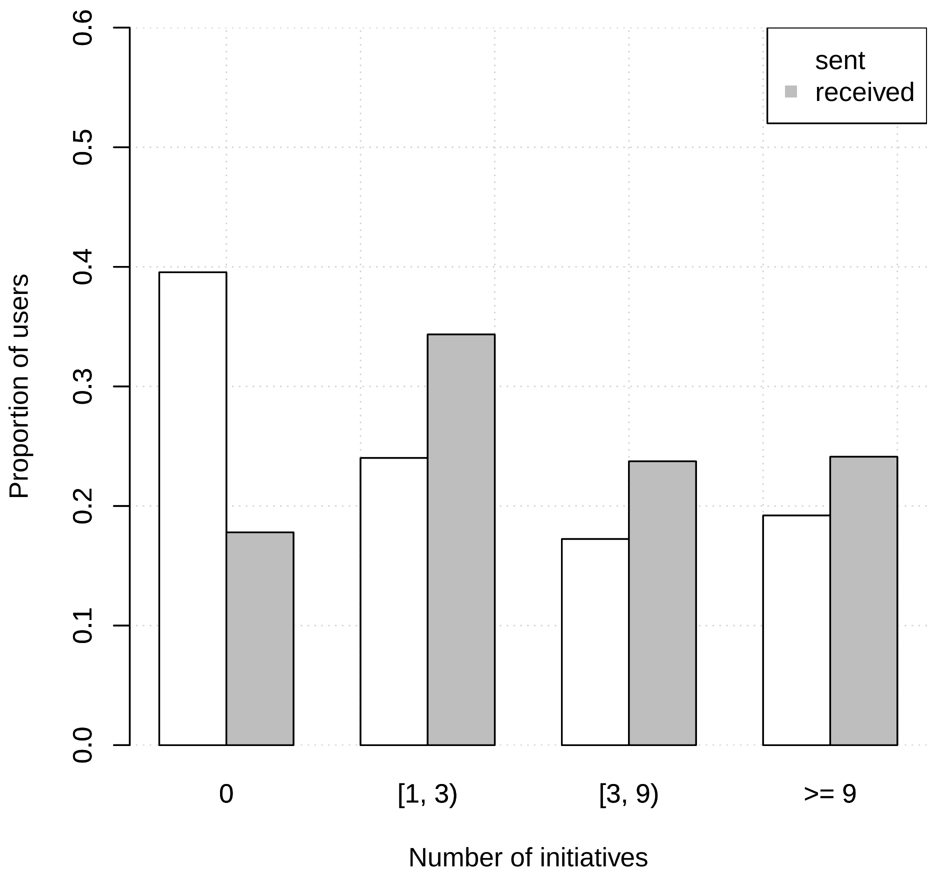
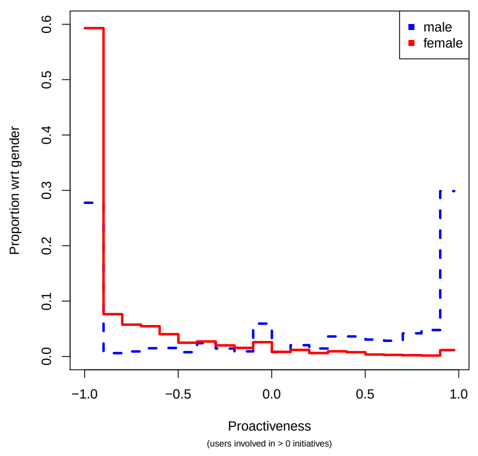
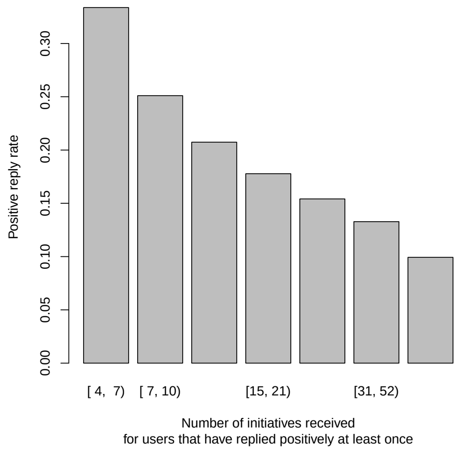


Figure 6: Distribution of users (active in training period) by initiatives sent and received during training period. Notice that users not involved in messages have been filtered out as described earlier in this section. Thus zero bars are underestimated with respect to the original data set.



(a) Proactiveness of users by gender.



(b) Success rate as a function of user popularity.

Figure 7: Effects of gender and popularity on messaging behavior. Figure types adapted from Pizzato et al. [52] where similar experiments were run on an Australian data set.

Range of initiatives received	$[1, 3)$	$[3, 9)$	≥ 9
Initiatives	5.3%	13.2%	81.5%
Reciprocated initiatives	24.3%	42.7%	33%

Table 8: Distribution of initiatives sent during training period by popularity of receiver. Heading denotes the range of initiatives received for users in each popularity group.

users even if much messaging data about reactive users are not available. Otherwise, proactive users may not find them and they may leave the service unsatisfied.

On the right-hand side of Figure 7, positive reply rate, the proportion of initiatives a user positively replies to, is visualized as a function of number of initiatives received. The graph seems to indicate that the more popular a user is, the less likely they are to respond positively. This is because of limited availability: a user can have meaningful communication with only so many users at one point in time. A popular user can also be more picky about who they respond to. It is important not to overburden popular users, as this may lead to churn on both sides of an initiative.

Finally, in Table 8, the distribution of initiatives and reciprocated initiatives is studied in the context of receiver popularity. The importance of balancing candidate popularity, so that not only popular users are recommended, is once again highlighted.

Although a majority of 81.5% of initiatives sent go to the most popular users, reciprocated initiatives (those that were positively replied to) are more evenly distributed across popularity groups. As the goal of a user of an online dating site is to presumably find a partner or friend, it is clear that a discussion must ensue before that can happen. The data seems to confirm that such discussion potential can be found regardless of receiver popularity, and that balancing recommendations across popularity groups is justified.

6 Comparative analysis of methods

Given algorithms of Section 3 and Section 4 and the data set described in Section 5, in this section, we evaluate how well the different recommenders perform, and in particular, how they compare. To that end, a variety of ways of measuring success on historical data are first listed (Section 6.1). Then the recommendations produced by the algorithms are put under scrutiny by comparing them under the different measuring criteria (Section 6.2). The section is concluded with a discussion of the results (Section 6.3).

The set-up of this evaluation was partly explained in the description of the data set (Section 5). In essence, the training portion of initiatives and all profile data are available for the recommenders to learn from. Each recommender produces a list of recommendations for each user, and the quality of these lists are evaluated based on different criteria, e.g., how well initiatives occurring in the test set are predicted in the lists.

An intuition of this setting (See Figure 4) is that the training data is the "past" and the test data is the "future". "Current time" is at the end of the first pause. Based on past behavior, we want to see how well the different methods are able to predict the future, very broadly speaking that is.

More elaborate evaluation protocols, such as traversing time step-by-step and updating recommender models in the process, could be used instead, and arguably, they may yield more accurate results [57]. However, the computational costs associated with such methods were deemed to be too high since our prototype recommender implementations do not support incremental updates to models.

A final note about the evaluation set-up. The recommendees to whom recommendations are given, as well as the candidates recommended, are limited to users who were involved in an initiative during the test period. Many users active in the training period have no actions during the test period (as witnessed in Table 7) and there is little reason to generate recommendations for them in order evaluate the algorithms. Also the candidates are filtered for the evaluation metrics to have more data to work on because much cannot be inferred from candidates who have not been involved in messages in the test period. This "post-filtering" step is performed *after* the core parts of the algorithms have finished to make sure that unfair advantage is not given to them.

6.1 Measures of success on historical data

Notation from Section 3.2 is used to define the measures. The measures introduced (with the exception of $POP@N$) have been widely applied in reciprocal recommender evaluations [28, 37, 46, 52, 67, 69].

Let X be the set of all users (involved in test set), M the initiatives of the test set, and N the desired size of the produced recommendation lists. Under a given algorithm, for each recommendee $x \in X$, a recommendation list R_x with maximum size of N is generated. These lists are then mass evaluated per algorithm using the hereinafter introduced criteria.

Precision at N ($P@N$),

$$P@N = \frac{\sum_x |R_x \cap M_{x,*}|}{\sum_x N} = \frac{\#\text{initiatives in recommendations}}{\#\text{recommendations}},$$

gives the proportion of the recommendations that correspond to occurred initiatives in the test set M regardless of reply type. It effectively measures the desirability of the recommended candidates from the viewpoint of the recommendee but not the candidate—it is in this sense a one-way measure.

Success rate at N ($S@N$),

$$S@N = \frac{\sum_x |R_x \cap M^+_{x,*}|}{\sum_x |R_x \cap M_{x,*}|} = \frac{\#\text{positive replies in recommendations}}{\#\text{initiatives in recommendations}},$$

(notice that $M^+_{x,*} \subseteq M_{x,*}$) is the proportion of (occurred) initiatives in the recommendations that are reciprocal (positively replied to). This is one of the most important measures we will be comparing. Reciprocated initiatives are a major goal because they translate to discussions between users of the service and may lead them further on into an actual relationship—the very reason users signed up on the site in the first place.

Failure rate at N ($F@N$),

$$F@N = \frac{\sum_x |R_x \cap M^-_{x,*}|}{\sum_x |R_x \cap M_{x,*}|} = \frac{\#\text{negative replies in recommendations}}{\#\text{initiatives in recommendations}}$$

(notice that $M^-_{x,*} \subseteq M_{x,*}$) is the proportion of (ocurrred) initiatives in the recommendations that have been rejected (negatively replied to). The smaller the proportion, the better, as repeated rejections may cause users to leave the service. Notice that this is not the opposite to success rate, as initiatives may also receive no reply.

Recall at N ($R@N$),

$$R@N = \frac{\sum_x |R_x \cap M^+_{x,*}|}{\sum_x |M^+_{x,*}|} = \frac{\#\text{positive replies in recommendations}}{\#\text{positive replies}},$$

is the proportion of all known reciprocated initiatives that occur in the recommendations [52]. Put differently, it measures how many of the known reciprocated initiatives in the entire test set are to be found in the recommendation lists. It measures how well the recommenders are able to predict positive interactions that actually occurred under the old system.

(Catalog) Coverage at N ($C@N$),

$$C@N = \frac{|\bigcup_x R_x|}{|X|} = \frac{\#\text{unique candidates}}{\#\text{unique users}},$$

measures the proportion of users that are present as candidates in the recommendation lists [22, 52]. In online dating, where users have limited availability, it is important not to overburden popular users and allow less popular users to surface in recommendations as well. Therefore this measure is important and ideally it should be close to one for a diverse set of recommendations.

Average popularity at N ($POP@N$),

$$POP@N = \frac{1}{\sum_x |R_x|} \sum_{x \in X} \sum_{r \in R_x} K_{train}(r),$$

where function $K_{train} : X \to \mathbb{N}$ gives the number of initiatives a user has received during the *training* period, is a novel measure used to compare biases different methods have towards popular users. The higher the value, the more the method favors popular users over less popular counterparts.

Besides the ones mentioned above, a great number of alternative measures have been used in the evaluation of recommender systems as well. The F_1 measure takes the harmonic mean between precision and recall (or in our case, $S@N$ and $R@N$) [38]. Another summarization of precision and recall is the Area Under the ROC Curve (AUC) [57].

If one wishes to take positions of candidates in the lists into consideration, a measure of ranking quality can be used. Normalized Discounted Cumulative Gain (NDCG) discounts positions logarithmically and has been used in several evaluations of reciprocal recommenders [14, 38, 63]. Other alternatives are Average Precision (AP) and Expected Reciprocal Rank (ERR) [14, 36]. Given a list of recommendations, users will typically focus on the first ones presented to them [26]. Thus, the ranking of recommendations matter. In our case, instead of directly measuring ranking quality, we used more easily interpretable measures and varied list size ($@N$) to capture patience.

If working with ratings data, e.g., attractiveness from 1 to 5, a more appropriate measure than precision may be MAE (mean absolute error) or NMAE (normalized mean absolute error) [10, 57].

A desired property of a modern recommender system is its ability to produce *serendipitous* recommendations: recommendations that are both surprising and useful to or liked by the recommendee [22, 42, 44, 57]. In the case of traditional product recommendation, the feeling of serendipity is eloquently described by Ge et al. [22]: "A feeling like seeing at a display window something you did not know that existed, but that perfectly fits your lifestyle". In a movie recommendation setting, an accurate recommender may suggest movies of the same genre and actors that a user has previously watched. A serendipitous recommender would suggest movies with actors unbeknown to the user, but that the user would later on discover he or she likes.

Evaluating serendipity is hard because of the subjectivity of the measure [22], and because a serendipitous recommender attempts to influence the behavior of the recommendee. Methods of evaluating serendipity on historical data have compared results produced by primitive recommenders, with high accuracy and low unexpectedness, to those produced by a target recommender [22, 44]. The usefulness of serendipity in reciprocal domains is unclear [52].

The recommenders introduced in this thesis can produce recommendations for everyone (although recommendation quality may be quite off). In cases where this is not true, user coverage can be measured over subjects [28].

Evaluating on historical data puts limits to what can be measured and the reliability of said measurements. Most prominently, the impact of a new recommender on user behavior cannot be assessed. As the data set has been gathered under the context of another system, either an old recommender or no recommender, when evaluating against it, we assume that user behavior would stay more or less similar had we deployed the new recommender before the test period [57].

A few dilemmas ensue as well. First of all, we are confined in our measures by the initiatives, reciprocations, and rejections occurring in the test set gathered under the old system [55]. For many recommendee-candidate pairs, it is impossible to deduce whether they would make a good match or not without an entry in the data. Perhaps they found each others profiles under the old system, but did not like each other enough to make contact? What if they were a good match, but under the old system, they were just unable to find each other? It is impossible to say with the data at hand. It is not surprising then that many measures have to discard a large portion of the recommendation lists in their evaluation.

The second point concerns the danger of local optimization. If we take the test data to be the ultimate ground truth towards which we should strive by all means, we may soon find ourselves following in the footsteps of the old system. This is of course pointless because the entire point of developing and evaluating these recommenders is to come up with a better one [35].

6.2 Results

Seven recommender systems were compared on criteria, introduced in the previous section, measuring desirable traits of an online dating recommender. Alongside the five main methods (RECON, RECON with negative preferences,

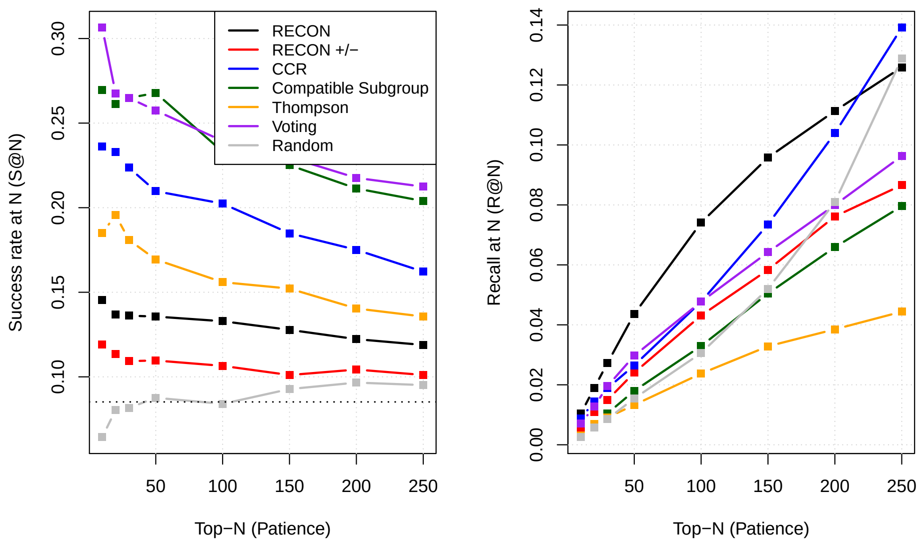


Figure 8: Comparison of success rate $S@N$ and recall $R@N$ by recommendation list size N. Dashed line represents the base rate in the test set.

CCR, multiple compatible subgroups, and Thompson sampling), a random method was evaluted as a baseline, and an ensemble of methods named 'voting' was evaluated as well.

The random method was implemented as a sort of a control to confirm that the results obtained are meaningful. For each active user, it generates a random list of unmet candidates of opposite sex who are within the explicitly defined age preference range of the active user and the active user is in the candidate's respective range.

The voting ensemble allowed each of the three methods, CCR, multiple compatible subgroups, and Thompson sampling, to vote on the position a candidate should get on the final recommendation list. The votes were then averaged per candidate and the final ranking produced.

Recommendation lists of maximum size 10, 20, 30, 50, 100, 200, and 250, corresponding to different patience levels, were evaluated and are illustrated in the plots that follow.

In Figure 8 the success rate and recall of the methods are compared. As is to be expected, success rate falls and recall increases as a function of N. The base success rate is marked as a thin dashed black line and reflects the proportion of initiatives in the test set that were positively replied to. Notice that recall at N ($R@N$) is inherently bounded above by list size N. This is because if a user is known to have initiated 20 reciprocated initiatives, with $N = 10$, only half of them could be returned in the best case scenario.

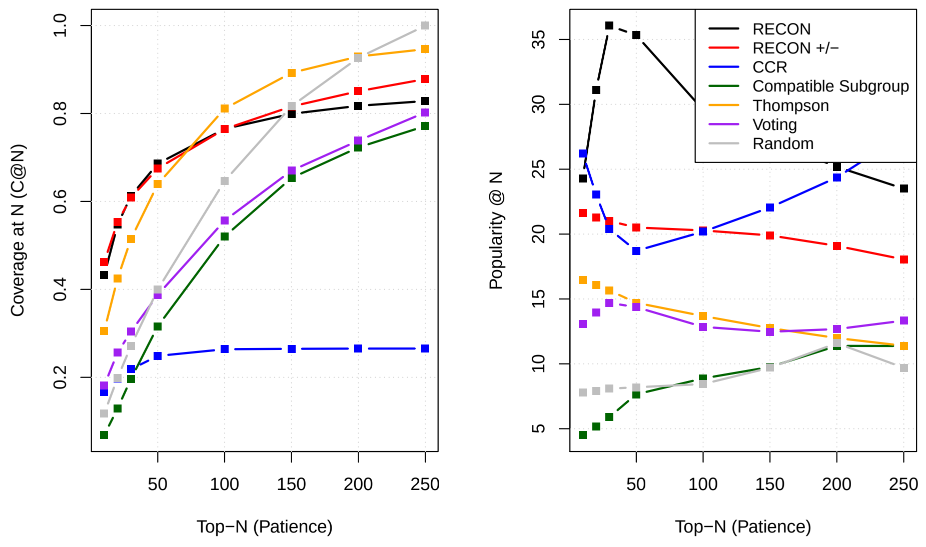


Figure 9: Comparison of coverage $C@N$ and average popularity $POP@N$ by recommendation list size N.

In Figure 9 coverage and average popularity are compared. As expected, coverage increases with list size, with the exception of CCR that plateaus at $N = 50$. Interestingly, coverages start off at quite different levels at $N = 10$. Average popularity exhibits no consistent growth or decay behavior throughout the different methods. Be that as it may, the methods do reside at different levels with respect to each other.

In Figure 10 precision and failure rate of the methods are compared. These measures are less important to our discussion but still worth studying. Only RECON with negatives can truly prevent rejections, as it and the random method are the only to achieve rates below base failure rate. The ability to predict initiatives occurring during the test period is low throughout all methods, as witnessed by the low values of precision.

6.3 Discussion

Given the results of the previous section, it is clear that different methods shine on different criteria and that a trade-off has to be made when choosing one.

The clear winners of success rate are the voting ensemble and multiple compatible subgroups method. CCR and Thompson are in the mid-tier, and RECONs perform worst but still above base rate. One reason RECONs' peformance is not up to par with the rest may be because of their minimal

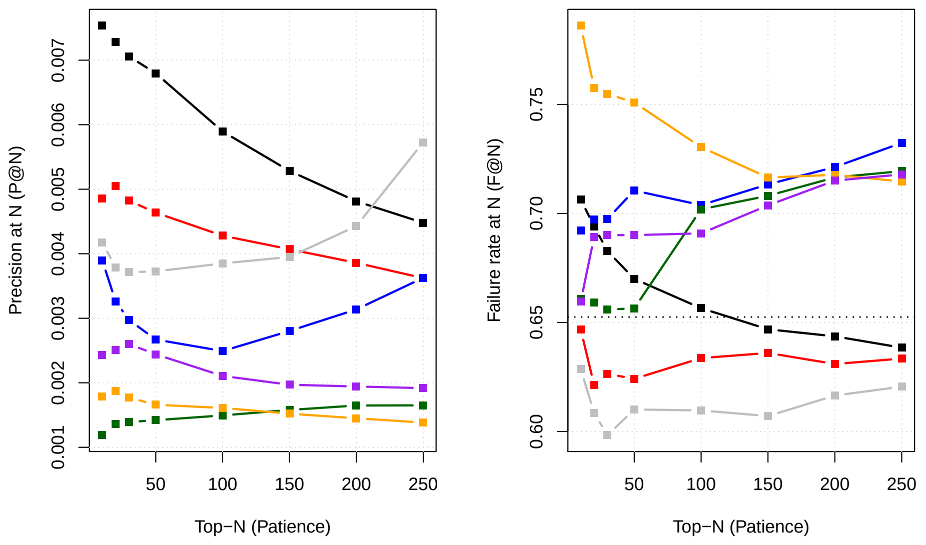


Figure 10: Comparison of precision $P@N$ and failure rate $F@N$ by recommendation list size N. Dashed line represents the base rate in the test set.

support for cold-start users. Rather unexpectedly, RECON with negatives performs worse than RECON which goes against the results of the original paper [55]. Why exactly this is, is not clear to the author of this thesis.

The high success rate of the multiple compatible subgroups method lends support to the idea that recommending to users based on the demography they belong to can lead to good recommendations. A concern with this approach had to do with its inability to provide truly personalized recommendations. For example, if a user's taste differs radically from his or her cohort, produced recommendations may not be to his or her liking. However, at least on average, and on offline data, recommendation based on subgroups seems viable. It is worth noting that success rate gains are also most likely procured by the method handling cold-start users appropriately, at least compared to RECON and CCR.

In recall, the recommenders start ($N = 10$) at approximately the same point. As recommendation list size increases, RECON gains an upper hand, and the growth of Thompson sampling stagnates with respect to others, which may perhaps be caused by its randomization components.

On the other hand, when it comes to coverage, Thompson sampling seems to benefit from randomization. RECONs perform very well on coverage as well. CCR's growth stagnates on coverage hinting at some inherent limit to the portion of users it can recommend. This is worrisome because as was discussed in Section 2.2.1, users have limited availability, and it is imperative to have a diverse set of candidates in recommendations.

Recall is rather hard to interpret in our context because it measures the portion of reciprocated initiatives that occurred in recommendations. A method with bad recall does not necessarily produce irrelevant recommendations. It may just be that its candidates do not correspond to occurred initiatives in the data set at hand. On the contrary, a method with good recall is able to provide recommendations that correspond to actually occurred reciprocated initiatives. This lends support to the idea that the method is providing relevant recommendations, at least from the point of view of the old system.

It is interesting that the ordering of methods for coverage is almost the reverse of what is for success rate, i.e., a method with high success rate has low coverage. We took notice of this phenomenon when developing our Thompson sampling approach: increases in coverage tend to go hand in hand with decreases in success rate. Perhaps by increasing coverage, we begin to cover candidates that never reply positively, and this hurts success rate? It would be interesting to study this point further, and to understand if both quantities can be increased simultaneously or not.

The average popularity of candidates recommended by the methods varies quite a lot. Multiple compatible subgroups method tends to recommend less popular users, while RECONs and CCR favor popular candidates. Thompson sampling and voting ensemble stay on middle ground.

There is uncertainty on how the popularity measure $POP@N$ should be interpreted. Average popularity being high is presumably undesirable, as it indicates that only popular users are being recommended. Should one, then, optimize for as low popularity as possible, or is there a middle ground to be found instead? Perhaps measuring the spread of candidates' popularity could be of utility as well when comparing reciprocal recommenders.

With regards to failure rate, only RECON with negatives seems to prevent negative replies from occurring compared to the baseline. Most methods actually incur worse failure rate than the baseline (Thompson sampling especially). On the other hand, users accepting recommendations from these other methods and sending initiatives accordingly will at least more often receive a negative response compared to none at all.

Since RECON, Thompson sampling, and multiple compatible subgroups do not explicitly consider negative replies, it is not surprising that their failure rate is not lowered. What is a bit surprising, is that CCR is unable to reduce failure rate compared to the baseline even though it takes negative replies into account when ranking users by support.

Lastly, RECON and RECON with negatives perform best in precision, CCR resides in the middle, and the rest come last. In a situation where reciprocity is less important and the attractiveness of recommendations matters solely from the viewpoint of the recommendee, RECON seems like a good choice. However, it is the opinion of the author of this thesis that although such recommendations may initially prove interesting to the user, in the long run, optimizing only for precision may lead to recommending candidates that do not reciprocate. Therefore the recommendee may become bored and/or frustrated because of lack of response or abundance of negative response. At least in online dating, success rate may be a more useful quantity to aim for instead.

Although Pizzato et al. [55] argue for the usefulness of avoiding negative replies, it is the opinion of the author of this thesis that failure rate is not perhaps such an important quantity to optimize for. If we ignore reciprocated initiatives for a moment, the two other options are: negative reply and no reply. It is the author's opinion that neither is really a better outcome than the other. Conversely, some time-conscious users may appreciate an honest "no" to no response at all.

If accuracy of predictions, i.e., success rate, is the most important quantity, then voting ensemble or multiple compatible subgroups should be chosen. Voting ensemble seems to have a slight upper hand in recall, coverage, and precision, so it may be the better choice albeit it is a hard-to-interpret black box model. If model interpretation matters, multiple compatible subgroups could be preferred. In addition to ease of interpretation, multiple compatible subgroups method also has lower average popularity than voting ensemble.

In recent recommender system research, it has been argued that focus should not lie solely on prediction accuracy because recommendations based purely on accuracy may not be useful to the user and may even hurt the user experience [42].

Looking at the results of the previous section, it would seem as if Thompson sampling provides balanced performance in different measures. To be specific, a decent success rate is obtained without compromising much coverage or favoring popular users. This means that a diverse set of users are recommended while still retaining reciprocal interest. Moreover, as explained in Section 4, it would seem as if the trade-off between coverage and success rate (+ recall) can be tuned by changing the number of clusters used.

A concern with Thompson sampling is its low recall at large N. On the other hand, as users tend to have limited patience and focus on the first items presented to them [53], the smallest list sizes may matter the most, and the difference in recall across methods is not so pronounced when $N < 50$.

An ideal method would most likely be one that attains high success rate, very high coverage, and low popularity. The role of recall, precision, and failure rate might be less important. Ideally, we would like Thompson sampling to have even higher success rate and coverage than it currently does.

It is important not to read these measures as absolute values, but rather as a comparative study between the different methods [57]. Just because voting ensemble attains a success rate of 25% at $N = 50$ on historical data, does not mean this result necessarily translates over to a real world situation. Indeed, evaluation on historical offline data should be done before deployment at design time, when developing and comparing algorithms [57]. After deploying the recommender system, an on-line evaluation with real users should be performed to verify whether any true improvement ocurred or not [57].

Furthermore, although we introduced several criteria to measure the quality of recommendations, it is unclear which of them, if any, lead to core business values: good user experience, users finding partners, as well as paying and loyal customers. The only way to verify such questions is by performing controlled experiments on a live system and/or doing focused user studies with possible questionnaires [57]. Have we had more time and resources, we would have performed split-testing on our novel Thompson sampling method to verify whether it is useful or not and whether low recall causes problems.

Until recently, published evaluations of reciprocal recommenders have tended to be based solely on historical data. In recent work by Kryzwicki et al. [35], the experience of deploying a reciprocal recommender system on a large, real-world online dating site is reported. Several interesting discoveries and lessons are shared. Firstly, they found out that performance of the methods on real-world experiments were consistent with those done on historical data. This lends support to the idea that comparing methods on historical data is a worthwhile endeavor, allowing one to pre-filter methods that are worth further testing in a production system. Secondly, they found that key metrics stay at a consistent level months after deployment, indicating that quality of recommendations is not a one-off phenomenon.

The astute reader may at this point question the very need for *reciprocal* recommenders—do they really perform any better than classical collaborative filtering or content-based recommenders? This question was not studied in this thesis, but has been evaluated in previous research. Li et al. [38] showed that CCR, RECON, and especially their own method MEET have benefits over state-of-art (of year 2012) collaborative filtering methods in reciprocal domains (online dating and a recruitment system to be precise). Zhao et al. [69] compared baseline collaborative filtering to their hybrid reciprocal recommender and noticed improvement in success rate and recall. Pizzato et al. [53] compared RECON to a variant of the method that considers only one-way preference, effectively rendering the method to be a typical content-based recommender. They noticed clear improvements in success rate and recall.

7 Conclusion

In this thesis, a special class of recommender systems for the domain of online dating was presented, namely reciprocal recommenders. Based on earlier work, a new reciprocal recommender was developed that attempts to address issues not resolved by methods described in previous literature.

Our new method alongside four other introduced reciprocal recommenders were evaluated on a historical data set gathered from a real-world online dating service. The results show that no method is a clear winner across the board; depending on the criteria being measured, different methods come on top.

What was left unclear is the importance of the different measuring criteria, i.e., how does success rate, recall, coverage, and average popularity affect user experience, user loyalty, profits earned, as well as the likelihood of users finding partners. Moreover, it is unclear if any of the (presented) offline measures can predict such qualities, albeit some existing research provides careful support for such an idea [35].

Future research should address these concern by performing live controlled experiments on actual dating services with real users. Indeed, the novel Thompson sampling approach introduced in this thesis requires more research and especially real-world validation before anything conclusive can be said about its recommendation performance. The author of this thesis would point anyone interested in such research to the excellent work by Kohavi et al. [32], which thoroughly covers the design and challenges of controlled experiments on the web.

In addition to the offline comparison of methods and the development of a new method, another contribution of this thesis is the exploratory analysis conducted in Section 5.2 that lends additional support to the claims made in Section 2.2.1 and originally by Pizzato et al. [52]. The data give support to the abundance of cold-start users and proactive-reactive roles in online dating. It also shows that reciprocated initiatives occur across popularity groups and that popular users tend to respond positively less often, supporting the idea of users having limited availability.

Criticism has been presented against matchmaking systems and online dating in general. Finkel et al. [16] claim that access to a large pool of candidates can cause people to make inferior decisions when choosing a partner and that "...online dating profiles reduces three-dimensional people to two-dimensional displays of information...". They also take note of the fact that sometimes connections made via messaging do no translate well to face-to-face connections outside the dating service. Especially matchmaking systems are criticised for being based around principles that are not so important to "relationship well-being", and it is pointed out that compelling evidence has not been presented to support the effectiveness of matchmaking algorithms.

The author of this thesis agrees that the effectiveness of matchmaking algorithms needs further study, but on the other hand, it seems from our results and previously published results that recommenders for online dating can do at least better than baseline random recommendation.

Future development of reciprocal recommender methods could involve infusing content from external sources to improve utility of recommendations. For example, the mobile dating application Tinder, which has disrupted the online dating market with simple user profiles and like-dislike mechanics, utilizes a user's Facebook profile to display mutual friends and interests with presented candidates. Such sort of infusing could also be taken advantage of in a recommender to provide better user models for recommendation.

Another line of research could involve applying psychological and sociological studies of online dating behavior to the development of recommendation models. Encoding such assumptions into the prediction models could allow better utilization of gathered data in furtherance of better recommendations.

Regarding the recommenders presented in Section 3 and Section 4, ideas for future development of each were presented in their respective subsections. To squeeze even more performance from these methods, heavier feature selection and engineering of profile attributes could be performed. Especially any information gained from profile photos and free-text descriptions could be of significant use.

The value of context-aware recommenders, as well as serendipity and transparency of recommendations, is unclear for the domain of online dating. Transparency is problematic because of privacy issues: explaining recommendations given may require divulging preferences of the candidates involved [52]. Serendipity may produce recommendations that go against the preferences of the user, which may lead to distrust in the system [52]. Context-aware recommenders take into consideration the contextual situation of a user, e.g., physical location or time of day. How contextual information could benefit online dating, may be an interesting line of research to pursue.

IBISWorld forecasts that in the U.S. "niche and mobile-based dating services will drive revenue growth" in 2015 in the dating service business [24]. Developers of novel dating services can learn from research done in online dating to develop services that allow people to find partners more effectively than ever before. The author of this thesis believes that reciprocal recommender systems can be a valuable part of such services.

References

[1] *Dictionary.com: Definition of Reciprocal.* `http://dictionary.reference.com/browse/reciprocal`, visited on 2015-05-09.

[2] *Statistics Finland: Finland in Figures (2013).* `http://www.stat.fi/tup/suoluk/suoluk_vaesto_en.html`, visited on 2015-02-23.

[3] Adomavicius, Gediminas and Tuzhilin, Alexander: *Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions.* IEEE Transactions on Knowledge and Data Engineering, 17(6):734–749, June 2005.

[4] Agrawal, Manish, Karimzadehgan, Maryam, and Zhai, ChengXian: *An Online News Recommender System for Social Networks.* In *Proceedings of the Workshop on Search in Social Media*, 2009, ISBN 3838315642.

[5] Akehurst, Joshua, Koprinska, Irena, Yacef, Kalina, Pizzato, Luiz, Kay, Judy, and Rej, Tomasz: *CCR - A Content-Collaborative Reciprocal Recommender for Online Dating.* In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, pages 2199–2204, 2011.

[6] Akehurst, Joshua, Koprinska, Irena, Yacef, Kalina, Pizzato, Luiz, Kay, Judy, and Rej, Tomasz: *Explicit and Implicit User Preferences in Online Dating.* In *New Frontiers in Applied Data Mining*, pages 15–27. 2012.

[7] Amatriain, Xavier, Pujol, Josep M, and Oliver, Nuria: *I Like It... I Like It Not: Evaluating User Ratings Noise in Recommender Systems.* In *User Modeling, Adaptation, and Personalization*, pages 247–258. 2009.

[8] Auer, Peter, Cesa-Bianchi, Nicolò, and Fischer, Paul: *Finite-time Analysis of the Multiarmed Bandit Problem.* Machine Learning, 47(2-3):235–256, 2002.

[9] Blecker, Thorsten, Kreutler, Gerold, Abdelkafi, Nizar, and Friedrich, Gerhard: *An Advisory System for Customers' Objective Needs Elicitation in Mass Customization.* In *Proceedings of the 4th Workshop on Information Systems for Mass Customization (ISMC 2004)*, pages 1–10. 2004.

[10] Brozovsky, Lukas and Petricek, Vaclav: *Recommender System for Online Dating Service.* arXiv preprint cs/0703042, 2007. `http://arxiv.org/abs/cs/0703042`.

[11] Burke, Robin: *Hybrid Web Recommender Systems.* In *The Adaptive Web*, pages 377–408. Springer, 2007.

[12] Chapelle, Olivier and Li, Lihong: *An Empirical Evaluation of Thompson Sampling.* In *Advances in Neural Information Processing Systems*, pages 2249–2257, 2011.

[13] Chen, Lin, Nayak, Richi, and Xu, Yue: *Improving Matching Process in Social Network.* In *2010 IEEE International Conference on Data Mining Workshops*, pages 305–311, December 2010.

[14] Diaz, Fernando, Metzler, Donald, and Amer-Yahia, Sihem: *Relevance and Ranking in Online Dating Systems.* In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 66–73, 2010.

[15] Domingos, Pedro: *A Few Useful Things to Know About Machine Learning.* Communications of the ACM, 55(10):78–87, October 2012.

[16] Finkel, Eli J., Eastwick, Paul W., Karney, Benjamin R., Reis, Harry T., and Sprecher, Susan: *Online Dating: A Critical Analysis From the Perspective of Psychological Science.* Psychological Science in the Public Interest, 13(1):3–66, March 2012.

[17] Fiore, Andrew T., Taylor, Lindsay Shaw, Mendelsohn, G.A., and Hearst, Marti: *Assessing Attractiveness in Online Dating Profiles.* In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '08)*, pages 797–806, April 2008.

[18] Forbes: *Netflix Yields $131 Value With User Recommendation Tools.* `http://www.forbes.com/sites/greatspeculations/2012/04/17/netflixs-yields-131-value-with-user-recommendation-tools/`, visited on 2015-05-10.

[19] Fortune: *Amazon's recommendation secret.* `http://fortune.com/2012/07/30/amazons-recommendation-secret/`, visited on 2015-05-10.

[20] Freyne, Jill, Jacovi, Michal, Guy, Ido, and Geyer, Werner: *Increasing Engagement Through Early Recommender Intervention.* In *Proceedings of the 3rd ACM Conference on Recommender Systems (RecSys '09)*, pages 85–92, October 2009.

[21] Friedman, Nir and Goldszmidt, Moises: *Building Classifiers using Bayesian Networks.* In *Proceedings of the 13th National Conference on Artificial Intelligence*, pages 1277–1284, 1996.

[22] Ge, Mouzhi, Delgado-Battenfeld, Carla, and Jannach, Dietmar: *Beyond Accuracy: Evaluating Recommender Systems by Coverage and Serendipity.* In *Proceedings of the 4th ACM Conference on Recommender Systems (RecSys '10)*, pages 257–260, 2010.

[23] Hancock, Jeffrey T., Toma, Catalina, and Ellison, Nicole: *The Truth about Lying in Online Dating Profiles.* In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '07)*, pages 449–452, April 2007.

[24] IBISWorld: *Dating Services in the US: Market Research Report (April 2015).* `http://www.ibisworld.com/industry/default.aspx?indid=1723`, visited on 2015-05-09.

[25] Jambor, Tamas and Wang, Jun: *Optimizing Multiple Objectives in Collaborative Filtering.* In *Proceedings of the 4th ACM Conference on Recommender Systems (RecSys '10)*, pages 55–62, September 2010.

[26] Joachims, Thorsten, Granka, Laura, Pan, Bing, Hembrooke, Helene, and Gay, Geri: *Accurately Interpreting Clickthrough Data as Implicit Feedback.* In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '05)*, pages 154–161, 2005.

[27] Karpathy, Andrej and Fei-Fei, Li: *Deep Visual-Semantic Alignments for Generating Image Descriptions.* December 2014. `http://arxiv.org/abs/1412.2306`.

[28] Kim, Yang Sok, Krzywicki, Alfred, Wobcke, Wayne, Mahidadia, Ashesh, Compton, Paul, Cai, Xiongcai, and Bain, Michael: *Hybrid Techniques to Address Cold Start Problems for People to People Recommendation in Social Networks.* In *PRICAI 2012: Trends in Artificial Intelligence*, pages 206–217. 2012.

[29] Kim, Yang Sok, Mahidadia, Ashesh, Compton, Paul, Cai, Xiongcai, Bain, Mike, Krzywicki, Alfred, and Wobcke, Wayne: *People Recommendation Based on Aggregated Bidirectional Intentions in Social Network Site.* In *Knowledge Management and Acquisition for Smart Systems and Services*, volume 6232 of *Lecture Notes in Computer Science*, pages 247–260. 2010.

[30] Kim, Yang Sok, Mahidadia, Ashesh, Compton, Paul, Krzywicki, Alfred, Wobcke, Wayne, Cai, Xiongcai, and Bain, Michael: *People-to-people recommendation using multiple compatible subgroups.* In *AI 2012: Advances in Artificial Intelligence*, pages 61–72. 2012.

[31] Kohavi, Ron: *Scaling Up the Accuracy of Naive-Bayes Classifiers: a Decision-Tree Hybrid.* In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining*, volume 7, pages 202–207, 1996.

[32] Kohavi, Ron, Longbotham, Roger, Sommerfield, Dan, and Henne, Randal M.: *Controlled experiments on the web: survey and practical guide.* Data Mining and Knowledge Discovery, 18(1):140–181, 2009.

[33] Korenius, Tuomo, Laurikkala, Jorma, Järvelin, Kalervo, and Juhola, Martti: *Stemming and Lemmatization in the Clustering of Finnish text Documents.* In *Proceedings of the 13th ACM Conference on Information and Knowledge Management (CIKM '04)*, pages 625–633, November 2004.

[34] Krzywicki, Alfred, Wobcke, Wayne, Cai, Xiongcai, Mahidadia, Ashesh, Bain, Michael, Compton, Paul, and Kim, Yang Sok: *Interaction-Based Collaborative Filtering Methods for Recommendation in Online Dating.* In *Web Information Systems Engineering (WISE 2010)*, volume 6488 of *Lecture Notes in Computer Science*, pages 342–356. 2010.

[35] Krzywicki, Alfred, Wobcke, Wayne, Kim, Yang Sok, Cai, Xiongcai, Bain, Michael, Compton, Paul, and Mahidadia, Ashesh: *Evaluation and Deployment of a People-to-People Recommender in Online Dating.* In *Proceedings of the 26th Annual Conference on Innovative Applications of Artificial Intelligence*, pages 2914–2921, September 2014.

[36] Kunegis, Jérôme, Gröner, Gerd, and Gottron, Thomas: *Online Dating Recommender Systems: The Split-complex Number Approach.* Proceedings of the 4th ACM RecSys Workshop on Recommender Systems and the Social Web, pages 37–44, 2012.

[37] Kutty, Sangeetha, Nayak, Richi, and Chen, Lin: *A people-to-people matching system using graph mining techniques.* World Wide Web, 17(3):311–349, 2014.

[38] Li, Lei and Li, Tao: *MEET: A Generalized Framework for Reciprocal Recommender Systems.* In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management (CIKM '12)*, pages 35–44, 2012.

[39] Mäkelä, Eetu: *Combining a REST Lexical Analysis Web Service with SPARQL for Mashup Semantic Annotation from Text.* In *The Semantic Web: ESWC 2014 Satellite Events*, pages 424–428, 2014.

[40] Malinowski, Jochen, Keim, Tobias, Wendt, Oliver, and Weitzel, Tim: *Matching People and Jobs: A Bilateral Recommendation Approach.* In *Proceedings of the 39th Annual Hawaii International Conference on System Sciences (HICSS '06)*, volume 6, pages 137c–137c, 2006.

[41] Match.com: *Match Fact Sheet*, 2015. `http://match.mediaroom.com/download/Match+Fact+Sheet_2015_20th.pdf`, visited on 2015-05-09.

[42] McNee, Sean M., Riedl, John, and Konstan, Joseph A.: *Being Accurate is Not Enough: How Accuracy Metrics have hurt Recommender Systems.* In *Extended Abstracts on Human Factors in Computing System (CHI '06)*, pages 1097–1101, 2006.

[43] McPherson, Miller, Smith-Lovin, Lynn, and Cook, James M: *Birds of a Feather: Homophily in Social Networks.* Annual Review of Sociology, 27(2001):415–444, 2001.

[44] Murakami, Tomoko, Mori, Koichiro, and Orihara, Ryohei: *Metrics for Evaluating the Serendipity of Recommendation Lists.* In *New Frontiers in Artificial Intelligence*, pages 40–46. 2008.

[45] Murtagh, Fionn and Legendre, Pierre: *Ward's Hierarchical Agglomerative Clustering Method: Which Algorithms Implement Ward's Criterion?* Journal of Classification, 31(3):274–295, October 2014.

[46] Nayak, Richi, Zhang, Meng, and Chen, Lin: *A Social Matching System for an Online Dating Network: A Preliminary Study.* In *IEEE International Conference on Data Mining (ICDM 2010)*, pages 352–357, 2010.

[47] O'Donovan, John and Smyth, Barry: *Trust in Recommender Systems.* In *Proceedings of the 10th International Conference on Intelligent User Interfaces (IUI '05)*, pages 167–174, January 2005.

[48] Pang, Bo and Lee, Lillian: *Opinion Mining and Sentiment Analysis.* Foundations and Trends in Information Retrieval, 2(1–2):1–135, January 2008.

[49] Park, Yoon Joo: *An Adaptive Match-Making System reflecting the explicit and implicit preferences of users.* Expert Systems with Applications, 40(4):1196–1204, March 2013.

[50] Pearl, Judea: *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* 1988, ISBN 978-1558604797.

[51] Pew Research Center: *5 facts about online dating.* `http://www.pewresearch.org/fact-tank/2015/04/20/5-facts-about-online-dating/`, visited on 2015-05-09.

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

[53] Pizzato, Luiz, Rej, Tomek, Chung, Thomas, Koprinska, Irena, and Kay, Judy: *RECON: A Reciprocal Recommender for Online Dating.* In *Proceedings of the 4th ACM Conference on Recommender Systems (RecSys '10)*, pages 207–214, 2010.

[54] Pizzato, Luiz, Rej, Tomek, Chung, Thomas, Yacef, Kalina, Koprinska, Irena, and Kay, Judy: *Reciprocal Recommenders.* In *Proceedings of the*

8th Workshop on Intelligent Techniques for Web Personalization and Recommender Systems (ITWP 2010), pages 20–24, 2010.

[55] Pizzato, Luiz Augusto, Rej, Tomek, Yacef, Kalina, Koprinska, Irena, and Kay, Judy: *Finding Someone You Will Like and Who Won't Reject You.* In *User Modeling, Adaption and Personalization*, pages 269–280. 2011.

[56] Pizzato, Luiz Augusto and Silvestrini, Cameron: *Stochastic Matching and Collaborative Filtering to Recommend People to People.* In *Proceedings of the 5th ACM Conference on Recommender Systems (RecSys '11)*, pages 341–344, October 2011.

[57] Ricci, Francesco, Rokach, Lior, Shapira, Bracha, and Kantor, Paul B: *Recommender Systems Handbook.* Springer, 2011, ISBN 978-0-387-85819-7.

[58] Rish, Irina: *An empirical study of the naive Bayes classifier.* In *IJCAI 2001 Workshop on Empirical Methods in Artificial Intelligence*, pages 41–46, 2001.

[59] Schafer, J Ben, Konstan, Joseph A, and Riedl, John: *E-Commerce Recommendation Applications.* In *Applications of Data Mining to Electronic Commerce*, pages 115–153, 2001.

[60] Schein, Andrew I., Popescul, Alexandrin, Ungar, Lyle H., and Pennock, David M.: *Methods and Metrics for Cold-Start Recommendations.* In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '02)*, pages 253–260, August 2002.

[61] Sirovich, Lawrence and Kirby, Michael: *Low-Dimensional Procedure for the Characterization of Human Faces.* Journal of the Optical Society of America, 4(3):519–524, March 1987.

[62] Tan, Pang Ning, Steinbach, Michael, and Kumar, Vipin: *Introduction to Data Mining.* Pearson Addison Wesley, 1st edition, 2006, ISBN 978-0321321367.

[63] Wang, Tingting, Liu, Hongyan, He, Jun, Jiang, Xuan, and Du, Xiaoyong: *Predicting New User's Behavior in Online Dating Systems.* In *Advanced Data Mining and Applications*, volume 7121 of *Lecture Notes in Computer Science*, pages 266–277. 2011.

[64] Ward, Joe H. Jr.: *Hierarchical Grouping to Optimize an Objective Function.* Journal of the American Statistical Association, 58(301):236–244, 1963.

[65] Wasserman, Larry: *All of Statistics: A Concise Course in Statistical Inference.* Springer, 1st edition, 2004, ISBN 978-0387402727.

[66] Wikipedia: *Comparison of online dating websites.* `http://en.wikipedia.org/wiki/Comparison_of_online_dating_websites`, visited on 2015-05-09.

[67] Xia, Peng, Liu, Benyuan, Sun, Yizhou, and Chen, Cindy: *Reciprocal Recommendation System for Online Dating.* January 2015. `http://arxiv.org/abs/1501.06247`.

[68] Yu, Hongtao, Liu, Chaoran, and Zhang, Fuzi: *Reciprocal Recommendation Algorithm for the Field of Recruitment.* Journal of Information & Computational Science, 8(16):4061–4068, 2011.

[69] Zhao, Kang, Wang, Xi, Yu, Mo, and Gao, Bo: *User Recommendations in Reciprocal and Bipartite Social Networks – An Online Dating Case Study.* IEEE Intelligent Systems, 29(2):27–35, March 2014.

[70] Zhou, Renjie, Khemmarat, Samamon, and Gao, Lixin: *The Impact of YouTube Recommendation System on Video Views.* In *Proceedings of the 10th Annual Conference on Internet Measurement (IMC '10)*, pages 404–410, November 2010.